Twitter4J

Version: ee322a7e54ad73591bfba3e828fc250a603d885e

Parents:

b2e77802a1e5742d6384001335a556e488b57a7f

ce93bb20aa0d4bd8edc5e77174ae2bd0d827f476

Merge base:

747982dfdccd6d1b2ef4857720fe289455a435a1

twitter4j/twitter4j-core/src/main/java/twitter4j/api/FriendsFollowersResources.java

Chunk 1: (version 2/commentary, method interface)

```
     *
<<<<<<< HEAD
=======
     * @param userId The ID of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @return list of friends
     * @throws TwitterException when Twitter service or network is unavailable
     * @see <a href="https://dev.twitter.com/docs/api/1.1/get/friends/list">GET friends/list
| Twitter Developers</a>
     * @since Twitter4J 4.0.2
     */
    PagableResponseList<User> getFriendsList(long userId, long cursor, int count) throws
TwitterException;

    /**
     * Returns a cursored collection of user objects for every user the specified user is
following (otherwise known as their "friends").<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the next_cursor value in subsequent requests. See <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls https://api.twitter.com/1.1/friends/list.json
     *
>>>>>>> ce93bb20aa0d4bd8edc5e77174ae2bd0d827f476
     * @param screenName The screen name of the user for whom to return results for.
```

```
     *
     * @param userId The ID of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @return list of friends
```

```
     * @throws TwitterException when Twitter service or network is unavailable
     * @see <a href="https://dev.twitter.com/docs/api/1.1/get/friends/list">GET friends/list
| Twitter Developers</a>
     */
    PagableResponseList<User> getFriendsList(long userId, long cursor, int count) throws
TwitterException;

    /**
     * Returns a cursored collection of user objects for every user the specified user is
following (otherwise known as their "friends").<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the next_cursor value in subsequent requests. See <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls https://api.twitter.com/1.1/friends/list.json
     *
     * @param screenName The screen name of the user for whom to return results for.
```

Chunk 2: (version 2/commentary)

```
     * @see <a href="https://dev.twitter.com/docs/api/1.1/get/friends/list">GET friends/list
| Twitter Developers</a>
<<<<<<< HEAD
=======
     * @since Twitter4J 4.0.2
>>>>>>> ce93bb20aa0d4bd8edc5e77174ae2bd0d827f476
     */
```

```
     * @see <a href="https://dev.twitter.com/docs/api/1.1/get/friends/list">GET friends/list
| Twitter Developers</a>
     * @since Twitter4J 3.0.2
     */
```

Chunk 3: (new code /commentary, method interface)

```
    /**
<<<<<<< HEAD
     * Returns a cursored collection of user objects for users following the specified
user.<br>
=======
     * Returns a cursored collection of user objects for every user the specified user is
following (otherwise known as their "friends").<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the next_cursor value in subsequent requests. See <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls https://api.twitter.com/1.1/friends/list.json
     *
     * @param userId The ID of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @param skipStatus When set to either true, statuses will not be included in the
returned user objects.
     * @param includeUserEntities The user object entities node will be disincluded when set
to false.
     * @return list of friends
```

```
     * @throws TwitterException when Twitter service or network is unavailable
     * @see <a href="https://dev.twitter.com/docs/api/1.1/get/friends/list">GET friends/list
| Twitter Developers</a>
     * @since Twitter4J 4.0.2
     */
    PagableResponseList<User> getFriendsList(long userId, long cursor, int count,
            boolean skipStatus, boolean includeUserEntities) throws TwitterException;

    /**
     * Returns a cursored collection of user objects for every user the specified user is
following (otherwise known as their "friends").<br>
>>>>>>> ce93bb20aa0d4bd8edc5e77174ae2bd0d827f476
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual
```

```
    /**
     * Returns a cursored collection of user objects for every user the specified user is
following (otherwise known as their "friends").<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the next_cursor value in subsequent requests. See <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls https://api.twitter.com/1.1/friends/list.json
     *
     * @param screenName The screen name of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @return list of friends
     * @throws TwitterException when Twitter service or network is unavailable
     * @see <a href="https://dev.twitter.com/docs/api/1.1/get/friends/list">GET friends/list
| Twitter Developers</a>
     * @since Twitter4J 4.0.2
     */
    PagableResponseList<User> getFriendsList(String screenName, long cursor, int count)
throws TwitterException;

    /**
     * Returns a cursored collection of user objects for every user the specified user is
following (otherwise known as their "friends").<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the next_cursor value in subsequent requests. See <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
```

Version: c8045b6727cd59de56976fc41fc93dac74df1576

Parents:

bf8653a741b94d23fce53db77820349a2a4718cc

f3e26f15293572aa652842bc48f1c8bfe4a61df1

Merge base:

0a838858951176e76090b84485e21992bf2bb082

twitter4j/twitter4j-core/src/internal-json/java/twitter4j/UserJSONImpl.java

Chunk 4:(version 2/annotation, commentary, method declaration)

```
    }

<<<<<<< HEAD
=======
    @Override
    public boolean isDefaultProfileImage() {
        return isDefaultProfileImage;
    }

    /**
     * {@inheritDoc}
     */
>>>>>>> f3e26f15293572aa652842bc48f1c8bfe4a61df1
    @Override
```

```
    }

    @Override
    public boolean isDefaultProfileImage() {
        return isDefaultProfileImage;
    }

    /**
     * {@inheritDoc}
     */
    @Override
```

Chunk 5: (Version 2/annotation, commentary, method declaration)

```
    }

<<<<<<< HEAD
=======
    @Override
    public boolean isDefaultProfile() {
        return isDefaultProfile;
    }

    /**
     * {@inheritDoc}
     */
>>>>>>> f3e26f15293572aa652842bc48f1c8bfe4a61df1
    @Override
```

```
    }
```

```
    @Override
    public boolean isDefaultProfile() {
        return isDefaultProfile;
    }

    /**
     * {@inheritDoc}
     */
    @Override
```

Version: f356ed99f0858a7be207036200876ae9edc97b6f

Parents:

8d3dd70552e9c8afb13533ecbfbc1eec927019a5

d3b8eaf58220bc3976fdd98881f53ba773a8159e

Merge base:

6b148528110cab823a73f4ae60f90cae89fd2fb5

twitter4j/twitter4j-core/src/main/java/twitter4j/api/ListsResources.java

Chunk 6: (Combination/ commentary, method interface)

```
    /**
<<<<<<< HEAD
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy.json
     *
     * @param listId The id of the list.
     * @param screenName The screen name of the member you wish to remove from the list.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *  @see  <a  href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy">POST
lists/members/destroy | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList destroyUserListMember(int listId, String screenName) throws TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
     * @param listId The id of the list.
     * @param screenNames The screen names of the members you wish to remove from the list.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList    destroyUserListMembers(int    listId,    String[]    screenNames)    throws
TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
     * @param listId The id of the list.
     * @param userIds The array of ids of the user to add as member of the list. up to 100
are allowed in a single request.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
```

```
     */
    UserList destroyUserListMembers(int listId, long[] userIds) throws TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
     * @param ownerScreenName The screen name of the user who owns the list being requested
by a slug.
     * @param slug            slug of the list
     * @param screenNames The screen names of the members you wish to remove from the list.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList   destroyUserListMembers(String   ownerScreenName,   String   slug,   String[]
screenNames) throws TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
     * @param ownerScreenName The screen name of the user who owns the list being requested
by a slug.
     * @param slug            slug of the list
     * @param userIds The array of ids of the user to add as member of the list. up to 100
are allowed in a single request.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList  destroyUserListMembers(String  ownerScreenName,  String  slug,  long[]  userIds)
throws TwitterException;

    /**
     * @deprecated use {@link #destroyUserList(int)} instead
=======
     * @deprecated use {@link #destroyUserList(long)} instead
>>>>>>> d3b8eaf58220bc3976fdd98881f53ba773a8159e
     */
```

```
    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy.json
     *
     * @param listId The id of the list.
     * @param screenName The screen name of the member you wish to remove from the list.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *  @see  <a  href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy">POST
lists/members/destroy | Twitter Developers</a>
     * @since Twitter4J 3.0.6
```

```
     */
    UserList destroyUserListMember(long listId, String screenName) throws TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
     * @param listId The id of the list.
     * @param screenNames The screen names of the members you wish to remove from the list.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList    destroyUserListMembers(long    listId,    String[]    screenNames)    throws
TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
     * @param listId The id of the list.
     * @param userIds The array of ids of the user to add as member of the list. up to 100
are allowed in a single request.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList destroyUserListMembers(long listId, long[] userIds) throws TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
     * @param ownerScreenName The screen name of the user who owns the list being requested
by a slug.
     * @param slug            slug of the list
     * @param screenNames The screen names of the members you wish to remove from the list.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList   destroyUserListMembers(String   ownerScreenName,   String   slug,   String[]
screenNames) throws TwitterException;

    /**
     * Removes the specified members from the list. The authenticated user must be the
list's owner to remove members from the list.
     * <br>This method calls https://api.twitter.com/1.1/lists/members/destroy_all.json
     *
```

```
     * @param ownerScreenName The screen name of the user who owns the list being requested
by a slug.
     * @param slug            slug of the list
     * @param userIds The array of ids of the user to add as member of the list. up to 100
are allowed in a single request.
     * @return the updated list
     * @throws TwitterException when Twitter service or network is unavailable
     *                                          @see                                        <a
href="https://dev.twitter.com/docs/api/1.1/post/lists/members/destroy_all">POST
lists/members/destroy_all | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
    UserList destroyUserListMembers(String ownerScreenName, String slug, long[] userIds)
throws TwitterException;

    /**
     * @deprecated use {@link #destroyUserList(long)} instead
     */
```

Version: 3a38698babb45aa77487616cdf666a6c10016f65

Parents:

e7101b00ef80ac27db9a65c5c1ddcfb72ab9b0e1

940c21996b66a12d106494b2832e1af424cf0763

Merge base:

6b148528110cab823a73f4ae60f90cae89fd2fb5

twitter4j/twitter4j-core/src/main/java/twitter4j/Status.java

Chunk 7: (concatenation/ commentary, method interface)

```
    String getIsoLanguageCode();
<<<<<<< HEAD

    /**
     * Returns the lang of the status text if available.
     *
     * @return two-letter iso language code
     */
    String getLang();
=======

    /**
     * Returns the targeting scopes applied to a status.
     *
     * @return the targeting scopes applied to a status.
     */
    Scopes getScopes();
>>>>>>> 940c21996b66a12d106494b2832e1af424cf0763
}
```

```
    String getIsoLanguageCode();

    /**
     * Returns the lang of the status text if available.
     *
     * @return two-letter iso language code
     * @since Twitter4J 3.0.6
     */
    String getLang();

    /**
     * Returns the targeting scopes applied to a status.
     *
     * @return the targeting scopes applied to a status.
     * @since Twitter4J 3.0.6
     */
    Scopes getScopes();
}
```

twitter4j/twitter4j-core/src/main/java/twitter4j/internal/json/StatusJSONImpl.java

Chunk 8:(concatenation/if statement)

```
            }
<<<<<<< HEAD
            if (!json.isNull("lang")) {
```

```
                lang = getUnescapedString("lang", json);
=======

            if (!json.isNull("scopes")) {
               JSONObject scopesJson = json.getJSONObject("scopes");
                if (!scopesJson.isNull("place_ids")) {
                        JSONArray placeIdsArray = scopesJson.getJSONArray("place_ids");
                        int len = placeIdsArray.length();
                        String[] placeIds = new String[len];
                        for (int i = 0; i < len; i++) {
                                placeIds[i] = placeIdsArray.getString(i);
                        }
                        scopes = new ScopesImpl(placeIds);
                }
>>>>>>> 940c21996b66a12d106494b2832e1af424cf0763
            }
```

```
            }
            if (!json.isNull("lang")) {
                lang = getUnescapedString("lang", json);
            }

            if (!json.isNull("scopes")) {
               JSONObject scopesJson = json.getJSONObject("scopes");
                if (!scopesJson.isNull("place_ids")) {
                        JSONArray placeIdsArray = scopesJson.getJSONArray("place_ids");
                        int len = placeIdsArray.length();
                        String[] placeIds = new String[len];
                        for (int i = 0; i < len; i++) {
                                placeIds[i] = placeIdsArray.getString(i);
                        }
                        scopes = new ScopesImpl(placeIds);
                }
            }
```

Version: e7101b00ef80ac27db9a65c5c1ddcfb72ab9b0e1

Parents:

04d29293b8acf912ceb3151e475274cfdb05426d

fbaff6c5fa7d5eddaa15af2c3389a7610e02083a

Merge base:

6b148528110cab823a73f4ae60f90cae89fd2fb5

twitter4j/twitter4j-core/src/main/java/twitter4j/api/FriendsFollowersResources.java

Chunk 9: (combination/commentary)

```
    PagableResponseList<User> getFollowersList(String screenName, long cursor) throws
TwitterException;
<<<<<<< HEAD

    /**
     * Returns a cursored collection of user objects for users following the specified
user.<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the next_cursor value in subsequent requests. See <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls http://api.twitter.com/1.1/friends/list.json
     *
     * @param userId The ID of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @return list of followers
     * @throws TwitterException when Twitter service or network is unavailable
     *     @see     <a     href="https://dev.twitter.com/docs/api/1.1/get/followers/list">GET
followers/list | Twitter Developers</a>
     * @since Twitter4J 3.0.6
=======

    /**
     * Returns a cursored collection of user objects for users following the specified
user.<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of {count} users (default 20) and multiple "pages" of results can be
navigated through using the next_cursor value in subsequent requests. See <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls http://api.twitter.com/1.1/friends/list.json
     *
     * @param userId The ID of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than {count}
users (default 20) at a time.
     * @param count No of records to fetch at a time with a cap of 200.
     * @return list of followers
     * @throws TwitterException when Twitter service or network is unavailable
```

```
     *     @see     <a     href="https://dev.twitter.com/docs/api/1.1/get/followers/list">GET
followers/list | Twitter Developers</a>
     * @since Twitter4J 3.0.2
>>>>>>> fbaff6c5fa7d5eddaa15af2c3389a7610e02083a
     */
```

```
    PagableResponseList<User>  getFollowersList(String  screenName,  long  cursor)  throws
TwitterException;

    /**
     * Returns a cursored collection of user objects for users following the specified
user.<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the      next_cursor      value      in      subsequent      requests.      See      <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls https://api.twitter.com/1.1/friends/list.json
     *
     * @param userId The ID of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @return list of followers
     * @throws TwitterException when Twitter service or network is unavailable
     *     @see     <a     href="https://dev.twitter.com/docs/api/1.1/get/followers/list">GET
followers/list | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
```

Chunk 10: (version 1/commentary)

```
     * Returns a cursored collection of user objects for users following the specified
user.<br>
<<<<<<< HEAD
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the      next_cursor      value      in      subsequent      requests.      See      <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls http://api.twitter.com/1.1/friends/list.json
     *
     * @param screenName The screen name of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @return list of followers
     * @throws TwitterException when Twitter service or network is unavailable
     *     @see     <a     href="https://dev.twitter.com/docs/api/1.1/get/followers/list">GET
followers/list | Twitter Developers</a>
     * @since Twitter4J 3.0.6
=======
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of {count} users (default 20) users and multiple "pages" of results can be
navigated  through  using  the  next_cursor  value  in  subsequent  requests.  See  <a
```

```
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls http://api.twitter.com/1.1/friends/list.json
     *
     * @param screenName The screen name of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than {count}
users (default 20) at a time.
     * @param count No of records to fetch at a time with a cap of 200.
     * @return list of followers
     * @throws TwitterException when Twitter service or network is unavailable
     *     @see     <a     href="https://dev.twitter.com/docs/api/1.1/get/followers/list">GET
followers/list | Twitter Developers</a>
     * @since Twitter4J 3.0.2
>>>>>>> fbaff6c5fa7d5eddaa15af2c3389a7610e02083a
     */
```

```
     * Returns a cursored collection of user objects for users following the specified
user.<br>
     * At this time, results are ordered with the most recent following first â€” however,
this ordering is subject to unannounced change and eventual consistency issues. Results are
given in groups of 20 users and multiple "pages" of results can be navigated through using
the      next_cursor      value      in      subsequent      requests.      See      <a
href="https://dev.twitter.com/docs/misc/cursoring">Using cursors to navigate collections</a>
for more information.
     * <br>This method calls https://api.twitter.com/1.1/friends/list.json
     *
     * @param screenName The screen name of the user for whom to return results for.
     * @param cursor Causes the results to be broken into pages of no more than 20 records
at a time.
     * @param count The number of users to return per page, up to a maximum of 200. Defaults
to 20.
     * @return list of followers
     * @throws TwitterException when Twitter service or network is unavailable
     *     @see     <a     href="https://dev.twitter.com/docs/api/1.1/get/followers/list">GET
followers/list | Twitter Developers</a>
     * @since Twitter4J 3.0.6
     */
```

Version: 56c311cdc3bd362df9a497bb32678e75a399e500

Parents:

a778abf3e9eebee855af459187c63a69e7148602

899dd55a0ba177818e9b1cc4d9d029d1d61cd088

Merge base:

6ae2d94986e170305f749be59b6e88e0f68e669f

twitter4j/twitter4j-core/src/test/java/twitter4j/internal/json/StatusJSONImplTest.java

Case 11: (version 2/method declaration)

```
    }
<<<<<<< HEAD
=======

    public void testLoadingGeoLocationWithCoordinatesField() throws Exception{

        //given
        String                                                                  json
="{\"filter_level\":\"medium\",\"contributors\":null,\"text\":\"@Livvy_Scott1 because I am a
clever
boy.\",\"geo\":{\"type\":\"Point\",\"coordinates\":[52.25604116,0.70928444]},\"retweeted\":f
alse,\"in_reply_to_screen_name\":\"Livvy_Scott1\",\"truncated\":false,\"lang\":\"en\",\"enti
ties\":{\"urls\":[],\"hashtags\":[],\"user_mentions\":[{\"id\":476669159,\"name\":\"livs
?\",\"indices\":[0,13],\"screen_name\":\"Livvy_Scott1\",\"id_str\":\"476669159\"}]},\"in_rep
ly_to_status_id_str\":\"320934680662794241\",\"id\":320936613498744832,\"source\":\"<a
href=\\\"http://twitter.com/download/android\\\"      rel=\\\"nofollow\\\">Twitter      for
Android<\\/a>\",\"in_reply_to_user_id_str\":\"476669159\",\"favorited\":false,\"in_reply_to_
status_id\":320934680662794241,\"retweet_count\":0,\"created_at\":\"Sun   Apr   07   16:30:26
+0000
2013\",\"in_reply_to_user_id\":476669159,\"favorite_count\":0,\"id_str\":\"32093661349874483
2\",\"place\":{\"id\":\"55c6bcd3013a0607\",\"bounding_box\":{\"type\":\"Polygon\",\"coordina
tes\":[[[0.38178,52.055592],[0.38178,52.400796],[0.967452,52.400796],[0.967452,52.055592]]]}
,\"place_type\":\"city\",\"contained_within\":[],\"name\":\"St.
Edmundsbury\",\"attributes\":{},\"country_code\":\"GB\",\"url\":\"http://api.twitter.com/1/g
eo/id/55c6bcd3013a0607.json\",\"polylines\":[],\"geometry\":null,\"country\":\"United
Kingdom\",\"full_name\":\"St.                                                    Edmundsbury,
Suffolk\"},\"user\":{\"location\":\"Attleborough\",\"default_profile\":false,\"profile_backg
round_tile\":true,\"statuses_count\":2520,\"lang\":\"en\",\"profile_link_color\":\"009999\",
\"profile_banner_url\":\"https://si0.twimg.com/profile_banners/448404395/1365018601\",\"id\"
:448404395,\"following\":null,\"protected\":false,\"favourites_count\":203,\"profile_text_co
lor\":\"333333\",\"description\":\"Born  at  a  very  young  age.  Top  Jock  -  DJ  Bookings-
maxwellDJ18@hotmail.co.uk
\",\"verified\":false,\"contributors_enabled\":false,\"profile_sidebar_border_color\":\"EEEE
EE\",\"name\":\"DJ  MaxwellJ  \",\"profile_background_color\":\"131516\",\"created_at\":\"Tue
Dec                          27                          23:49:48                          +0000
2011\",\"default_profile_image\":false,\"followers_count\":309,\"profile_image_url_https\":\
"https://si0.twimg.com/profile_images/3472432483/07133836faedec0252f17d691cb7eb5d_normal.jpe
g\",\"geo_enabled\":true,\"profile_background_image_url\":\"http://a0.twimg.com/images/theme
s/theme14/bg.gif\",\"profile_background_image_url_https\":\"https://si0.twimg.com/images/the
mes/theme14/bg.gif\",\"follow_request_sent\":null,\"url\":null,\"utc_offset\":0,\"time_zone\
":\"Casablanca\",\"notifications\":null,\"profile_use_background_image\":true,\"friends_coun
t\":342,\"profile_sidebar_fill_color\":\"EFEFEF\",\"screen_name\":\"maxwellDJ18\",\"id_str\"
:\"448404395\",\"profile_image_url\":\"http://a0.twimg.com/profile_images/3472432483/0713383
6faedec0252f17d691cb7eb5d_normal.jpeg\",\"listed_count\":0,\"is_translator\":false},\"coordi
nates\":{\"type\":\"Point\",\"coordinates\":[0.70928444,52.25604116]}}\n";
```

```
        //when
        Status status = DataObjectFactory.createStatus(json);
        GeoLocation geoLocation = status.getGeoLocation();
        //then
        assertNotNull(geoLocation);
        assertEquals(geoLocation.getLatitude(),0.70928444,0.00000001);
        assertEquals(geoLocation.getLongitude(),52.25604116,0.00000001);
    }


    public void testLangWithLangFieldDefined() throws Exception{
        //given
        String rawJson = "{ \"created_at\":\"Mon Mar 11 19:37:00 +0000 2013\",
\"id\":311199093852618752, \"id_str\":\"311199093852618752\", \"text\":\"Introducing
application-only authentication for the Twitter REST API v1.1 https:\\/\\/t.co\\/BrLLpVyuCe
^TS\", \"source\":\"web\", \"truncated\":false, \"in_reply_to_status_id\":null,
\"in_reply_to_status_id_str\":null, \"in_reply_to_user_id\":null,
\"in_reply_to_user_id_str\":null, \"in_reply_to_screen_name\":null, \"user\":{
\"id\":6253282, \"id_str\":\"6253282\", \"name\":\"Twitter API\",
\"screen_name\":\"twitterapi\", \"location\":\"San Francisco, CA\", \"description\":\"The
Real Twitter API. I tweet about API changes, service issues and happily answer questions
about Twitter and our API. Don't get an answer? It's on my website.\",
\"url\":\"http:\\/\\/dev.twitter.com\", \"entities\":{ \"url\":{ \"urls\":[ {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{ \"urls\":[ ] } }, \"protected\":false, \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\", \"geo_enabled\":true, \"verified\":true, \"statuses_count\":3392,
\"lang\":\"en\", \"contributors_enabled\":true, \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\", \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
", \"profile_link_color\":\"0084B4\", \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\", \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true, \"default_profile\":false,
\"default_profile_image\":false, \"following\":null, \"follow_request_sent\":false,
\"notifications\":null }, \"geo\":null, \"coordinates\":null, \"place\":null,
\"contributors\":[ 819797 ], \"retweet_count\":131, \"entities\":{ \"hashtags\":[ ],
\"urls\":[ { \"url\":\"https:\\/\\/t.co\\/BrLLpVyuCe\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/docs\\/auth\\/application-only-auth\",
\"display_url\":\"dev.twitter.com\\/docs\\/auth\\/applâ€¦\", \"indices\":[ 74, 97 ] } ],
\"user_mentions\":[ ] }, \"favorited\":false, \"retweeted\":true,
\"possibly_sensitive\":false, \"lang\":\"en\" }";

        //when
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        //then
        assertNotNull(status.getLang());
        assertEquals("en", status.getLang());
    }
>>>>>>> 899dd55a0ba177818e9b1cc4d9d029d1d61cd088
}
```

```
    }

    public void testLoadingGeoLocationWithCoordinatesField() throws Exception {

        //given
        String                                    json                                    =
"{\"filter_level\":\"medium\",\"contributors\":null,\"text\":\"@Livvy_Scott1 because I am a
clever
boy.\",\"geo\":{\"type\":\"Point\",\"coordinates\":[52.25604116,0.70928444]},\"retweeted\":f
alse,\"in_reply_to_screen_name\":\"Livvy_Scott1\",\"truncated\":false,\"lang\":\"en\",\"enti
ties\":{\"urls\":[],\"hashtags\":[],\"user_mentions\":[{\"id\":476669159,\"name\":\"livs
?\",\"indices\":[0,13],\"screen_name\":\"Livvy_Scott1\",\"id_str\":\"476669159\"}]},\"in_rep
ly_to_status_id_str\":\"320934680662794241\",\"id\":320936613498744832,\"source\":\"<a
href=\\\"http://twitter.com/download/android\\\"      rel=\\\"nofollow\\\">Twitter      for
Android<\\/a>\",\"in_reply_to_user_id_str\":\"476669159\",\"favorited\":false,\"in_reply_to_
status_id\":320934680662794241,\"retweet_count\":0,\"created_at\":\"Sun   Apr   07   16:30:26
+0000
2013\",\"in_reply_to_user_id\":476669159,\"favorite_count\":0,\"id_str\":\"32093661349874483
2\",\"place\":{\"id\":\"55c6bcd3013a0607\",\"bounding_box\":{\"type\":\"Polygon\",\"coordina
tes\":[[[0.38178,52.055592],[0.38178,52.400796],[0.967452,52.400796],[0.967452,52.055592]]]}
,\"place_type\":\"city\",\"contained_within\":[],\"name\":\"St.
Edmundsbury\",\"attributes\":{},\"country_code\":\"GB\",\"url\":\"http://api.twitter.com/1/g
eo/id/55c6bcd3013a0607.json\",\"polylines\":[],\"geometry\":null,\"country\":\"United
Kingdom\",\"full_name\":\"St.                                                    Edmundsbury,
Suffolk\"},\"user\":{\"location\":\"Attleborough\",\"default_profile\":false,\"profile_backg
round_tile\":true,\"statuses_count\":2520,\"lang\":\"en\",\"profile_link_color\":\"009999\",
\"profile_banner_url\":\"https://si0.twimg.com/profile_banners/448404395/1365018601\",\"id\"
:448404395,\"following\":null,\"protected\":false,\"favourites_count\":203,\"profile_text_co
lor\":\"333333\",\"description\":\"Born at a very young age. Top Jock - DJ Bookings-
maxwellDJ18@hotmail.co.uk
\",\"verified\":false,\"contributors_enabled\":false,\"profile_sidebar_border_color\":\"EEEE
EE\",\"name\":\"DJ MaxwellJ \",\"profile_background_color\":\"131516\",\"created_at\":\"Tue
Dec                        27                                23:49:48                        +0000
2011\",\"default_profile_image\":false,\"followers_count\":309,\"profile_image_url_https\":\
"https://si0.twimg.com/profile_images/3472432483/07133836faedec0252f17d691cb7eb5d_normal.jpe
g\",\"geo_enabled\":true,\"profile_background_image_url\":\"http://a0.twimg.com/images/theme
s/theme14/bg.gif\",\"profile_background_image_url_https\":\"https://si0.twimg.com/images/the
mes/theme14/bg.gif\",\"follow_request_sent\":null,\"url\":null,\"utc_offset\":0,\"time_zone\
":\"Casablanca\",\"notifications\":null,\"profile_use_background_image\":true,\"friends_coun
t\":342,\"profile_sidebar_fill_color\":\"EFEFEF\",\"screen_name\":\"maxwellDJ18\",\"id_str\"
:\"448404395\",\"profile_image_url\":\"http://a0.twimg.com/profile_images/3472432483/0713383
6faedec0252f17d691cb7eb5d_normal.jpeg\",\"listed_count\":0,\"is_translator\":false},\"coordi
nates\":{\"type\":\"Point\",\"coordinates\":[0.70928444,52.25604116]}}\n";

        //when
        Status status = DataObjectFactory.createStatus(json);
        GeoLocation geoLocation = status.getGeoLocation();
        //then
        assertNotNull(geoLocation);
        assertEquals(geoLocation.getLongitude(), 0.70928444, 0.00000001);
        assertEquals(geoLocation.getLatitude(), 52.25604116, 0.00000001);
    }


    public void testLangWithLangFieldDefined() throws Exception {
        //given
        String  rawJson  =  "{  \"created_at\":\"Mon  Mar  11  19:37:00  +0000  2013\",
\"id\":311199093852618752,     \"id_str\":\"311199093852618752\",     \"text\":\"Introducing
application-only authentication for the Twitter REST API v1.1 https:\\/\\/t.co\\/BrLLpVyuCe
^TS\",     \"source\":\"web\",     \"truncated\":false,     \"in_reply_to_status_id\":null,
```

```
\"in_reply_to_status_id_str\":null,                              \"in_reply_to_user_id\":null,
\"in_reply_to_user_id_str\":null,        \"in_reply_to_screen_name\":null,        \"user\":{
\"id\":6253282,          \"id_str\":\"6253282\",          \"name\":\"Twitter          API\",
\"screen_name\":\"twitterapi\",  \"location\":\"San  Francisco,  CA\",  \"description\":\"The
Real Twitter API. I tweet about API changes, service issues and happily answer questions
about  Twitter  and  our  API.  Don't  get  an  answer?  It's  on  my  website.\",
\"url\":\"http:\\/\\/dev.twitter.com\",     \"entities\":{     \"url\":{     \"urls\":[     {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{  \"urls\":[  ]  }  },  \"protected\":false,  \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\",       \"geo_enabled\":true,       \"verified\":true,       \"statuses_count\":3392,
\"lang\":\"en\",          \"contributors_enabled\":true,          \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\",              \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
",      \"profile_link_color\":\"0084B4\",      \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\",               \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true,                          \"default_profile\":false,
\"default_profile_image\":false,      \"following\":null,      \"follow_request_sent\":false,
\"notifications\":null     },     \"geo\":null,     \"coordinates\":null,     \"place\":null,
\"contributors\":[  819797  ],  \"retweet_count\":131,  \"entities\":{  \"hashtags\":[  ],
\"urls\":[                  {                   \"url\":\"https:\\/\\/t.co\\/BrLLpVyuCe\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/docs\\/auth\\/application-only-auth\",
\"display_url\":\"dev.twitter.com\\/docs\\/auth\\/applâ€¦\", \"indices\":[ 74, 97 ] } ],
\"user_mentions\":[       ]       },       \"favorited\":false,       \"retweeted\":true,
\"possibly_sensitive\":false, \"lang\":\"en\" }";

        //when
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        //then
        assertNotNull(status.getLang());
        assertEquals("en", status.getLang());
    }
}
```

Version: 5c85a42621ae65ec8564921515f3775fe3592b91

Parents:

b9c1734e6512983e58f4159352c1ebb21a534d20

0536705426269ce11008799ea6646080c71b27aa

Merge base:

a0ba0cf39491d746b704d445d68462357c0249e0

twitter4j/twitter4j-core/src/test/java/twitter4j/internal/json/StatusJSONImplTest.java

Chunk 12: (combination/method declaration)

```
    }

<<<<<<< HEAD
    public     void     testIsRetweetedWhenStatusIsRetweeted()     throws     JSONException,
TwitterException {
        String   rawJson   =   "{   \"created_at\":\"Mon   Mar   11   19:37:00   +0000   2013\",
\"id\":311199093852618752,     \"id_str\":\"311199093852618752\",     \"text\":\"Introducing
application-only authentication for the Twitter REST API v1.1 https:\\/\\/t.co\\/BrLLpVyuCe
^TS\",     \"source\":\"web\",     \"truncated\":false,     \"in_reply_to_status_id\":null,
\"in_reply_to_status_id_str\":null,                          \"in_reply_to_user_id\":null,
\"in_reply_to_user_id_str\":null,        \"in_reply_to_screen_name\":null,        \"user\":{
\"id\":6253282,          \"id_str\":\"6253282\",          \"name\":\"Twitter          API\",
\"screen_name\":\"twitterapi\",  \"location\":\"San  Francisco,  CA\",  \"description\":\"The
Real  Twitter  API.  I  tweet  about  API  changes,  service  issues  and  happily  answer  questions
about   Twitter   and   our   API.   Don't   get   an   answer?   It's   on   my   website.\",
\"url\":\"http:\\/\\/dev.twitter.com\",     \"entities\":{     \"url\":{     \"urls\":[     {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{  \"urls\":[  ]  }  },  \"protected\":false,  \"followers_count\":1533137,
\"friends_count\":33,  \"listed_count\":11369,  \"created_at\":\"Wed  May  23  06:01:13  +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\",      \"geo_enabled\":true,      \"verified\":true,      \"statuses_count\":3392,
\"lang\":\"en\",           \"contributors_enabled\":true,           \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\",               \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
",      \"profile_link_color\":\"0084B4\",      \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\",                \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true,                              \"default_profile\":false,
\"default_profile_image\":false,      \"following\":null,      \"follow_request_sent\":false,
\"notifications\":null     },     \"geo\":null,     \"coordinates\":null,     \"place\":null,
\"contributors\":[  819797  ],  \"retweet_count\":131,  \"entities\":{  \"hashtags\":[  ],
\"urls\":[                 {                 \"url\":\"https:\\/\\/t.co\\/BrLLpVyuCe\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/docs\\/auth\\/application-only-auth\",
\"display_url\":\"dev.twitter.com\\/docs\\/auth\\/applâ€¦\",  \"indices\":[ 74,  97  ]  }  ],
\"user_mentions\":[       ]       },       \"favorited\":false,       \"retweeted\":true,
\"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
```

```
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(true, status.isRetweeted());
    }

    public void testIsRetweetedWhenStatusIsNotRetweeted() throws JSONException,
TwitterException {
        String rawJson = "{ \"created_at\":\"Tue Mar 05 23:57:32 +0000 2013\",
\"id\":309090333021581313, \"id_str\":\"309090333021581313\", \"text\":\"As announced,
@anywhere has been retired per https:\\/\\/t.co\\/bWXjhurvwp The js file now logs a message
to the console and exits quietly. ^ARK\", \"source\":\"web\", \"truncated\":false,
\"in_reply_to_status_id\":null, \"in_reply_to_status_id_str\":null,
\"in_reply_to_user_id\":null, \"in_reply_to_user_id_str\":null,
\"in_reply_to_screen_name\":null, \"user\":{ \"id\":6253282, \"id_str\":\"6253282\",
\"name\":\"Twitter API\", \"screen_name\":\"twitterapi\", \"location\":\"San Francisco,
CA\", \"description\":\"The Real Twitter API. I tweet about API changes, service issues and
happily answer questions about Twitter and our API. Don't get an answer? It's on my
website.\", \"url\":\"http:\\/\\/dev.twitter.com\", \"entities\":{ \"url\":{ \"urls\":[ {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{ \"urls\":[ ] } }, \"protected\":false, \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\", \"geo_enabled\":true, \"verified\":true, \"statuses_count\":3392,
\"lang\":\"en\", \"contributors_enabled\":true, \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\", \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
", \"profile_link_color\":\"0084B4\", \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\", \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true, \"default_profile\":false,
\"default_profile_image\":false, \"following\":null, \"follow_request_sent\":false,
\"notifications\":null }, \"geo\":null, \"coordinates\":null, \"place\":null,
\"contributors\":[ 7588892 ], \"retweet_count\":74, \"entities\":{ \"hashtags\":[ ],
\"urls\":[ { \"url\":\"https:\\/\\/t.co\\/bWXjhurvwp\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/blog\\/sunsetting-anywhere\",
\"display_url\":\"dev.twitter.com\\/blog\\/sunsettinâ€¦\", \"indices\":[ 45, 68 ] } ],
\"user_mentions\":[ { \"screen_name\":\"anywhere\", \"name\":\"Anywhere\", \"id\":9576402,
\"id_str\":\"9576402\", \"indices\":[ 14, 23 ] } ] }, \"favorited\":false,
\"retweeted\":false, \"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(false, status.isRetweeted());
    }

    public void testIsRetweetedWhenStatusHasNoRetweetedProperty() throws JSONException,
TwitterException {
        String rawJson = "{ \"created_at\":\"Tue Mar 05 23:57:32 +0000 2013\",
\"id\":309090333021581313, \"id_str\":\"309090333021581313\", \"text\":\"As announced,
@anywhere has been retired per https:\\/\\/t.co\\/bWXjhurvwp The js file now logs a message
to the console and exits quietly. ^ARK\", \"source\":\"web\", \"truncated\":false,
\"in_reply_to_status_id\":null, \"in_reply_to_status_id_str\":null,
\"in_reply_to_user_id\":null, \"in_reply_to_user_id_str\":null,
\"in_reply_to_screen_name\":null, \"user\":{ \"id\":6253282, \"id_str\":\"6253282\",
\"name\":\"Twitter API\", \"screen_name\":\"twitterapi\", \"location\":\"San Francisco,
```

```
CA\", \"description\":\"The Real Twitter API. I tweet about API changes, service issues and
happily answer questions about Twitter and our API. Don't get an answer? It's on my
website.\", \"url\":\"http:\\/\\/dev.twitter.com\", \"entities\":{ \"url\":{ \"urls\":[ {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{ \"urls\":[ ] } }, \"protected\":false, \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\", \"geo_enabled\":true, \"verified\":true, \"statuses_count\":3392,
\"lang\":\"en\", \"contributors_enabled\":true, \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\", \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
", \"profile_link_color\":\"0084B4\", \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\", \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true, \"default_profile\":false,
\"default_profile_image\":false, \"following\":null, \"follow_request_sent\":false,
\"notifications\":null }, \"geo\":null, \"coordinates\":null, \"place\":null,
\"contributors\":[ 7588892 ], \"retweet_count\":74, \"entities\":{ \"hashtags\":[ ],
\"urls\":[ { \"url\":\"https:\\/\\/t.co\\/bWXjhurvwp\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/blog\\/sunsetting-anywhere\",
\"display_url\":\"dev.twitter.com\\/blog\\/sunsettinâ€¦\", \"indices\":[ 45, 68 ] } ],
\"user_mentions\":[ { \"screen_name\":\"anywhere\", \"name\":\"Anywhere\", \"id\":9576402,
\"id_str\":\"9576402\", \"indices\":[ 14, 23 ] } ] }, \"favorited\":false,
\"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(false, status.isRetweeted());
    }

    public void testIsRetweetedWhenStatusHasNotBooleanRetweeted() throws JSONException,
TwitterException {
        String rawJson = "{ \"created_at\":\"Tue Mar 05 23:57:32 +0000 2013\",
\"id\":309090333021581313, \"id_str\":\"309090333021581313\", \"text\":\"As announced,
@anywhere has been retired per https:\\/\\/t.co\\/bWXjhurvwp The js file now logs a message
to the console and exits quietly. ^ARK\", \"source\":\"web\", \"truncated\":false,
\"in_reply_to_status_id\":null, \"in_reply_to_status_id_str\":null,
\"in_reply_to_user_id\":null, \"in_reply_to_user_id_str\":null,
\"in_reply_to_screen_name\":null, \"user\":{ \"id\":6253282, \"id_str\":\"6253282\",
\"name\":\"Twitter API\", \"screen_name\":\"twitterapi\", \"location\":\"San Francisco,
CA\", \"description\":\"The Real Twitter API. I tweet about API changes, service issues and
happily answer questions about Twitter and our API. Don't get an answer? It's on my
website.\", \"url\":\"http:\\/\\/dev.twitter.com\", \"entities\":{ \"url\":{ \"urls\":[ {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{ \"urls\":[ ] } }, \"protected\":false, \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\", \"geo_enabled\":true, \"verified\":true, \"statuses_count\":3392,
\"lang\":\"en\", \"contributors_enabled\":true, \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\", \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
```

```
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
",      \"profile_link_color\":\"0084B4\",      \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\",               \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true,                          \"default_profile\":false,
\"default_profile_image\":false,      \"following\":null,      \"follow_request_sent\":false,
\"notifications\":null     },     \"geo\":null,     \"coordinates\":null,     \"place\":null,
\"contributors\":[   7588892   ],   \"retweet_count\":74,   \"entities\":{   \"hashtags\":[   ],
\"urls\":[                  {                  \"url\":\"https:\\/\\/t.co\\/bWXjhurvwp\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/blog\\/sunsetting-anywhere\",
\"display_url\":\"dev.twitter.com\\/blog\\/sunsettinâ€¦\",   \"indices\":[   45,   68   ]   }   ],
\"user_mentions\":[  {  \"screen_name\":\"anywhere\",  \"name\":\"Anywhere\",  \"id\":9576402,
\"id_str\":\"9576402\",    \"indices\":[    14,    23    ]    }    ]    },    \"favorited\":false,
\"retweeted\":\"yes\", \"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(false, status.isRetweeted());
    }
=======
    public void testLoadingGeoLocationWithCoordinatesField() throws Exception{

        //given
        String                                                                          json
="{\"filter_level\":\"medium\",\"contributors\":null,\"text\":\"@Livvy_Scott1 because I am a
clever
boy.\",\"geo\":{\"type\":\"Point\",\"coordinates\":[52.25604116,0.70928444]},\"retweeted\":f
alse,\"in_reply_to_screen_name\":\"Livvy_Scott1\",\"truncated\":false,\"lang\":\"en\",\"enti
ties\":{\"urls\":[],\"hashtags\":[],\"user_mentions\":[{\"id\":476669159,\"name\":\"livs
?\",\"indices\":[0,13],\"screen_name\":\"Livvy_Scott1\",\"id_str\":\"476669159\"}]},\"in_rep
ly_to_status_id_str\":\"320934680662794241\",\"id\":320936613498744832,\"source\":\"<a
href=\\\"http://twitter.com/download/android\\\"      rel=\\\"nofollow\\\">Twitter      for
Android<\\/a>\",\"in_reply_to_user_id_str\":\"476669159\",\"favorited\":false,\"in_reply_to_
status_id\":320934680662794241,\"retweet_count\":0,\"created_at\":\"Sun   Apr   07   16:30:26
+0000
2013\",\"in_reply_to_user_id\":476669159,\"favorite_count\":0,\"id_str\":\"32093661349874483
2\",\"place\":{\"id\":\"55c6bcd3013a0607\",\"bounding_box\":{\"type\":\"Polygon\",\"coordina
tes\":[[[0.38178,52.055592],[0.38178,52.400796],[0.967452,52.400796],[0.967452,52.055592]]]}
,\"place_type\":\"city\",\"contained_within\":[],\"name\":\"St.
Edmundsbury\",\"attributes\":{},\"country_code\":\"GB\",\"url\":\"http://api.twitter.com/1/g
eo/id/55c6bcd3013a0607.json\",\"polylines\":[],\"geometry\":null,\"country\":\"United
Kingdom\",\"full_name\":\"St.                                                    Edmundsbury,
Suffolk\"},\"user\":{\"location\":\"Attleborough\",\"default_profile\":false,\"profile_backg
round_tile\":true,\"statuses_count\":2520,\"lang\":\"en\",\"profile_link_color\":\"009999\",
\"profile_banner_url\":\"https://si0.twimg.com/profile_banners/448404395/1365018601\",\"id\"
:448404395,\"following\":null,\"protected\":false,\"favourites_count\":203,\"profile_text_co
lor\":\"333333\",\"description\":\"Born at a very young age. Top Jock - DJ Bookings-
maxwellDJ18@hotmail.co.uk
\",\"verified\":false,\"contributors_enabled\":false,\"profile_sidebar_border_color\":\"EEEE
EE\",\"name\":\"DJ MaxwellJ \",\"profile_background_color\":\"131516\",\"created_at\":\"Tue
Dec                        27                        23:49:48                        +0000
2011\",\"default_profile_image\":false,\"followers_count\":309,\"profile_image_url_https\":\
"https://si0.twimg.com/profile_images/3472432483/07133836faedec0252f17d691cb7eb5d_normal.jpe
g\",\"geo_enabled\":true,\"profile_background_image_url\":\"http://a0.twimg.com/images/theme
s/theme14/bg.gif\",\"profile_background_image_url_https\":\"https://si0.twimg.com/images/the
mes/theme14/bg.gif\",\"follow_request_sent\":null,\"url\":null,\"utc_offset\":0,\"time_zone\
":\"Casablanca\",\"notifications\":null,\"profile_use_background_image\":true,\"friends_coun
t\":342,\"profile_sidebar_fill_color\":\"EFEFEF\",\"screen_name\":\"maxwellDJ18\",\"id_str\"
:\"448404395\",\"profile_image_url\":\"http://a0.twimg.com/profile_images/3472432483/0713383
```

```
6faedec0252f17d691cb7eb5d_normal.jpeg\",\"listed_count\":0,\"is_translator\":false},\"coordi
nates\":{\"type\":\"Point\",\"coordinates\":[0.70928444,52.25604116]}}\n";

        //when
        Status status = DataObjectFactory.createStatus(json);
        GeoLocation geoLocation = status.getGeoLocation();
        //then
        assertNotNull(geoLocation);
        assertEquals(geoLocation.getLatitude(),0.70928444,0.00000001);
        assertEquals(geoLocation.getLongitude(),52.25604116,0.00000001);
    }

>>>>>>> 0536705426269ce11008799ea6646080c71b27aa
}
```

```
    }

    public     void     testIsRetweetedWhenStatusIsRetweeted()     throws     JSONException,
TwitterException {
        String   rawJson   =   "{   \"created_at\":\"Mon   Mar   11   19:37:00   +0000   2013\",
\"id\":311199093852618752,     \"id_str\":\"311199093852618752\",     \"text\":\"Introducing
application-only authentication for the Twitter REST API v1.1 https:\\/\\/t.co\\/BrLLpVyuCe
^TS\",     \"source\":\"web\",     \"truncated\":false,     \"in_reply_to_status_id\":null,
\"in_reply_to_status_id_str\":null,                              \"in_reply_to_user_id\":null,
\"in_reply_to_user_id_str\":null,        \"in_reply_to_screen_name\":null,        \"user\":{
\"id\":6253282,          \"id_str\":\"6253282\",          \"name\":\"Twitter          API\",
\"screen_name\":\"twitterapi\",  \"location\":\"San  Francisco,  CA\",  \"description\":\"The
Real Twitter API. I tweet about API changes, service issues and happily answer questions
about   Twitter   and   our   API.   Don't   get   an   answer?   It's   on   my   website.\",
\"url\":\"http:\\/\\/dev.twitter.com\",     \"entities\":{     \"url\":{     \"urls\":[     {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{  \"urls\":[  ]  }  },  \"protected\":false,  \"followers_count\":1533137,
\"friends_count\":33,  \"listed_count\":11369,  \"created_at\":\"Wed  May  23  06:01:13  +0000
2007\",  \"favourites_count\":25,  \"utc_offset\":-28800,  \"time_zone\":\"Pacific  Time  (US  &
Canada)\",       \"geo_enabled\":true,       \"verified\":true,       \"statuses_count\":3392,
\"lang\":\"en\",           \"contributors_enabled\":true,           \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\",               \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
",      \"profile_link_color\":\"0084B4\",      \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\",               \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true,                           \"default_profile\":false,
\"default_profile_image\":false,     \"following\":null,     \"follow_request_sent\":false,
\"notifications\":null    },     \"geo\":null,     \"coordinates\":null,     \"place\":null,
\"contributors\":[  819797  ],  \"retweet_count\":131,  \"entities\":{  \"hashtags\":[  ],
\"urls\":[                  {                  \"url\":\"https:\\/\\/t.co\\/BrLLpVyuCe\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/docs\\/auth\\/application-only-auth\",
\"display_url\":\"dev.twitter.com\\/docs\\/auth\\/applâ€¦\",  \"indices\":[ 74,  97  ]  } ],
\"user_mentions\":[       ]       },       \"favorited\":false,       \"retweeted\":true,
\"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(true, status.isRetweeted());
```

```
    }

    public void testIsRetweetedWhenStatusIsNotRetweeted() throws JSONException,
TwitterException {
        String rawJson = "{ \"created_at\":\"Tue Mar 05 23:57:32 +0000 2013\",
\"id\":309090333021581313, \"id_str\":\"309090333021581313\", \"text\":\"As announced,
@anywhere has been retired per https:\\/\\/t.co\\/bWXjhurvwp The js file now logs a message
to the console and exits quietly. ^ARK\", \"source\":\"web\", \"truncated\":false,
\"in_reply_to_status_id\":null, \"in_reply_to_status_id_str\":null,
\"in_reply_to_user_id\":null, \"in_reply_to_user_id_str\":null,
\"in_reply_to_screen_name\":null, \"user\":{ \"id\":6253282, \"id_str\":\"6253282\",
\"name\":\"Twitter API\", \"screen_name\":\"twitterapi\", \"location\":\"San Francisco,
CA\", \"description\":\"The Real Twitter API. I tweet about API changes, service issues and
happily answer questions about Twitter and our API. Don't get an answer? It's on my
website.\", \"url\":\"http:\\/\\/dev.twitter.com\", \"entities\":{ \"url\":{ \"urls\":[ {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{ \"urls\":[ ] } }, \"protected\":false, \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\", \"geo_enabled\":true, \"verified\":true, \"statuses_count\":3392,
\"lang\":\"en\", \"contributors_enabled\":true, \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\", \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
", \"profile_link_color\":\"0084B4\", \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\", \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true, \"default_profile\":false,
\"default_profile_image\":false, \"following\":null, \"follow_request_sent\":false,
\"notifications\":null }, \"geo\":null, \"coordinates\":null, \"place\":null,
\"contributors\":[ 7588892 ], \"retweet_count\":74, \"entities\":{ \"hashtags\":[ ],
\"urls\":[ { \"url\":\"https:\\/\\/t.co\\/bWXjhurvwp\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/blog\\/sunsetting-anywhere\",
\"display_url\":\"dev.twitter.com\\/blog\\/sunsettinâ€¦\", \"indices\":[ 45, 68 ] } ],
\"user_mentions\":[ { \"screen_name\":\"anywhere\", \"name\":\"Anywhere\", \"id\":9576402,
\"id_str\":\"9576402\", \"indices\":[ 14, 23 ] } ] }, \"favorited\":false,
\"retweeted\":false, \"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(false, status.isRetweeted());
    }

    public void testIsRetweetedWhenStatusHasNoRetweetedProperty() throws JSONException,
TwitterException {
        String rawJson = "{ \"created_at\":\"Tue Mar 05 23:57:32 +0000 2013\",
\"id\":309090333021581313, \"id_str\":\"309090333021581313\", \"text\":\"As announced,
@anywhere has been retired per https:\\/\\/t.co\\/bWXjhurvwp The js file now logs a message
to the console and exits quietly. ^ARK\", \"source\":\"web\", \"truncated\":false,
\"in_reply_to_status_id\":null, \"in_reply_to_status_id_str\":null,
\"in_reply_to_user_id\":null, \"in_reply_to_user_id_str\":null,
\"in_reply_to_screen_name\":null, \"user\":{ \"id\":6253282, \"id_str\":\"6253282\",
\"name\":\"Twitter API\", \"screen_name\":\"twitterapi\", \"location\":\"San Francisco,
CA\", \"description\":\"The Real Twitter API. I tweet about API changes, service issues and
happily answer questions about Twitter and our API. Don't get an answer? It's on my
website.\", \"url\":\"http:\\/\\/dev.twitter.com\", \"entities\":{ \"url\":{ \"urls\":[ {
```

```
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{  \"urls\":[  ]  }  },  \"protected\":false,  \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\",      \"geo_enabled\":true,      \"verified\":true,      \"statuses_count\":3392,
\"lang\":\"en\",          \"contributors_enabled\":true,          \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\",              \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
",      \"profile_link_color\":\"0084B4\",      \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\",               \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true,                          \"default_profile\":false,
\"default_profile_image\":false,      \"following\":null,      \"follow_request_sent\":false,
\"notifications\":null     },     \"geo\":null,     \"coordinates\":null,     \"place\":null,
\"contributors\":[  7588892  ],  \"retweet_count\":74,  \"entities\":{  \"hashtags\":[  ],
\"urls\":[                  {                  \"url\":\"https:\\/\\/t.co\\/bWXjhurvwp\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/blog\\/sunsetting-anywhere\",
\"display_url\":\"dev.twitter.com\\/blog\\/sunsettinâ€¦\",  \"indices\":[  45,  68  ]  } ],
\"user_mentions\":[ {  \"screen_name\":\"anywhere\",  \"name\":\"Anywhere\",  \"id\":9576402,
\"id_str\":\"9576402\",   \"indices\":[   14,   23   ]   }   ]   },   \"favorited\":false,
\"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(false, status.isRetweeted());
    }

    public void testIsRetweetedWhenStatusHasNotBooleanRetweeted() throws JSONException,
TwitterException {
        String rawJson = "{  \"created_at\":\"Tue Mar 05 23:57:32 +0000 2013\",
\"id\":309090333021581313,   \"id_str\":\"309090333021581313\",   \"text\":\"As announced,
@anywhere has been retired per https:\\/\\/t.co\\/bWXjhurvwp The js file now logs a message
to the console and exits quietly. ^ARK\",  \"source\":\"web\",  \"truncated\":false,
\"in_reply_to_status_id\":null,                          \"in_reply_to_status_id_str\":null,
\"in_reply_to_user_id\":null,                            \"in_reply_to_user_id_str\":null,
\"in_reply_to_screen_name\":null,    \"user\":{    \"id\":6253282,    \"id_str\":\"6253282\",
\"name\":\"Twitter API\",  \"screen_name\":\"twitterapi\",  \"location\":\"San Francisco,
CA\", \"description\":\"The Real Twitter API. I tweet about API changes, service issues and
happily answer questions about Twitter and our API. Don't get an answer? It's on my
website.\",  \"url\":\"http:\\/\\/dev.twitter.com\",  \"entities\":{  \"url\":{  \"urls\":[  {
\"url\":\"http:\\/\\/dev.twitter.com\", \"expanded_url\":null, \"indices\":[ 0, 22 ] } ] },
\"description\":{  \"urls\":[  ]  }  },  \"protected\":false,  \"followers_count\":1533137,
\"friends_count\":33, \"listed_count\":11369, \"created_at\":\"Wed May 23 06:01:13 +0000
2007\", \"favourites_count\":25, \"utc_offset\":-28800, \"time_zone\":\"Pacific Time (US &
Canada)\",      \"geo_enabled\":true,      \"verified\":true,      \"statuses_count\":3392,
\"lang\":\"en\",          \"contributors_enabled\":true,          \"is_translator\":false,
\"profile_background_color\":\"C0DEED\",
\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/656
927849\\/miyt9dpjz77sc0w3d4vj.png\",
\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_imag
es\\/656927849\\/miyt9dpjz77sc0w3d4vj.png\",              \"profile_background_tile\":true,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2284174872\\/7df3h38zabcv
jylnyfe3_normal.png\",
\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2284174872\\/7df3
h38zabcvjylnyfe3_normal.png\",
```

```
\"profile_banner_url\":\"https:\\/\\/si0.twimg.com\\/profile_banners\\/6253282\\/1347394302\
",      \"profile_link_color\":\"0084B4\",      \"profile_sidebar_border_color\":\"C0DEED\",
\"profile_sidebar_fill_color\":\"DDEEF6\",               \"profile_text_color\":\"333333\",
\"profile_use_background_image\":true,                          \"default_profile\":false,
\"default_profile_image\":false,      \"following\":null,      \"follow_request_sent\":false,
\"notifications\":null     },     \"geo\":null,     \"coordinates\":null,     \"place\":null,
\"contributors\":[  7588892  ],  \"retweet_count\":74,  \"entities\":{  \"hashtags\":[  ],
\"urls\":[                  {                  \"url\":\"https:\\/\\/t.co\\/bWXjhurvwp\",
\"expanded_url\":\"https:\\/\\/dev.twitter.com\\/blog\\/sunsetting-anywhere\",
\"display_url\":\"dev.twitter.com\\/blog\\/sunsettinâ€¦\",  \"indices\":[  45,  68  ]  }  ],
\"user_mentions\":[  {  \"screen_name\":\"anywhere\",  \"name\":\"Anywhere\",  \"id\":9576402,
\"id_str\":\"9576402\",   \"indices\":[   14,   23   ]   }   ]   },   \"favorited\":false,
\"retweeted\":\"yes\", \"possibly_sensitive\":false, \"lang\":\"en\" }";
        JSONObject json = new JSONObject(rawJson);
        StatusJSONImpl status = new StatusJSONImpl(json);

        assertEquals(false, status.isRetweeted());
    }

    public void testLoadingGeoLocationWithCoordinatesField() throws Exception{

        //given
        String                                                                          json
="{\"filter_level\":\"medium\",\"contributors\":null,\"text\":\"@Livvy_Scott1 because I am a
clever
boy.\",\"geo\":{\"type\":\"Point\",\"coordinates\":[52.25604116,0.70928444]},\"retweeted\":f
alse,\"in_reply_to_screen_name\":\"Livvy_Scott1\",\"truncated\":false,\"lang\":\"en\",\"enti
ties\":{\"urls\":[],\"hashtags\":[],\"user_mentions\":[{\"id\":476669159,\"name\":\"livs
?\",\"indices\":[0,13],\"screen_name\":\"Livvy_Scott1\",\"id_str\":\"476669159\"}]},\"in_rep
ly_to_status_id_str\":\"320934680662794241\",\"id\":320936613498744832,\"source\":\"<a
href=\\\"http://twitter.com/download/android\\\"      rel=\\\"nofollow\\\">Twitter      for
Android<\\/a>\",\"in_reply_to_user_id_str\":\"476669159\",\"favorited\":false,\"in_reply_to_
status_id\":320934680662794241,\"retweet_count\":0,\"created_at\":\"Sun   Apr   07   16:30:26
+0000
2013\",\"in_reply_to_user_id\":476669159,\"favorite_count\":0,\"id_str\":\"32093661349874483
2\",\"place\":{\"id\":\"55c6bcd3013a0607\",\"bounding_box\":{\"type\":\"Polygon\",\"coordina
tes\":[[[0.38178,52.055592],[0.38178,52.400796],[0.967452,52.400796],[0.967452,52.055592]]]}
,\"place_type\":\"city\",\"contained_within\":[],\"name\":\"St.
Edmundsbury\",\"attributes\":{},\"country_code\":\"GB\",\"url\":\"http://api.twitter.com/1/g
eo/id/55c6bcd3013a0607.json\",\"polylines\":[],\"geometry\":null,\"country\":\"United
Kingdom\",\"full_name\":\"St.                                                   Edmundsbury,
Suffolk\"},\"user\":{\"location\":\"Attleborough\",\"default_profile\":false,\"profile_backg
round_tile\":true,\"statuses_count\":2520,\"lang\":\"en\",\"profile_link_color\":\"009999\",
\"profile_banner_url\":\"https://si0.twimg.com/profile_banners/448404395/1365018601\",\"id\"
:448404395,\"following\":null,\"protected\":false,\"favourites_count\":203,\"profile_text_co
lor\":\"333333\",\"description\":\"Born at a very young age. Top Jock - DJ Bookings-
maxwellDJ18@hotmail.co.uk
\",\"verified\":false,\"contributors_enabled\":false,\"profile_sidebar_border_color\":\"EEEE
EE\",\"name\":\"DJ MaxwellJ \",\"profile_background_color\":\"131516\",\"created_at\":\"Tue
Dec                        27                        23:49:48                        +0000
2011\",\"default_profile_image\":false,\"followers_count\":309,\"profile_image_url_https\":\
"https://si0.twimg.com/profile_images/3472432483/07133836faedec0252f17d691cb7eb5d_normal.jpe
g\",\"geo_enabled\":true,\"profile_background_image_url\":\"http://a0.twimg.com/images/theme
s/theme14/bg.gif\",\"profile_background_image_url_https\":\"https://si0.twimg.com/images/the
mes/theme14/bg.gif\",\"follow_request_sent\":null,\"url\":null,\"utc_offset\":0,\"time_zone\
":\"Casablanca\",\"notifications\":null,\"profile_use_background_image\":true,\"friends_coun
t\":342,\"profile_sidebar_fill_color\":\"EFEFEF\",\"screen_name\":\"maxwellDJ18\",\"id_str\"
:\"448404395\",\"profile_image_url\":\"http://a0.twimg.com/profile_images/3472432483/0713383
6faedec0252f17d691cb7eb5d_normal.jpeg\",\"listed_count\":0,\"is_translator\":false},\"coordi
nates\":{\"type\":\"Point\",\"coordinates\":[0.70928444,52.25604116]}}\n";

        //when
```

```
        Status status = DataObjectFactory.createStatus(json);
        GeoLocation geoLocation = status.getGeoLocation();
        //then
        assertNotNull(geoLocation);
        assertEquals(geoLocation.getLatitude(),0.70928444,0.00000001);
        assertEquals(geoLocation.getLongitude(),52.25604116,0.00000001);
    }
}
```

Version: 0536705426269ce11008799ea6646080c71b27aa

Parents:

e9086df1485e39ee262afc29765c66daa9a05f9b

01b290c32a760f0439c2ef472798f4921bfe1b85

Merge base:

a0ba0cf39491d746b704d445d68462357c0249e0

twitter4j/twitter4j-core/src/test/java/twitter4j/internal/json/StatusJSONImplTest.java

Chunk 13:(version 1/Method invocation)

```
        assertEquals(geoLocation.getLongitude(),52.25604116,0.00000001);
<<<<<<< HEAD
=======
        assertEquals(geoLocation.getType(),"Point");
>>>>>>> 01b290c32a760f0439c2ef472798f4921bfe1b85
    }
```

```
        assertEquals(geoLocation.getLongitude(),52.25604116,0.00000001);
    }
```

Version: 3b3f666c707087fa713cf8249472e42ed70e23bd

Parents:

36dfbaead954e1cefde2056051ca5af74bafa06b

49005ffb26252150068991221 2b7286627de4544

Merge base:

a0ba0cf39491d746b704d445d68462357c0249e0

twitter4j/twitter4j-core/src/main/java/twitter4j/conf/ConfigurationBase.java

Chunk 14: (concatenation/method declaration)

```
    }

<<<<<<< HEAD
    public void setIncludeMyRetweetEnabled(boolean enabled) {
        this.includeMyRetweetEnabled = enabled;
    }
=======
        public void setIncludeMyRetweetEnabled(boolean enabled) {
                this.includeMyRetweetEnabled = enabled;
        }

        public boolean isTrimUserEnabled() {
                return this.trimUserEnabled;
        }

        public void setTrimUserEnabled(boolean enabled) {
                this.trimUserEnabled = enabled;
        }
>>>>>>> 49005ffb26252150068991221 2b7286627de4544

    public boolean isJSONStoreEnabled() {
```

```
    }

    public boolean isIncludeMyRetweetEnabled() {
        return this.includeMyRetweetEnabled;
    }

        public void setIncludeMyRetweetEnabled(boolean enabled) {
                this.includeMyRetweetEnabled = enabled;
        }

        public boolean isTrimUserEnabled() {
                return this.trimUserEnabled;
        }

        public void setTrimUserEnabled(boolean enabled) {
                this.trimUserEnabled = enabled;
        }

    public boolean isJSONStoreEnabled() {
```

Version: a085f5dafa9570f77147506ee0f07e6e1f88dfff

Parents:

a7f0e05e5a65a9e703f28d2f5bd26e47690ed38e

aa6e62fb6c5ca1114703e133cd2bff6c6ccd92a9

Merge base:

1a87b363305207505270fc83d08ffd8132d98dad

twitter4j/twitter4j-core/src/test/java/twitter4j/internal/json/UserJSONImplTest.java

Chunk 15: (version 2/method declaration)

```
    }
<<<<<<< HEAD

    public void testGetURLEntity1() throws JSONException, TwitterException {
        // full url entity
        String                                  rawJson                                     =
"{\"profile_sidebar_fill_color\":\"DDEEF6\",\"id\":219570539,\"favourites_count\":252,\"prof
ile_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_images\\/7
36851794\\/e6f0992a68eda21a2809efbec2f88050.jpeg\",\"screen_name\":\"gjmp10\",\"following\":
false,\"location\":\"\\u4eac\\u90fd\\u5e9c\\u4eac\\u90fd\\u5e02\\u6771\\u5c71\\u533a\\u4e09\
\u6761\\u901a\\u5357\\u4e8c\\u7b4b\\u76ee\\u767d\\u5ddd\\u7b4b\\u897f\\u5165\\u30eb\\u4e8c\\
u4e01\\u76ee\",\"contributors_enabled\":false,\"profile_background_color\":\"FF8F00\",\"time
_zone\":\"Osaka\",\"utc_offset\":32400,\"name\":\"\\u9577\\u3044\\u9577\\u3044\\u9577\\u3044
\\u9577\\u3044\\u3042\\u3042\\u3042\\u3042\\u3042\\u3042\\u304b\\u3055\\u305f\\u306a\\u306f\
\u3046\",\"notifications\":false,\"geo_enabled\":true,\"profile_image_url\":\"http:\\/\\/a0.
twimg.com\\/profile_images\\/2964940756\\/51e2ad9e42297e2be926ec4794b95267_normal.jpeg\",\"i
d_str\":\"219570539\",\"default_profile\":false,\"follow_request_sent\":false,\"profile_back
ground_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_background_images\\/736851794\\/e6f09
92a68eda21a2809efbec2f88050.jpeg\",\"protected\":false,\"profile_link_color\":\"0084B4\",\"v
erified\":false,\"entities\":{\"description\":{\"urls\":[{\"expanded_url\":\"http:\\/\\/jigt
wi.jp\\/\",\"display_url\":\"jigtwi.jp\",\"url\":\"http:\\/\\/t.co\\/n578Jr6S\",\"indices\":
[31,51]},{\"expanded_url\":\"http:\\/\\/twitpic.com\\/7tx0wk\",\"display_url\":\"twitpic.com
\\/7tx0wk\",\"url\":\"http:\\/\\/t.co\\/i2cX7pEY\",\"indices\":[52,72]},{\"expanded_url\":\"
http:\\/\\/twitter.com\\/ak_12\\/status\\/12813969077051392\",\"display_url\":\"twitter.com\
\/ak_12\\/status\\/1\\u2026\",\"url\":\"http:\\/\\/t.co\\/PlItS8jh\",\"indices\":[77,97]}]},
\"url\":{\"urls\":[{\"display_url\":\"google.co.jp\\/search?q=adgjg\",\"expanded_url\":\"htt
ps:\\/\\/www.google.co.jp\\/search?q=adgjg\",\"indices\":[0,21],\"url\":\"https:\\/\\/t.co\\
/gLcBbz6G\"}]}},\"listed_count\":5,\"profile_use_background_image\":true,\"statuses_count\":
979,\"profile_text_color\":\"333333\",\"created_at\":\"Thu    Nov    25    06:48:26    +0000
2010\",\"lang\":\"ja\",\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_ima
ges\\/2964940756\\/51e2ad9e42297e2be926ec4794b95267_normal.jpeg\",\"is_translator\":false,\"
profile_sidebar_border_color\":\"FFFFFF\",\"friends_count\":12,\"url\":\"https:\\/\\/t.co\\/
gLcBbz6G\",\"default_profile_image\":false,\"followers_count\":10,\"description\":\"\\n09012
345678\\n@kouda12345     #test     http:\\/\\/t.co\\/n578Jr6S     http:\\/\\/t.co\\/i2cX7pEY
http:\\/\\/t.co\\/PlItS8jh                                  0123456789\\n\\u3066\\u3059\\u3068
@gjmp9\",\"profile_background_tile\":true}";
        JSONObject json = new JSONObject(rawJson);
        UserJSONImpl user = new UserJSONImpl(json);

        URLEntity urlEntity = user.getURLEntity();
        assertNotNull(urlEntity);
        assertEquals("https://www.google.co.jp/search?q=adgjg", urlEntity.getExpandedURL());
        assertEquals("google.co.jp/search?q=adgjg", urlEntity.getDisplayURL());
        assertEquals("https://t.co/gLcBbz6G", urlEntity.getURL());
        assertEquals(0, urlEntity.getStart());
        assertEquals(21, urlEntity.getEnd());
    }
```

```
    public void testGetURLEntity2() throws JSONException, TwitterException {
        // url, indices only
        String                                  rawJson                                  =
"{\"profile_sidebar_fill_color\":\"F3F3F3\",\"id\":4311171,\"favourites_count\":118,\"profil
e_background_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_background_images\\/725
168446\\/fed71f230b033a13a69d803d464871ec.gif\",\"screen_name\":\"ApplePedlar\",\"following\
":false,\"location\":\"Fukui,
Japan\",\"contributors_enabled\":false,\"profile_background_color\":\"FDF4D3\",\"time_zone\"
:\"Tokyo\",\"utc_offset\":32400,\"name\":\"Naoya
Hatayama\",\"notifications\":false,\"geo_enabled\":true,\"profile_image_url\":\"http:\\/\\/a
0.twimg.com\\/profile_images\\/1197534673\\/100x100_normal.png\",\"id_str\":\"4311171\",\"de
fault_profile\":false,\"follow_request_sent\":false,\"profile_background_image_url\":\"http:
\\/\\/a0.twimg.com\\/profile_background_images\\/725168446\\/fed71f230b033a13a69d803d464871e
c.gif\",\"protected\":false,\"profile_link_color\":\"990000\",\"verified\":false,\"entities\
":{\"description\":{\"urls\":[]},\"url\":{\"urls\":[{\"display_url\":null,\"expanded_url\":n
ull,\"indices\":[0,30],\"url\":\"http:\\/\\/twitter.com\\/ApplePedlar\"}]}},\"listed_count\"
:74,\"profile_use_background_image\":true,\"statuses_count\":10130,\"profile_text_color\":\"
333333\",\"created_at\":\"Thu           Apr           12           06:27:44           +0000
2007\",\"lang\":\"ja\",\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_ima
ges\\/1197534673\\/100x100_normal.png\",\"is_translator\":false,\"profile_sidebar_border_col
or\":\"FFFFFF\",\"friends_count\":984,\"url\":\"http:\\/\\/twitter.com\\/ApplePedlar\",\"def
ault_profile_image\":false,\"followers_count\":901,\"description\":\"\\u3081\\u304c\\u306d\\
u4f1a\\u9928\\u3067\\u50cd\\u304f\\u30e2\\u30d0\\u30a4\\u30eb\\u7cfb\\u958b\\u767a\\u8005\\u
3067\\u3059\\n#kosen           #fukui           #sabae           #jigtwi           #Twitter4J
#DQ10\",\"profile_background_tile\":true}";
        JSONObject json = new JSONObject(rawJson);
        UserJSONImpl user = new UserJSONImpl(json);

        URLEntity urlEntity = user.getURLEntity();
        assertNotNull(urlEntity);
        assertEquals("http://twitter.com/ApplePedlar", urlEntity.getExpandedURL());
        assertEquals("http://twitter.com/ApplePedlar", urlEntity.getDisplayURL());
        assertEquals("http://twitter.com/ApplePedlar", urlEntity.getURL());
        assertEquals(0, urlEntity.getStart());
        assertEquals(30, urlEntity.getEnd());
    }

    public void testGetURLEntity3() throws JSONException, TwitterException {
        // url is null
        String    rawJson    =    "{\"id\":78941611,\"id_str\":\"78941611\",\"name\":\"Harsha
Bhogle\",\"screen_name\":\"bhogleharsha\",\"location\":\"Mumbai,
India\",\"description\":\"once upon a time a chem engr and mgmt grad,now cricket and
motivational                                speaking                                my
calling!\",\"url\":null,\"entities\":{\"description\":{\"urls\":[]}},\"protected\":false,\"f
ollowers_count\":540458,\"friends_count\":83,\"listed_count\":4229,\"created_at\":\"Thu  Oct
01                                  16:18:03                                +0000
2009\",\"favourites_count\":3,\"utc_offset\":19800,\"time_zone\":\"Chennai\",\"geo_enabled\"
:false,\"verified\":true,\"statuses_count\":12205,\"lang\":\"en\",\"contributors_enabled\":f
alse,\"is_translator\":false,\"profile_background_color\":\"C0DEED\",\"profile_background_im
age_url\":\"http:\\/\\/a0.twimg.com\\/images\\/themes\\/theme1\\/bg.png\",\"profile_backgrou
nd_image_url_https\":\"https:\\/\\/si0.twimg.com\\/images\\/themes\\/theme1\\/bg.png\",\"pro
file_background_tile\":false,\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_image
s\\/2769690929\\/61db65334849e8876c538051e885cdab_normal.png\",\"profile_image_url_https\":\
"https:\\/\\/si0.twimg.com\\/profile_images\\/2769690929\\/61db65334849e8876c538051e885cdab_
normal.png\",\"profile_link_color\":\"0084B4\",\"profile_sidebar_border_color\":\"C0DEED\",\
"profile_sidebar_fill_color\":\"DDEEF6\",\"profile_text_color\":\"333333\",\"profile_use_bac
kground_image\":true,\"default_profile\":true,\"default_profile_image\":false,\"following\":
null,\"follow_request_sent\":null,\"notifications\":null}";
        JSONObject json = new JSONObject(rawJson);
        UserJSONImpl user = new UserJSONImpl(json);

        assertNull(user.getURL());
```

```
        URLEntity urlEntity = user.getURLEntity();
        assertNotNull(urlEntity);
        assertEquals("", urlEntity.getExpandedURL());
        assertEquals("", urlEntity.getDisplayURL());
        assertEquals("", urlEntity.getURL());
        assertEquals(0, urlEntity.getStart());
        assertEquals(0, urlEntity.getEnd());
    }


=======

    public void testProfileImageURL() throws JSONException, TwitterException {
        String rawJsonWithoutProfileImageExtension =
"{\"id\":400609977,\"id_str\":\"400609977\",\"name\":\"Chris
Bautista\",\"screen_name\":\"ayecrispy\",\"location\":\"Jacksonville,
FL\",\"description\":\"I'm a gamer and will always be one. I like to keep up with the
entertainment life. Where it's celebrities, technology or trying to keep up with latest
trend\",\"url\":null,\"entities\":{\"description\":{\"urls\":[]}},\"protected\":false,\"foll
owers_count\":17,\"friends_count\":177,\"listed_count\":0,\"created_at\":\"Sat Oct 29
09:23:10 +0000 2011\",\"favourites_count\":0,\"utc_offset\":-18000,\"time_zone\":\"Eastern
Time (US &
Canada)\",\"geo_enabled\":false,\"verified\":false,\"statuses_count\":113,\"lang\":\"en\",\"
status\":{\"created_at\":\"Sun Dec 16 02:37:57 +0000
2012\",\"id\":280139673333035008,\"id_str\":\"280139673333035008\",\"text\":\"Gotta love
olive Garden!\",\"source\":\"\\u003ca href=\\\"http:\\/\\/tweedleapp.com\\/\\\"
rel=\\\"nofollow\\\"\\u003e
Tweedle\\u003c\\/a\\u003e\",\"truncated\":false,\"in_reply_to_status_id\":null,\"in_reply_to
_status_id_str\":null,\"in_reply_to_user_id\":null,\"in_reply_to_user_id_str\":null,\"in_rep
ly_to_screen_name\":null,\"geo\":null,\"coordinates\":null,\"place\":null,\"contributors\":n
ull,\"retweet_count\":0,\"entities\":{\"hashtags\":[],\"urls\":[],\"user_mentions\":[]},\"fa
vorited\":false,\"retweeted\":false},\"contributors_enabled\":false,\"is_translator\":false,
\"profile_background_color\":\"1A1B1F\",\"profile_background_image_url\":\"http:\\/\\/a0.twi
mg.com\\/images\\/themes\\/theme9\\/bg.gif\",\"profile_background_image_url_https\":\"https:
\\/\\/si0.twimg.com\\/images\\/themes\\/theme9\\/bg.gif\",\"profile_background_tile\":false,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/1835646533\\/gu44kEhi_nor
mal\",\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/1835646533\
\/gu44kEhi_normal\",\"profile_link_color\":\"2FC2EF\",\"profile_sidebar_border_color\":\"181
A1E\",\"profile_sidebar_fill_color\":\"252429\",\"profile_text_color\":\"666666\",\"profile_
use_background_image\":true,\"default_profile\":false,\"default_profile_image\":false,\"foll
owing\":false,\"follow_request_sent\":false,\"notifications\":false}";
        JSONObject jsonWithoutProfileImageExtension = new
JSONObject(rawJsonWithoutProfileImageExtension);
        UserJSONImpl userWithoutProfileImageExtension = new
UserJSONImpl(jsonWithoutProfileImageExtension);

        assertEquals(userWithoutProfileImageExtension.getBiggerProfileImageURL(),
"http://si0.twimg.com/profile_images/1835646533/gu44kEhi_bigger");
        assertEquals(userWithoutProfileImageExtension.getOriginalProfileImageURL(),
"http://si0.twimg.com/profile_images/1835646533/gu44kEhi");

        String rawJsonWithProfileImageExtension =
"{\"id\":613742117,\"id_str\":\"613742117\",\"name\":\"Tweedle\",\"screen_name\":\"tweedleap
p\",\"location\":\"\",\"description\":\"Twitter application for Android, follow this account
for updates.\\r\\n\\r\\nDeveloped by
@HandlerExploit\",\"url\":\"http:\\/\\/tweedleapp.com\",\"entities\":{\"url\":{\"urls\":[{\"
url\":\"http:\\/\\/tweedleapp.com\",\"expanded_url\":null,\"indices\":[0,21]}]},\"descriptio
n\":{\"urls\":[]}},\"protected\":false,\"followers_count\":2210,\"friends_count\":1,\"listed
_count\":20,\"created_at\":\"Wed Jun 20 20:42:48 +0000
2012\",\"favourites_count\":1,\"utc_offset\":null,\"time_zone\":null,\"geo_enabled\":false,\
"verified\":false,\"statuses_count\":323,\"lang\":\"en\",\"status\":{\"created_at\":\"Sun
```

```
Dec                          16                          06:47:33                          +0000
2012\",\"id\":280202487317790721,\"id_str\":\"280202487317790721\",\"text\":\"@bcw_  It  is
better in some aspects, but in others it is not ideal. We need to add a way to notify the
user                                                                                  better
though.\",\"source\":\"web\",\"truncated\":false,\"in_reply_to_status_id\":28020231408046489
6,\"in_reply_to_status_id_str\":\"280202314080464896\",\"in_reply_to_user_id\":101066646,\"i
n_reply_to_user_id_str\":\"101066646\",\"in_reply_to_screen_name\":\"bcw_\",\"geo\":null,\"c
oordinates\":null,\"place\":null,\"contributors\":null,\"retweet_count\":0,\"entities\":{\"h
ashtags\":[],\"urls\":[],\"user_mentions\":[{\"screen_name\":\"bcw_\",\"name\":\"Verified
Bradley\",\"id\":101066646,\"id_str\":\"101066646\",\"indices\":[0,5]}]},\"favorited\":false
,\"retweeted\":false},\"contributors_enabled\":false,\"is_translator\":false,\"profile_backg
round_color\":\"C0DEED\",\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/images
\\/themes\\/theme1\\/bg.png\",\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg
.com\\/images\\/themes\\/theme1\\/bg.png\",\"profile_background_tile\":false,\"profile_image
_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2433552309\\/dltv4u9hajvoxsne5bly_normal
.png\",\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2433552309
\\/dltv4u9hajvoxsne5bly_normal.png\",\"profile_link_color\":\"0084B4\",\"profile_sidebar_bor
der_color\":\"C0DEED\",\"profile_sidebar_fill_color\":\"DDEEF6\",\"profile_text_color\":\"33
3333\",\"profile_use_background_image\":true,\"default_profile\":true,\"default_profile_imag
e\":false,\"following\":true,\"follow_request_sent\":false,\"notifications\":false}";
        JSONObject              jsonWithProfileImageExtension              =              new
JSONObject(rawJsonWithProfileImageExtension);
        UserJSONImpl             userWithProfileImageExtension             =             new
UserJSONImpl(jsonWithProfileImageExtension);

        assertEquals(userWithProfileImageExtension.getBiggerProfileImageURL(),
"http://si0.twimg.com/profile_images/2433552309/dltv4u9hajvoxsne5bly_bigger.png");
        assertEquals(userWithProfileImageExtension.getOriginalProfileImageURL(),
"http://si0.twimg.com/profile_images/2433552309/dltv4u9hajvoxsne5bly.png");
    }
>>>>>>> aa6e62fb6c5ca1114703e133cd2bff6c6ccd92a9
}
```

```
    }

    public void testProfileImageURL() throws JSONException, TwitterException {
        String                      rawJsonWithoutProfileImageExtension                      =
"{\"id\":400609977,\"id_str\":\"400609977\",\"name\":\"Chris
Bautista\",\"screen_name\":\"ayecrispy\",\"location\":\"Jacksonville,
FL\",\"description\":\"I'm a gamer and will always be one. I like to keep up with the
entertainment life. Where it's celebrities, technology or trying to keep up with latest
trend\",\"url\":null,\"entities\":{\"description\":{\"urls\":[]}},\"protected\":false,\"foll
owers_count\":17,\"friends_count\":177,\"listed_count\":0,\"created_at\":\"Sat    Oct    29
09:23:10  +0000  2011\",\"favourites_count\":0,\"utc_offset\":-18000,\"time_zone\":\"Eastern
Time                                  (US                                                &
Canada)\",\"geo_enabled\":false,\"verified\":false,\"statuses_count\":113,\"lang\":\"en\",\"
status\":{\"created_at\":\"Sun          Dec          16          02:37:57          +0000
2012\",\"id\":280139673333035008,\"id_str\":\"280139673333035008\",\"text\":\"Gotta     love
olive       Garden!\",\"source\":\"\\u003ca       href=\\\"http:\\/\\/tweedleapp.com\\/\\\"
rel=\\\"nofollow\\\"\\u003e
Tweedle\\u003c\\/a\\u003e\",\"truncated\":false,\"in_reply_to_status_id\":null,\"in_reply_to
_status_id_str\":null,\"in_reply_to_user_id\":null,\"in_reply_to_user_id_str\":null,\"in_rep
ly_to_screen_name\":null,\"geo\":null,\"coordinates\":null,\"place\":null,\"contributors\":n
ull,\"retweet_count\":0,\"entities\":{\"hashtags\":[],\"urls\":[],\"user_mentions\":[]},\"fa
vorited\":false,\"retweeted\":false},\"contributors_enabled\":false,\"is_translator\":false,
\"profile_background_color\":\"1A1B1F\",\"profile_background_image_url\":\"http:\\/\\/a0.twi
mg.com\\/images\\/themes\\/theme9\\/bg.gif\",\"profile_background_image_url_https\":\"https:
\\/\\/si0.twimg.com\\/images\\/themes\\/theme9\\/bg.gif\",\"profile_background_tile\":false,
\"profile_image_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/1835646533\\/gu44kEhi_nor
mal\",\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/1835646533\
\/gu44kEhi_normal\",\"profile_link_color\":\"2FC2EF\",\"profile_sidebar_border_color\":\"181
```

```
A1E\",\"profile_sidebar_fill_color\":\"252429\",\"profile_text_color\":\"666666\",\"profile_
use_background_image\":true,\"default_profile\":false,\"default_profile_image\":false,\"foll
owing\":false,\"follow_request_sent\":false,\"notifications\":false}";
        JSONObject             jsonWithoutProfileImageExtension             =             new
JSONObject(rawJsonWithoutProfileImageExtension);
        UserJSONImpl            userWithoutProfileImageExtension            =            new
UserJSONImpl(jsonWithoutProfileImageExtension);

        assertEquals(userWithoutProfileImageExtension.getBiggerProfileImageURL(),
"http://si0.twimg.com/profile_images/1835646533/gu44kEhi_bigger");
        assertEquals(userWithoutProfileImageExtension.getOriginalProfileImageURL(),
"http://si0.twimg.com/profile_images/1835646533/gu44kEhi");

        String                        rawJsonWithProfileImageExtension                        =
"{\"id\":613742117,\"id_str\":\"613742117\",\"name\":\"Tweedle\",\"screen_name\":\"tweedleap
p\",\"location\":\"\",\"description\":\"Twitter application for Android, follow this account
for                          updates.\\r\\n\\r\\nDeveloped                                    by
@HandlerExploit\",\"url\":\"http:\\/\\/tweedleapp.com\",\"entities\":{\"url\":{\"urls\":[{\"
url\":\"http:\\/\\/tweedleapp.com\",\"expanded_url\":null,\"indices\":[0,21]}]},\"descriptio
n\":{\"urls\":[]}},\"protected\":false,\"followers_count\":2210,\"friends_count\":1,\"listed
_count\":20,\"created_at\":\"Wed          Jun          20          20:42:48          +0000
2012\",\"favourites_count\":1,\"utc_offset\":null,\"time_zone\":null,\"geo_enabled\":false,\
"verified\":false,\"statuses_count\":323,\"lang\":\"en\",\"status\":{\"created_at\":\"Sun
Dec                        16                             06:47:33                          +0000
2012\",\"id\":280202487317790721,\"id_str\":\"280202487317790721\",\"text\":\"@bcw_  It  is
better in some aspects, but in others it is not ideal. We need to add a way to notify the
user                                                                                  better
though.\",\"source\":\"web\",\"truncated\":false,\"in_reply_to_status_id\":28020231408046489
6,\"in_reply_to_status_id_str\":\"280202314080464896\",\"in_reply_to_user_id\":101066646,\"i
n_reply_to_user_id_str\":\"101066646\",\"in_reply_to_screen_name\":\"bcw_\",\"geo\":null,\"c
oordinates\":null,\"place\":null,\"contributors\":null,\"retweet_count\":0,\"entities\":{\"h
ashtags\":[],\"urls\":[],\"user_mentions\":[{\"screen_name\":\"bcw_\",\"name\":\"Verified
Bradley\",\"id\":101066646,\"id_str\":\"101066646\",\"indices\":[0,5]}]},\"favorited\":false
,\"retweeted\":false},\"contributors_enabled\":false,\"is_translator\":false,\"profile_backg
round_color\":\"C0DEED\",\"profile_background_image_url\":\"http:\\/\\/a0.twimg.com\\/images
\\/themes\\/theme1\\/bg.png\",\"profile_background_image_url_https\":\"https:\\/\\/si0.twimg
.com\\/images\\/themes\\/theme1\\/bg.png\",\"profile_background_tile\":false,\"profile_image
_url\":\"http:\\/\\/a0.twimg.com\\/profile_images\\/2433552309\\/dltv4u9hajvoxsne5bly_normal
.png\",\"profile_image_url_https\":\"https:\\/\\/si0.twimg.com\\/profile_images\\/2433552309
\\/dltv4u9hajvoxsne5bly_normal.png\",\"profile_link_color\":\"0084B4\",\"profile_sidebar_bor
der_color\":\"C0DEED\",\"profile_sidebar_fill_color\":\"DDEEF6\",\"profile_text_color\":\"33
3333\",\"profile_use_background_image\":true,\"default_profile\":true,\"default_profile_imag
e\":false,\"following\":true,\"follow_request_sent\":false,\"notifications\":false}";
        JSONObject             jsonWithProfileImageExtension             =             new
JSONObject(rawJsonWithProfileImageExtension);
        UserJSONImpl            userWithProfileImageExtension            =            new
UserJSONImpl(jsonWithProfileImageExtension);

        assertEquals(userWithProfileImageExtension.getBiggerProfileImageURL(),
"http://si0.twimg.com/profile_images/2433552309/dltv4u9hajvoxsne5bly_bigger.png");
        assertEquals(userWithProfileImageExtension.getOriginalProfileImageURL(),
"http://si0.twimg.com/profile_images/2433552309/dltv4u9hajvoxsne5bly.png");
    }
}
```

Version: fbe7241af8e00776d8128555486d795c03d99d62

Parents:

4e2eb32f5fdbc4c9a7ead4060da2f1f4cb0e781e

2a4bcaaae44eededf2a5b1da1d4ac46fcc001868

Merge base:

aaa0f9a9ac5eac6841f95da337239d024c02d7a2

twitter4j/twitter4j-stream/src/main/java/twitter4j/TwitterStreamImpl.java

Chunk 16: (none/ commentary, method declaration)

```
     */
<<<<<<< HEAD
    @Override
=======
    public JSONStream getJSONStream(String endpoint) throws TwitterException {
        ensureAuthorizationEnabled();
        try {
            return  new  JSONStreamImpl(getDispatcher(),  http.get(conf.getStreamBaseURL()  +
endpoint,
                    auth), conf);
        } catch (IOException e) {
            throw new TwitterException(e);
        }
    }

    /**
     * {@inheritDoc}
     */
    public InputStream getRawInputStream(String endpoint) throws TwitterException {
        ensureAuthorizationEnabled();
        return http.get(conf.getStreamBaseURL() + endpoint, auth).asStream();
    }

    /**
     * {@inheritDoc}
     */
>>>>>>> 2a4bcaaae44eededf2a5b1da1d4ac46fcc001868
    public void user() {
```

```
     */
    @Override
    public StatusStream getSampleStream() throws TwitterException {
        ensureAuthorizationEnabled();
        try {
            return new StatusStreamImpl(getDispatcher(), http.get(conf.getStreamBaseURL() +
"statuses/sample.json?"
                    + stallWarningsGetParam, auth), conf);
        } catch (IOException e) {
            throw new TwitterException(e);
        }
    }

    /**
     * {@inheritDoc}
```

```
     */
    public void user() {
```

Version: 695951bf58f8ad20525054e5be68cd677074a0a6

Parents:

09e1f40040f80e67e1b0ebe173d13bb2ffc59f5f

8f7659f4bc81996d57278c445fdc50bd76d4d58c

Merge base:

90adbc8dbb32a00bbc268d2109c2872835a825dc

twitter4j/twitter4j-appengine/src/main/java/twitter4j/internal/json/LazyDirectMessage.java

Chunk 17: (version 1/Import declaration)

```
package twitter4j.internal.json;

<<<<<<< HEAD
import twitter4j.*;
=======
import twitter4j.DirectMessage;
import twitter4j.HashtagEntity;
import twitter4j.MediaEntity;
import twitter4j.RateLimitStatus;
import twitter4j.TwitterException;
import twitter4j.TwitterRuntimeException;
import twitter4j.URLEntity;
import twitter4j.User;
import twitter4j.UserMentionEntity;

>>>>>>> 8f7659f4bc81996d57278c445fdc50bd76d4d58c
import javax.annotation.Generated;
```

```
package twitter4j.internal.json;

import twitter4j.*;
import javax.annotation.Generated;
```

twitter4j/twitter4j-core/src/main/java/twitter4j/internal/json/DirectMessageJSONImpl.java

Chunk 18: (none/variable)

```
    private String recipientScreenName;
<<<<<<< HEAD
=======

>>>>>>> 8f7659f4bc81996d57278c445fdc50bd76d4d58c
    private UserMentionEntity[] userMentionEntities;
```

```
    private String recipientScreenName;
    private UserMentionEntity[] userMentionEntities;
```

Version: 09e1f40040f80e67e1b0ebe173d13bb2ffc59f5f

Parents:

8926bd9547d09cd2bc9cdf58eb6b5ea398caf7a6

bf10ec99c1b635028260b69bf388823d57ce9a77

Merge base:

65372621cdd47dd9063bf6a661f4c29bc2db7e15

twitter4j/twitter4j-appengine/src/main/java/twitter4j/internal/json/LazyDirectMessage.java

Chunk 19: (version 1/import declaration)

```
package twitter4j.internal.json;

<<<<<<< HEAD
import twitter4j.*;
=======
import twitter4j.DirectMessage;
import twitter4j.HashtagEntity;
import twitter4j.MediaEntity;
import twitter4j.RateLimitStatus;
import twitter4j.TwitterException;
import twitter4j.TwitterRuntimeException;
import twitter4j.URLEntity;
import twitter4j.User;
import twitter4j.UserMentionEntity;
>>>>>>> bf10ec99c1b635028260b69bf388823d57ce9a77

import javax.annotation.Generated;
```

```
package twitter4j.internal.json;

import twitter4j.*;
import javax.annotation.Generated;
```

Version: 62c60c6fea7542c475866ea0bad2b5c46a9ee28b

Parents:

4f30c6c68ec1fffec232c163ee4cc6351f1e1076

33d648fcb367938919c5ba25f00e7fd49b2351a9

Merge base:

71a4748994ee552d0e51d0b2dca535cb622ebd95

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterImpl.java

Chunk 20: (new code/ return statement)

```
    public ResponseList<Status> getFavorites(long userId, Paging paging) throws
TwitterException {
<<<<<<< HEAD
        return              factory.createStatusList(get(conf.getRestBaseURL()              +
"favorites/list.json?user_id=" + userId, paging.asPostParameterArray()));
=======
        ensureAuthorizationEnabled();
        return factory.createStatusList(get(conf.getRestBaseURL() + "favorites/list.json" ,
                mergeParameters(new HttpParameter[]{new HttpParameter("user_id", userId)}
                        , paging.asPostParameterArray())));
>>>>>>> 33d648fcb367938919c5ba25f00e7fd49b2351a9
    }
```

```
    public ResponseList<Status> getFavorites(long userId) throws TwitterException {
        return              factory.createStatusList(get(conf.getRestBaseURL()              +
"favorites/list.json?user_id=" + userId));
    }
```

Chunk 21: (new code/return statement)

```
    public ResponseList<Status> getFavorites(String screenName, Paging paging) throws
TwitterException {
<<<<<<< HEAD
        return              factory.createStatusList(get(conf.getRestBaseURL()              +
"favorites/list.json?screen_name=" + screenName, paging.asPostParameterArray()));
=======
        ensureAuthorizationEnabled();
        return factory.createStatusList(get(conf.getRestBaseURL() + "favorites/list.json" ,
                mergeParameters(new     HttpParameter[]{new     HttpParameter("screen_name",
screenName)}
                        , paging.asPostParameterArray())));
>>>>>>> 33d648fcb367938919c5ba25f00e7fd49b2351a9
    }
```

```
    public ResponseList<Status> getFavorites(String screenName) throws TwitterException {
        return              factory.createStatusList(get(conf.getRestBaseURL()              +
"favorites/list.json?screen_name=" + screenName));
    }
```

Version: b2a99e8e41404709b31cadf62f77c737b42b8f4e

Parents:

4eecf47cd13ccb37b690751c5e41f9d05a931282

d2e01c6146f7f3cf99bea81b0a24fe356eb92907

Merge base:

90adbc8dbb32a00bbc268d2109c2872835a825dc

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterImpl.java

Chunk 22: (version 1/ annotation, method declaration)

```
     */
<<<<<<< HEAD
    @Override
    public IDs getFriendsIDs(long userId, long cursor) throws TwitterException {
        ensureAuthorizationEnabled();
        return  factory.createIDs(get(conf.getRestBaseURL()  +  "friends/ids.json?user_id="  +
userId +
                "&cursor=" + cursor));
=======
    public Status showStatus(long id) throws TwitterException {
        return  factory.createStatus(get(conf.getRestBaseURL()  +  "statuses/show/"  +  id  +
".json?include_my_retweet=1&include_entities="
                + conf.isIncludeEntitiesEnabled()));
>>>>>>> d2e01c6146f7f3cf99bea81b0a24fe356eb92907
    }
```

```
    */
    @Override
    public IDs getFriendsIDs(long userId, long cursor) throws TwitterException {
        ensureAuthorizationEnabled();
        return  factory.createIDs(get(conf.getRestBaseURL()  +  "friends/ids.json?user_id="  +
userId +
                "&cursor=" + cursor));
    }
```

twitter4j/twitter4j-core/src/main/java/twitter4j/internal/json/StatusJSONImpl.java

Chunk 23: (combination/variable)

```
    private long retweetCount;
<<<<<<< HEAD
    private boolean wasRetweetedByMe;
    private boolean isPossiblySensitive;
=======
>>>>>>> d2e01c6146f7f3cf99bea81b0a24fe356eb92907

    private long[] contributorsIDs;
```

```
    private long retweetCount;
    private boolean isPossiblySensitive;

    private long[] contributorsIDs;
```

Chunk 24: (new code/other)

```
                ", retweetCount=" + retweetCount +
<<<<<<< HEAD
                ", wasRetweetedByMe=" + wasRetweetedByMe +
                ", isPossiblySensitive=" + isPossiblySensitive +
                ", contributorsIDs=" + contributorsIDs +
=======
                ", myRetweetedStatus=" + myRetweetedStatus +
                ",    contributors="    +    (contributorsIDs    ==    null    ?    null    :
Arrays.asList(contributorsIDs)) +
                ", annotations=" + annotations +
>>>>>>> d2e01c6146f7f3cf99bea81b0a24fe356eb92907
                ", retweetedStatus=" + retweetedStatus +
```

```
                ", retweetCount=" + retweetCount +
                ", isPossiblySensitive=" + isPossiblySensitive +
                ", contributorsIDs=" + contributorsIDs +
                ", retweetedStatus=" + retweetedStatus +
```

It is also possible to see the variables that are in the class, but the better result is the manual one.

Version: 6ad6899312799114ce763d3cee568b2135acc518

Parents:

4492173d3eb4a48991b6eeb1e051143ee3e6b21e

8fc7159fb8315e90f5799468f55afeebe9d449c7

Merge base:

6a3f7174054845bdc9865035959737d78bc98f2a

twitter4j/twitter4j-core/src/main/java/twitter4j/internal/http/HTMLEntity.java

Chunk 25: (version 2/method declaration)

```
public final class HTMLEntity {
<<<<<<< HEAD
    public static String escape(String original) {
        StringBuilder buf = new StringBuilder(original);
        escape(buf);
        return buf.toString();
    }

    public static void escape(StringBuilder original) {
        int index = 0;
        String escaped;
        while (index < original.length()) {
            escaped = entityEscapeMap.get(original.substring(index, index + 1));
            if (escaped != null) {
                original.replace(index, index + 1, escaped);
                index += escaped.length();
            } else {
                index++;
            }
        }
=======
        public static String escape(String original) {
                return HTMLEntityString.escape(original).getConvertedText().toString();
    }

    public static void escape(StringBuffer original) {
                HTMLEntityString.escape(original);
>>>>>>> 8fc7159fb8315e90f5799468f55afeebe9d449c7
    }
```

```
public final class HTMLEntity {
    public static String escape(String original) {
                return HTMLEntityString.escape(original).getConvertedText().toString();
    }

    public static void escape(StringBuilder original) {
                HTMLEntityString.escape(original);
    }

    public static String unescape(String original) {
```

Chunk 26: (version 2/method declaration)

```
    }

<<<<<<< HEAD
```

```
    public static void unescape(StringBuilder original) {
        int index = 0;
        int semicolonIndex;
        String escaped;
        String entity;
        while (index < original.length()) {
            index = original.indexOf("&", index);
            if (-1 == index) {
                break;
            }
            semicolonIndex = original.indexOf(";", index);
            if (-1 != semicolonIndex) {
                escaped = original.substring(index, semicolonIndex + 1);
                entity = escapeEntityMap.get(escaped);
                if (entity != null) {
                    original.replace(index, semicolonIndex + 1, entity);
                }
                index++;
            } else {
                break;
            }
        }
=======
    public static void unescape(StringBuffer original) {
        HTMLEntityString.unescape(original);
>>>>>>> 8fc7159fb8315e90f5799468f55afeebe9d449c7
    }
```

```
    }

    public static void unescape(StringBuilder original) {
        HTMLEntityString.unescape(original);
    }
```

twitter4j/twitter4j-core/src/main/java/twitter4j/internal/json/MediaEntityJSONImpl.java

Chunk 27: (version 2/method signature)

```
    private String type;

<<<<<<< HEAD
    MediaEntityJSONImpl(JSONObject json) throws TwitterException {
=======
    public MediaEntityJSONImpl(HTMLEntityString.IndexMapper indexMapper, JSONObject json)
throws TwitterException {
>>>>>>> 8fc7159fb8315e90f5799468f55afeebe9d449c7
        try {
```

```
    private String type;

    public MediaEntityJSONImpl(HTMLEntityString.IndexMapper indexMapper, JSONObject json)
throws TwitterException {
        try {
```

twitter4j/twitter4j-core/src/main/java/twitter4j/internal/json/StatusJSONImpl.java

Chunk 28: (version 2/Import declaration)

```
import java.util.Date;
```

```
<<<<<<< HEAD
import static twitter4j.internal.util.z_T4JInternalParseUtil.*;
=======
import static twitter4j.internal.util.z_T4JInternalParseUtil.getBoolean;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getDate;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getUnescapedEntityString;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getLong;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getUnescapedString;
>>>>>>> 8fc7159fb8315e90f5799468f55afeebe9d449c7

/**
```

```
import java.util.Date;

import static twitter4j.internal.util.z_T4JInternalParseUtil.getBoolean;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getDate;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getUnescapedEntityString;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getLong;
import static twitter4j.internal.util.z_T4JInternalParseUtil.getUnescapedString;

/**
```

Version: 92708a174b7c103d12f911d331e7e9c23596ecad

Parents:

2b2236f0bef880fbc529dcabfd5b37845ba40db8

39c5d428350c6ce21374566e41dd0eed11709088

Merge base:

ab332e7eb076ac9427c47758215a300e0416c746

twitter4j/twitter4j-core/src/main/java/twitter4j/Status.java

Chunk 29: (combination/ commentary, method interface)

```
    boolean isRetweetedByMe();

<<<<<<< HEAD
=======
     /**
     * Returns true if the status contains a link that is identified as sensitive.
     *
     * @return whether the status contains sensitive links
     * @since Twitter4J 3.0.0
     */
    boolean isPossiblySensitive();

    /**
     * Returns the annotations, or null if no annotations are associated with this status.
     *
     * @since Twitter4J 2.1.4
     *     @deprecated     Annotations     is     not     available     for     now.     <a
href="http://groups.google.com/group/twitter-development-
talk/browse_thread/thread/4d5ff2ec4d2ce4a7">Annotations - Twitter Development Talk | Google
Groups</a>
     */
    Annotations getAnnotations();
>>>>>>> 39c5d428350c6ce21374566e41dd0eed11709088
}
```

```
    boolean isRetweetedByMe();

     /**
     * Returns true if the status contains a link that is identified as sensitive.
     *
     * @return whether the status contains sensitive links
     * @since Twitter4J 3.0.0
     */
    boolean isPossiblySensitive();
}
```

Version: 8724fa85f0454c84024745394da849ca1a9ffae4

Parents:

c4abf92b056b498d9e7cc9fcbf6442622b2cb552

8ea2136dad8700c3ef30dd9e4b21299a3c41bf2a

Merge base:

04d1babf45ca2ca97bf826f95934b8109af16998

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterImpl.java

Chunk 30: (combination/ if statement, method invocation, return statement, try statement)

```
        ensureAuthorizationEnabled();
<<<<<<< HEAD
        return factory.createUserList(get(conf.getRestBaseURL() +
                "users/lookup.json", new HttpParameter[]{
                new HttpParameter("screen_name", z_T4JInternalStringUtil.join(screenNames))
                , INCLUDE_ENTITIES}));
=======
        try {
            return factory.createUserList(get(conf.getRestBaseURL() +
                    "users/lookup.json", new HttpParameter[]{
                    new                                         HttpParameter("screen_name",
T4JInternalStringUtil.join(screenNames))
                    , INCLUDE_ENTITIES}));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
>>>>>>> 8ea2136dad8700c3ef30dd9e4b21299a3c41bf2a
    }
```

```
        ensureAuthorizationEnabled();
        try {
            return factory.createUserList(get(conf.getRestBaseURL() +
                    "users/lookup.json", new HttpParameter[]{
                    new                                         HttpParameter("screen_name",
z_T4JInternalStringUtil.join(screenNames))
                    , INCLUDE_ENTITIES}));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
    }
```

Chunk 31: (combination/ if statement, method invocation, return statement, try statement)

```
        ensureAuthorizationEnabled();
<<<<<<< HEAD
```

```
        return factory.createUserList(get(conf.getRestBaseURL() +
                "users/lookup.json", new HttpParameter[]{
                new HttpParameter("user_id", z_T4JInternalStringUtil.join(ids))
                , INCLUDE_ENTITIES}));
=======
        try {
            return factory.createUserList(get(conf.getRestBaseURL() +
                    "users/lookup.json", new HttpParameter[]{
                    new HttpParameter("user_id", T4JInternalStringUtil.join(ids))
                    , INCLUDE_ENTITIES}));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
>>>>>>> 8ea2136dad8700c3ef30dd9e4b21299a3c41bf2a
    }
```

```
        ensureAuthorizationEnabled();
        try {
            return factory.createUserList(get(conf.getRestBaseURL() +
                    "users/lookup.json", new HttpParameter[]{
                    new HttpParameter("user_id", z_T4JInternalStringUtil.join(ids))
                    , INCLUDE_ENTITIES}));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
    }
```

Chunk 32: (combination/ if statement, method invocation, return statement, try statement)

```
        ensureAuthorizationEnabled();
<<<<<<< HEAD
        return factory.createFriendshipList(get(conf.getRestBaseURL()
                +                  "friendships/lookup.json?screen_name="                  +
z_T4JInternalStringUtil.join(screenNames)));
=======
        try {
            return factory.createFriendshipList(get(conf.getRestBaseURL()
                    +                "friendships/lookup.json?screen_name="                  +
T4JInternalStringUtil.join(screenNames)));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
>>>>>>> 8ea2136dad8700c3ef30dd9e4b21299a3c41bf2a
    }
```

```
        ensureAuthorizationEnabled();
        try {
```

```
            return factory.createFriendshipList(get(conf.getRestBaseURL()
                    +                "friendships/lookup.json?screen_name="                +
z_T4JInternalStringUtil.join(screenNames)));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
    }
```

Chunk 33: (combination/ if statement, method invocation, return statement, try statement)

```
        ensureAuthorizationEnabled();
<<<<<<< HEAD
        return            factory.createFriendshipList(get(conf.getRestBaseURL()            +
"friendships/lookup.json?user_id=" + z_T4JInternalStringUtil.join(ids)));
=======
        try {
            return          factory.createFriendshipList(get(conf.getRestBaseURL()          +
"friendships/lookup.json?user_id=" + T4JInternalStringUtil.join(ids)));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
>>>>>>> 8ea2136dad8700c3ef30dd9e4b21299a3c41bf2a
    }
```

```
        ensureAuthorizationEnabled();
        try {
            return          factory.createFriendshipList(get(conf.getRestBaseURL()          +
"friendships/lookup.json?user_id=" + z_T4JInternalStringUtil.join(ids)));
        } catch (TwitterException te) {
            if (404 == te.getStatusCode()) {
                return factory.createEmptyResponseList(te);
            } else {
                throw te;
            }
        }
    }
```

Version: 98caafc967fa4ec339da2fc2f4686d6c1c78b69a

Parents:

e31a68eda44627daca1262e626daa69ca8e477a5

ea9d8b95030225896bee32518406a4344eaf2655

Merge base:

5a66487555f83b61e3d26e211faab6864219fc0b

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterImpl.java

Chunk 34: (version 1/method invocation, return statement, try statement)

```
                + categorySlug + ".json");
<<<<<<< HEAD
            return factory.createUserListFromJSONArray_Users(res);
=======
        try {
            return factory.createUserList(res.asJSONObject().getJSONArray("users"), res);
        } catch (JSONException jsone) {
            throw new TwitterException(jsone);
        }
>>>>>>> ea9d8b95030225896bee32518406a4344eaf2655
    }
```

```
                + categorySlug + ".json");
            return factory.createUserListFromJSONArray_Users(res);
    }
```

Chunk 35: (version 1/method invocation, return statement)

```
                + categorySlug + "/members.json");
<<<<<<< HEAD
        return factory.createUserListFromJSONArray(res);
=======
        return factory.createUserList(res.asJSONArray(), res);
>>>>>>> ea9d8b95030225896bee32518406a4344eaf2655
    }
```

```
                + categorySlug + "/members.json");
        return factory.createUserListFromJSONArray(res);
    }
```

Chunk 36: (new code/other)

```
                + conf.isIncludeEntitiesEnabled() + "&user_id=" + userId
<<<<<<< HEAD
                + "&cursor=" + cursor));
=======
                + "&cursor=" + cursor, null));
>>>>>>> ea9d8b95030225896bee32518406a4344eaf2655
    }
```

```
                + conf.isIncludeEntitiesEnabled() + "&cursor=" + cursor));
    }
```

twitter4j/twitter4j-core/src/main/java/twitter4j/internal/util/ParseUtil.java

Chunk 37: (version 2/import declaration)

```
import twitter4j.internal.http.HttpResponse;
<<<<<<< HEAD
import twitter4j.internal.http.HttpResponseCode;
=======
>>>>>>> ea9d8b95030225896bee32518406a4344eaf2655
import twitter4j.internal.org.json.JSONException;
```

```
import twitter4j.internal.http.HttpResponse;
import twitter4j.internal.org.json.JSONException;
```

Version: d3156b94736b942639b6d6a64fcd2e1d9590be29

Parents:

174037a1e41e18d40bac2f817bf37790c83e4f15

a10d5b7a0e17857c806b9f514fd21d6a93f825f1

Merge base:

895cf30f16437e50637074629402cdac90155e2e

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterImpl.java

Chunk 38: (combination/annotation, commentary, method declaration)

```
                }), conf);
<<<<<<< HEAD
=======
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public IDs getNoRetweetIds() throws TwitterException {
        ensureAuthorizationEnabled();
        return             new             IDsJSONImpl(get(conf.getRestBaseURL()             +
"friendships/no_retweet_ids.json"), conf);
>>>>>>> a10d5b7a0e17857c806b9f514fd21d6a93f825f1
    }
```

```
                }), conf);
    }

    /**
     * {@inheritDoc}
     */
    public IDs getNoRetweetIds() throws TwitterException {
        ensureAuthorizationEnabled();
        return             new             IDsJSONImpl(get(conf.getRestBaseURL()             +
"friendships/no_retweet_ids.json"), conf);
    }
```

Chunk 39: (version 2/ other)

```
        return             new             UserJSONImpl(post(conf.getRestBaseURL()             +
"notifications/follow.json?include_entities="
<<<<<<< HEAD
                + conf.isIncludeEntitiesEnabled() + "&userId=" + userId), conf);
=======
                + conf.isIncludeEntitiesEnabled() + "&user_id=" + userId), conf);
>>>>>>> a10d5b7a0e17857c806b9f514fd21d6a93f825f1
    }
```

```
        return             new             UserJSONImpl(post(conf.getRestBaseURL()             +
"notifications/follow.json?include_entities="
                + conf.isIncludeEntitiesEnabled() + "&user_id=" + userId), conf);
    }
```

Version: 6d232238d7078d2f0eae1468e7b77e74f824e3d4

Parents:

a4531326eb263345012810bfb72f1b613c52e9fd

c4c57282d85fac3d0cbf9c3ca7b779bb4b5063b4

Merge base:

4a9169ceeaf2c518aaab46003f2fcaa773bafc24

twitter4j/twitter4j-stream/src/test/java/twitter4j/DAOTest.java

Chunk 40: (version 1/blank)

```
    Configuration conf = ConfigurationContext.getInstance();
<<<<<<< HEAD

=======
>>>>>>> c4c57282d85fac3d0cbf9c3ca7b779bb4b5063b4
    public DAOTest(String name) {
```

```
    Configuration conf = ConfigurationContext.getInstance();

    public DAOTest(String name) {
```

Version: d817976ac3cc6444d1d75ea69adc89c03a86c21a

Parents:

3719eac704a22383a27071ca0a396d32a8873332

888a4ff8a9ddbdf077606426e53a1125b66e79f5

Merge base:

7de89cc7221fc03aa3085bb408206051fbb44687

twitter4j/twitter4j-core/src/main/java/twitter4j/StatusAdapter.java

Chunk 41: (version 2/method declaration)

```
public class StatusAdapter implements StatusListener {
<<<<<<< HEAD
    public void onException(Exception ex) {
    }

    public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {
    }

    public void onDeletionNotice(int directMessageId, int userId) {
=======
    public void onStatus(Status status) {
    }
    public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {
>>>>>>> 888a4ff8a9ddbdf077606426e53a1125b66e79f5
    }
```

```
public class StatusAdapter implements StatusListener {
    public void onStatus(Status status) {
    }
    public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {
    }
```

twitter4j/twitter4j-core/src/main/java/twitter4j/UserStreamAdapter.java

Chunk 42: (version 2/class signature)

```
*/
<<<<<<< HEAD
public class UserStreamAdapter extends StatusAdapter {
=======
public class UserStreamAdapter extends StatusAdapter implements UserStreamListener{
>>>>>>> 888a4ff8a9ddbdf077606426e53a1125b66e79f5
    public void onFriendList(int[] friendIds) {
```

```
*/
public class UserStreamAdapter extends StatusAdapter implements UserStreamListener{
    public void onFriendList(int[] friendIds) {
```

Version: 17dc888cdb95ef89615a740d2ac8ebd47b4ff107

Parents:

a0bf4e00e235089fedd13daf9cbb1d7d7fd50258

e6b5de185832910f10b727e8af35768492d2e21e

Merge base:

f0299deabfe7fc79484aceb357a18a5a00e2e918

twitter4j/twitter4j-core/src/main/java/twitter4j/User.java

Chunk 43: (version 1/commentary, method interface)

```
    boolean isVerified();
<<<<<<< HEAD

    boolean isFollowing();

    /**
     * Returns the number of public lists the user is listed on, or -1
     * if the count is unavailable.
     *
     * @return the number of public lists the user is listed on.
     * @since Twitter4J 2.1.4
     */
    int getListedCount();

    /**
     * Returns true if the authenticating user has requested to follow this user,
     * otherwise false.
     *
     * @return true if the authenticating user has requested to follow this user.
     */
    boolean isFollowRequestSent();
=======
>>>>>>> e6b5de185832910f10b727e8af35768492d2e21e
}
```

```
    boolean isVerified();

    boolean isFollowing();

    /**
     * Returns the number of public lists the user is listed on, or -1
     * if the count is unavailable.
     *
     * @return the number of public lists the user is listed on.
     * @since Twitter4J 2.1.4
     */
    int getListedCount();

    /**
     * Returns true if the authenticating user has requested to follow this user,
     * otherwise false.
     *
     * @return true if the authenticating user has requested to follow this user.
     * @since Twitter4J 2.1.4
     */
    boolean isFollowRequestSent();
```

```
}
```

twitter4j/twitter4j-core/src/main/java/t witter4j/UserJSONImpl.java

Chunk 44: (version 1/variable)

```
    private boolean isVerified;
<<<<<<< HEAD
    private boolean isFollowing;
    private int listedCount;
    private boolean isFollowRequestSent;
=======
>>>>>>> e6b5de185832910f10b727e8af35768492d2e21e
    private static final long serialVersionUID = -6345893237975349030L;
```

```
    private boolean isVerified;
    private boolean isFollowing;
    private int listedCount;
    private boolean isFollowRequestSent;
    private static final long serialVersionUID = -6345893237975349030L;
```

Chunk 45: (version 1/method declaration)

```
}

<<<<<<< HEAD
    /**
     * {@inheritDoc}
     */
    public boolean isFollowing() {
        return isFollowing;
    }

    /**
     * {@inheritDoc}
     */
    public int getListedCount() {
        return listedCount;
    }

    /**
     * {@inheritDoc}
     */
    public boolean isFollowRequestSent() {
        return isFollowRequestSent;
    }

=======
>>>>>>> e6b5de185832910f10b727e8af35768492d2e21e
    /*package*/  static  PagableResponseList<User>  createPagableUserList(HttpResponse  res)
throws TwitterException {
```

```
    }

    /**
     * {@inheritDoc}
     */
    public boolean isFollowing() {
        return isFollowing;
    }
```

```
    /**
     * {@inheritDoc}
     */
    public int getListedCount() {
        return listedCount;
    }

    /**
     * {@inheritDoc}
     */
    public boolean isFollowRequestSent() {
        return isFollowRequestSent;
    }

    /*package*/ static PagableResponseList<User> createPagableUserList(HttpResponse res)
throws TwitterException {
```

Version: a0bf4e00e235089fedd13daf9cbb1d7d7fd50258

Parents:

d6c69d11bb3f6a984dbfd0d161e802ec88ec3e72

10f015db1674e246b012ef1292bca98c85458771

Merge base:

5c7e849c7551364a6cd9c3d3d21dce92f2c5a419

twitter4j/twitter4j-core/src/main/java/twitter4j/Status.java

Chunk 46: (concatenation/ commentary, method interface)

```
    /**
<<<<<<< HEAD
     * Returns the number of times this tweet has been retweeted, or -1 when the tweet was
     * created before this feature was enabled.
     *
     * @return the retweet count.
     */
    long getRetweetCount();

    /**
     * Returns true if the authenticating user has retweeted this tweet, or false when the
tweet was
     * created before this feature was enabled.
     *
     * @return whether the authenticating user has retweeted this tweet.
     * @since Twitter4J 2.1.4
     */
    boolean wasRetweetedByMe();

    /**
     * Returns an array of users mentioned in the tweet, or null if no users were mentioned.
     * Note that these users only have data for ID, screen name, and name.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    User[] getUserMentions();

    /**
     * Returns an array of URLs mentioned in the tweet, or null if no URLs were mentioned.
     *
     * @return An array of URLs mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    URL[] getURLs();

    /**
     * Returns an array of hashtags mentioned in the tweet, or null if no hashtags were
mentioned.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    String[] getHashtags();
=======
     * Returns the annotations, or null if no annotations are associated with this status.
     * @since Twitter4J 2.1.4
```

```
     */
    Annotations getAnnotations();
>>>>>>> 10f015db1674e246b012ef1292bca98c85458771
}
```

```
    /**
     * Returns the number of times this tweet has been retweeted, or -1 when the tweet was
     * created before this feature was enabled.
     *
     * @return the retweet count.
     */
    long getRetweetCount();

    /**
     * Returns true if the authenticating user has retweeted this tweet, or false when the
tweet was
     * created before this feature was enabled.
     *
     * @return whether the authenticating user has retweeted this tweet.
     * @since Twitter4J 2.1.4
     */
    boolean wasRetweetedByMe();

    /**
     * Returns an array of users mentioned in the tweet, or null if no users were mentioned.
     * Note that these users only have data for ID, screen name, and name.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    User[] getUserMentions();

    /**
     * Returns an array of URLs mentioned in the tweet, or null if no URLs were mentioned.
     *
     * @return An array of URLs mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    URL[] getURLs();

    /**
     * Returns an array of hashtags mentioned in the tweet, or null if no hashtags were
mentioned.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    String[] getHashtags();

    /**
     * Returns the annotations, or null if no annotations are associated with this status.
     * @since Twitter4J 2.1.4
     */
    Annotations getAnnotations();
}
```

twitter4j/twitter4j-core/src/main/java/twitter4j/StatusJSONImpl.java

Chunk 47: (version 1/if statement, try statement)

```
        }
```

```
<<<<<<< HEAD
        if (!json.isNull("entities")) {
            try {
                JSONObject entities = json.getJSONObject("entities");

                JSONArray userMentionsArray = entities.getJSONArray("user_mentions");
                userMentions = new User[userMentionsArray.length()];
                for(int i=0;i<userMentionsArray.length();i++){
                    userMentions[i] = new UserJSONImpl(userMentionsArray.getJSONObject(i));
                }

                JSONArray urlArray = entities.getJSONArray("urls");
                urls = new URL[urlArray.length()];
                for(int i=0;i<urlArray.length();i++){
                    try {
                        urls[i] = new URL(urlArray.getJSONObject(i).getString("url"));
                    } catch (MalformedURLException e) {
                        urls[i] = null;
                    }
                }

                JSONArray hashtagsArray = entities.getJSONArray("hashtags");
                hashtags = new String[hashtagsArray.length()];
                for(int i=0;i<hashtagsArray.length();i++){
                    hashtags[i] = hashtagsArray.getJSONObject(i).getString("text");
                }
=======
        if (!json.isNull("annotations")) {
            try {
                JSONArray annotationsArray = json.getJSONArray("annotations");
                annotations = new Annotations(annotationsArray);
>>>>>>> 10f015db1674e246b012ef1292bca98c85458771
            } catch (JSONException ignore) {
```

```
        }
        if (!json.isNull("entities")) {
            try {
                JSONObject entities = json.getJSONObject("entities");

                JSONArray userMentionsArray = entities.getJSONArray("user_mentions");
                userMentions = new User[userMentionsArray.length()];
                for(int i=0;i<userMentionsArray.length();i++){
                    userMentions[i] = new UserJSONImpl(userMentionsArray.getJSONObject(i));
                }

                JSONArray urlArray = entities.getJSONArray("urls");
                urls = new URL[urlArray.length()];
                for(int i=0;i<urlArray.length();i++){
                    try {
                        urls[i] = new URL(urlArray.getJSONObject(i).getString("url"));
                    } catch (MalformedURLException e) {
                        urls[i] = null;
                    }
                }

                JSONArray hashtagsArray = entities.getJSONArray("hashtags");
                hashtags = new String[hashtagsArray.length()];
                for(int i=0;i<hashtagsArray.length();i++){
                    hashtags[i] = hashtagsArray.getJSONObject(i).getString("text");
                }
            } catch (JSONException ignore) {
```

Version: a0bf4e00e235089fedd13daf9cbb1d7d7fd50258

Parents:

d6c69d11bb3f6a984dbfd0d161e802ec88ec3e72

10f015db1674e246b012ef1292bca98c85458771

Merge base:

5c7e849c7551364a6cd9c3d3d21dce92f2c5a419

twitter4j/twitter4j-core/src/main/java/twitter4j/Status.java

Chunk 48: (concatenation/commentary, method interface)

```
    /**
<<<<<<< HEAD
     * Returns the number of times this tweet has been retweeted, or -1 when the tweet was
     * created before this feature was enabled.
     *
     * @return the retweet count.
     */
    long getRetweetCount();

    /**
     * Returns true if the authenticating user has retweeted this tweet, or false when the
tweet was
     * created before this feature was enabled.
     *
     * @return whether the authenticating user has retweeted this tweet.
     * @since Twitter4J 2.1.4
     */
    boolean wasRetweetedByMe();

    /**
     * Returns an array of users mentioned in the tweet, or null if no users were mentioned.
     * Note that these users only have data for ID, screen name, and name.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    User[] getUserMentions();

    /**
     * Returns an array of URLs mentioned in the tweet, or null if no URLs were mentioned.
     *
     * @return An array of URLs mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    URL[] getURLs();

    /**
     * Returns an array of hashtags mentioned in the tweet, or null if no hashtags were
mentioned.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    String[] getHashtags();
=======
     * Returns the annotations, or null if no annotations are associated with this status.
```

```
     * @since Twitter4J 2.1.4
     */
    Annotations getAnnotations();
>>>>>>> 10f015db1674e246b012ef1292bca98c85458771
}
```

```
    /**
     * Returns the number of times this tweet has been retweeted, or -1 when the tweet was
     * created before this feature was enabled.
     *
     * @return the retweet count.
     */
    long getRetweetCount();

    /**
     * Returns true if the authenticating user has retweeted this tweet, or false when the
tweet was
     * created before this feature was enabled.
     *
     * @return whether the authenticating user has retweeted this tweet.
     * @since Twitter4J 2.1.4
     */
    boolean wasRetweetedByMe();

    /**
     * Returns an array of users mentioned in the tweet, or null if no users were mentioned.
     * Note that these users only have data for ID, screen name, and name.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    User[] getUserMentions();

    /**
     * Returns an array of URLs mentioned in the tweet, or null if no URLs were mentioned.
     *
     * @return An array of URLs mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    URL[] getURLs();

    /**
     * Returns an array of hashtags mentioned in the tweet, or null if no hashtags were
mentioned.
     *
     * @return An array of users mentioned in the tweet.
     * @since Twitter4J 2.1.4
     */
    String[] getHashtags();

    /**
     * Returns the annotations, or null if no annotations are associated with this status.
     * @since Twitter4J 2.1.4
     */
    Annotations getAnnotations();
}
```

twitter4j/twitter4j-core/src/main/java/twitter4j/StatusJSONImpl.java

Chunk 49: (new code/If statement, Try statement)

```
        }
<<<<<<< HEAD
        if (!json.isNull("entities")) {
            try {
                JSONObject entities = json.getJSONObject("entities");

                JSONArray userMentionsArray = entities.getJSONArray("user_mentions");
                userMentions = new User[userMentionsArray.length()];
                for(int i=0;i<userMentionsArray.length();i++){
                    userMentions[i] = new UserJSONImpl(userMentionsArray.getJSONObject(i));
                }

                JSONArray urlArray = entities.getJSONArray("urls");
                urls = new URL[urlArray.length()];
                for(int i=0;i<urlArray.length();i++){
                    try {
                        urls[i] = new URL(urlArray.getJSONObject(i).getString("url"));
                    } catch (MalformedURLException e) {
                        urls[i] = null;
                    }
                }

                JSONArray hashtagsArray = entities.getJSONArray("hashtags");
                hashtags = new String[hashtagsArray.length()];
                for(int i=0;i<hashtagsArray.length();i++){
                    hashtags[i] = hashtagsArray.getJSONObject(i).getString("text");
                }
=======
        if (!json.isNull("annotations")) {
            try {
                JSONArray annotationsArray = json.getJSONArray("annotations");
                annotations = new Annotations(annotationsArray);
>>>>>>> 10f015db1674e246b012ef1292bca98c85458771
            } catch (JSONException ignore) {
```

```
        }
        if (!json.isNull("entities")) {
            try {
                JSONObject entities = json.getJSONObject("entities");

                JSONArray userMentionsArray = entities.getJSONArray("user_mentions");
                userMentions = new User[userMentionsArray.length()];
                for(int i=0;i<userMentionsArray.length();i++){
                    userMentions[i] = new UserJSONImpl(userMentionsArray.getJSONObject(i));
                }

                JSONArray urlArray = entities.getJSONArray("urls");
                urls = new URL[urlArray.length()];
                for(int i=0;i<urlArray.length();i++){
                    try {
                        urls[i] = new URL(urlArray.getJSONObject(i).getString("url"));
                    } catch (MalformedURLException e) {
                        urls[i] = null;
                    }
                }

                JSONArray hashtagsArray = entities.getJSONArray("hashtags");
```

```
                hashtags = new String[hashtagsArray.length()];
                for(int i=0;i<hashtagsArray.length();i++){
                    hashtags[i] = hashtagsArray.getJSONObject(i).getString("text");
                }
            } catch (JSONException ignore) {
            }
        }
        if (!json.isNull("annotations")) {
            try {
                JSONArray annotationsArray = json.getJSONArray("annotations");
                annotations = new Annotations(annotationsArray);
            } catch (JSONException ignore) {
```

Version: d6c69d11bb3f6a984dbfd0d161e802ec88ec3e72

Parents:

d5acff3ff861b3e99e3c65f789160039216e8386

f28c58773d47618bade9fb6c024bfa025b0d1829

Merge base:

f0299deabfe7fc79484aceb357a18a5a00e2e918

twitter4j/twitter4j-core/src/main/java/twitter4j/User.java

Chunk 50: (version 2/ commentary, method interface)

```
    boolean isFollowing();

<<<<<<< HEAD
    int getListedCount();
=======
    /**
     * Returns the number of public lists the user is listed on, or -1
     * if the count is unavailable.
     *
     * @return the number of public lists the user is listed on.
     */
    int getListedCount();

    /**
     * Returns true if the authenticating user has requested to follow this user,
     * otherwise false.
     *
     * @return true if the authenticating user has requested to follow this user.
     */
    boolean isFollowRequestSent();
>>>>>>> f28c58773d47618bade9fb6c024bfa025b0d1829
}
```

```
    boolean isFollowing();

    /**
     * Returns the number of public lists the user is listed on, or -1
     * if the count is unavailable.
     *
     * @return the number of public lists the user is listed on.
     * @since Twitter4J 2.1.4
     */
    int getListedCount();

    /**
     * Returns true if the authenticating user has requested to follow this user,
     * otherwise false.
     *
     * @return true if the authenticating user has requested to follow this user.
     */
    boolean isFollowRequestSent();
}
```

twitter4j/twitter4j-core/src/main/java/twitter4j/UserJSONImpl.java

Chunk 51:(version 2/variable)

```
    private int listedCount;
<<<<<<< HEAD
=======
    private boolean isFollowRequestSent;
>>>>>>> f28c58773d47618bade9fb6c024bfa025b0d1829
    private static final long serialVersionUID = -6345893237975349030L;
```

```
    private int listedCount;
    private boolean isFollowRequestSent;
    private static final long serialVersionUID = -6345893237975349030L;
```

Chunk 52: (version2/commentary, method declaration)

```
    }

<<<<<<< HEAD
=======
    /**
     * {@inheritDoc}
     */
    public boolean isFollowRequestSent() {
        return isFollowRequestSent;
    }

>>>>>>> f28c58773d47618bade9fb6c024bfa025b0d1829
    /*package*/  static  PagableResponseList<User>  createPagableUserList(HttpResponse  res)
throws TwitterException {
```

```
    }

    /**
     * {@inheritDoc}
     */
    public boolean isFollowRequestSent() {
        return isFollowRequestSent;
    }

    /*package*/  static  PagableResponseList<User>  createPagableUserList(HttpResponse  res)
throws TwitterException {
```

twitter4j/twitter4j-core/src/test/java/twitter4j/TwitterTestUnit.java

Chunk 53: (combination/ try statement, variable)

```
        assertEquals(52, status.getUser().getId());
<<<<<<< HEAD
        try {
            Status status2 = unauthenticated.showStatus(10001);
            assertEquals(52, status2.getUser().getId());
            assertNotNull(status.getRateLimitStatus());

            status2 = unauthenticated.showStatus(9993834691);
            assertEquals("01010100  01110010  01101001  01110101  01101101  01110000  01101000
<3", status2.getText());
            status2 = unauthenticated.showStatus(71857373721);
            assertEquals("\\u5e30%u5e30 <%}& foobar", status2.getText());
        } catch (TwitterException te) {
```

```
            assertEquals(400, te.getStatusCode());
        }
=======
        Status status2 = unauthenticated.showStatus(10001l);
        assertEquals(52, status2.getUser().getId());
        assertNotNull(status.getRateLimitStatus());

        status2 = unauthenticated.showStatus(9993834691l);
        assertEquals("01010100  01110010  01101001  01110101  01101101  01110000  01101000
<3", status2.getText());
        status2 = unauthenticated.showStatus(71857373721l);
        assertEquals("\\u5e30%u5e30 <%}& foobar",status2.getText());

        status = twitterAPI2.showStatus(10001l);
        assertTrue(-1 <= status.getRetweetCount());
        assertFalse(status.wasRetweetedByMe());
>>>>>>> f28c58773d47618bade9fb6c024bfa025b0d1829
    }
```

```
        assertEquals(52, status.getUser().getId());
        try {
            Status status2 = unauthenticated.showStatus(10001l);
            assertEquals(52, status2.getUser().getId());
            assertNotNull(status.getRateLimitStatus());

            status2 = unauthenticated.showStatus(999383469l);
            assertEquals("01010100  01110010  01101001  01110101  01101101  01110000  01101000
<3", status2.getText());
            status2 = unauthenticated.showStatus(71857373721l);
            assertEquals("\\u5e30%u5e30 <%}& foobar", status2.getText());
        } catch (TwitterException te) {
            assertEquals(400, te.getStatusCode());
        }

        status = twitterAPI2.showStatus(10001l);
        assertTrue(-1 <= status.getRetweetCount());
        assertFalse(status.wasRetweetedByMe());
    }
```

Version: e76c43809f5cfe7fba4ceff2139609dc94dd63d4

Parents:

15c017753a2fada168b7cadf0b5574a90f46815e

f0299deabfe7fc79484aceb357a18a5a00e2e918

Merge base:

ef005063a3519e88380cab9d2843c3096bcb57d5

twitter4j/twitter4j-core/src/main/java/twitter4j/util/ImageUpload.java

Chunk 54: (version 2/method declaration, method interface)

```
    public static String DEFAULT_TWITPIC_API_KEY = null;
<<<<<<< HEAD

    public abstract String upload (File image) throws TwitterException;
    public abstract String upload (File image, String message) throws TwitterException;
    public abstract String upload (String imageFileName, InputStream imageBody) throws
TwitterException;
    public abstract String upload (String imageFileName, InputStream imageBody, String
message) throws TwitterException;

    /** Returns an OAuth image uploader to img.ly */
    public static ImageUpload getImgLyUploader (OAuthAuthorization auth)
    {
        return new ImgLyOAuthUploader (auth);
    }

    /** Returns an OAuth image uploader to Twitgoo */
    public static ImageUpload getTwitgooUploader(OAuthAuthorization auth)
    {
        return new TwitgooOAuthUploader (auth);
    }

    /** Returns an image uploader to Twitpic. Handles both BasicAuth and OAuth.
     *  Note: When using OAuth, the Twitpic API Key needs to be specified, either with the
field ImageUpload.DEFAULT_TWITPIC_API_KEY,
     *    or using the getTwitpicUploader (String twitpicAPIKey, OAuthAuthorization auth)
method
=======

    public abstract String upload(File image) throws TwitterException;

    public abstract String upload(File image, String message) throws TwitterException;

    public abstract String upload(String imageFileName, InputStream imageBody) throws
TwitterException;

    public abstract String upload(String imageFileName, InputStream imageBody, String
message) throws TwitterException;


    /**
     * Returns an image uploader to Twitpic. Handles both BasicAuth and OAuth.
     * Note: When using OAuth, the Twitpic API Key needs to be specified, either with the
field ImageUpload.DEFAULT_TWITPIC_API_KEY,
     * or using the getTwitpicUploader (String twitpicAPIKey, OAuthAuthorization auth)
method
>>>>>>> f0299deabfe7fc79484aceb357a18a5a00e2e918
     */
```

```
    public static String DEFAULT_TWITPIC_API_KEY = null;


    public abstract String upload(File image) throws TwitterException;

    public abstract String upload(File image, String message) throws TwitterException;

    public abstract String upload(String imageFileName, InputStream imageBody) throws
TwitterException;

    public abstract String upload(String imageFileName, InputStream imageBody, String
message) throws TwitterException;


    /**
     * Returns an image uploader to Twitpic. Handles both BasicAuth and OAuth.
     * Note: When using OAuth, the Twitpic API Key needs to be specified, either with the
field ImageUpload.DEFAULT_TWITPIC_API_KEY,
     * or using the getTwitpicUploader (String twitpicAPIKey, OAuthAuthorization auth)
method
     */
```

Chunk 55: (version 2/commentary, method declaration)

```
    }
<<<<<<< HEAD

    /** Returns an OAuth image uploader to TweetPhoto */
    public static ImageUpload getTweetPhoto2Uploader (String tweetPhotoAPIKey,
OAuthAuthorization auth)
    {
        return new TweetPhoto2OAuthUploader (tweetPhotoAPIKey, auth);
    }

    /** Returns an image uploader to YFrog. Handles both BasicAuth and OAuth */
    public static ImageUpload getYFrogUploader (Twitter twitter) throws TwitterException
    {
        Authorization auth = twitter.getAuthorization ();
=======

    /**
     * Returns an image uploader to YFrog. Handles both BasicAuth and OAuth
     */
    public static ImageUpload getYFrogUploader(Twitter twitter) throws TwitterException {
        Authorization auth = twitter.getAuthorization();
>>>>>>> f0299deabfe7fc79484aceb357a18a5a00e2e918
        if (auth instanceof OAuthAuthorization)
```

```
    }

    /**
     * Returns an image uploader to YFrog. Handles both BasicAuth and OAuth
     */
    public static ImageUpload getYFrogUploader(Twitter twitter) throws TwitterException {
        Authorization auth = twitter.getAuthorization();
        if (auth instanceof OAuthAuthorization)
```

Chunk 56: (version 2/commentary, method signature)

```
    }
```

```
<<<<<<< HEAD
    // Described at http://groups.google.com/group/tweetphoto/web/multipart-form-data-upload
    //  and http://groups.google.com/group/tweetphoto/web/oauth-echo
    public static class TweetPhotoOAuthUploader extends ImageUpload
    {
=======
    // Described at http://dev.twitpic.com/docs/2/upload/

    public static class TweetPhotoOAuthUploader extends ImageUpload {
>>>>>>> f0299deabfe7fc79484aceb357a18a5a00e2e918
        private String tweetPhotoAPIKey;
```

```
    }

    // Described at http://dev.twitpic.com/docs/2/upload/

    private static class TwitpicOAuthUploader extends ImageUpload {
        private String twitpicAPIKey;
```

Chunk 57: (version 2/ commentary, method signature)

```
    }

<<<<<<< HEAD
    // Described at http://groups.google.com/group/tweetphoto/web/multipart-form-data-upload
    //  and http://groups.google.com/group/tweetphoto/web/oauth-echo
    public static class TweetPhotoOAuthUploader extends ImageUpload
    {
=======
    // Described at http://dev.twitpic.com/docs/2/upload/

    public static class TweetPhotoOAuthUploader extends ImageUpload {
>>>>>>> f0299deabfe7fc79484aceb357a18a5a00e2e918
        private String tweetPhotoAPIKey;
```

```
    }

    // Described at http://dev.twitpic.com/docs/2/upload/

    private static class TwitpicOAuthUploader extends ImageUpload {
        private String twitpicAPIKey;
```

Chunk 58: (New code/annotation, class signature, commentary, method declaration, variable)

```
    }
<<<<<<< HEAD

    // Described at http://groups.google.com/group/tweetphoto/web/upload-v2-0-api
    //  and http://groups.google.com/group/tweetphoto/web/oauth-echo
    public static class TweetPhoto2OAuthUploader extends ImageUpload
    {
        private String tweetPhotoAPIKey;
        private OAuthAuthorization auth;

        // uses the secure upload URL, not the one specified in the Twitpic FAQ
        private       static       final       String       TWEETPHOTO_UPLOAD_URL       =
"https://tweetphotoapi.com/api/tpapi.svc/upload2";
        private      static      final      String      TWITTER_VERIFY_CREDENTIALS      =
"https://api.twitter.com/1/account/verify_credentials.xml";
```

```
        public TweetPhoto2OAuthUploader (String tweetPhotoAPIKey, OAuthAuthorization auth)
        {
            if (tweetPhotoAPIKey == null || "".equals (tweetPhotoAPIKey))
                throw new IllegalArgumentException ("The TweetPhoto API Key supplied to the
OAuth image uploader can't be null or empty");

            this.tweetPhotoAPIKey = tweetPhotoAPIKey;
            this.auth = auth;
        }

        @Override
        public String upload (File image) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image)
                    });
        }

        @Override
        public String upload (File image, String message) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image),
                    new HttpParameter ("message", message)
                    });
        }

        @Override
        public   String   upload   (String   imageFileName,   InputStream   imageBody)   throws
TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    });
        }

        @Override
        public String upload (String imageFileName, InputStream imageBody, String message)
throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    new HttpParameter ("message", message)
                    });
        }

        private String upload (HttpParameter[] additionalParams) throws TwitterException
        {
            // step 1 - generate HTTP request headers
            String                  verifyCredentialsAuthorizationHeader                    =
generateVerifyCredentialsAuthorizationHeader ();

            Map<String, String> headers = new HashMap<String, String>();
            headers.put ("X-Auth-Service-Provider", TWITTER_VERIFY_CREDENTIALS);
            headers.put                                ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);
            headers.put("TPLAT", Double.toString(convert(34, 55, 37.50)));
            headers.put("TPLONG", Double.toString(convert(135, 45, 37.94)));

            // step 2 - generate HTTP parameters
            HttpParameter[] params = {
```

```
                    new HttpParameter ("api_key", tweetPhotoAPIKey),
                    };
            params = appendHttpParameters(params, additionalParams);

            // step 3 - upload the file
            HttpClientWrapper client = new HttpClientWrapper ();
            HttpResponse   httpResponse   =   client.post   (TWEETPHOTO_UPLOAD_URL,   params,
headers);

            // step 4 - check the response
            int statusCode = httpResponse.getStatusCode ();
            if (statusCode != 201)
                throw new TwitterException ("Twitpic image upload returned invalid status
code", httpResponse);

            String response = httpResponse.asString ();

            if (response.contains("<Error><ErrorCode>")) {
                String   error   =   response.substring(response.indexOf("<ErrorCode>")   +
"<ErrorCode>".length(), response.lastIndexOf("</ErrorCode>"));
                throw new TwitterException("TweetPhoto image upload failed with this error
message: " + error, httpResponse);
            }
            if (response.contains("<Status>OK</Status>")) {
                String   media   =   response.substring(response.indexOf("<MediaUrl>")   +
"<MediaUrl>".length(), response.indexOf("</MediaUrl>"));
                return media;
            }

            throw new TwitterException ("Unknown TweetPhoto response", httpResponse);
        }

       private double convert(int xdo, int xhun, double xbyou){
              return xdo + ( xhun * 60 + xbyou )/3600;
       }
       private String generateVerifyCredentialsAuthorizationHeader ()
        {
            List<HttpParameter> oauthSignatureParams = auth.generateOAuthSignatureHttpParams
("GET", TWITTER_VERIFY_CREDENTIALS);
            return          "OAuth          realm=\"http://api.twitter.com/\","          +
OAuthAuthorization.encodeParameters (oauthSignatureParams, ",", true);
        }
    }

    // Described at http://dev.twitpic.com/docs/2/upload/
    public static class ImgLyOAuthUploader extends ImageUpload
    {
        private OAuthAuthorization auth;

        // uses the secure upload URL, not the one specified in the Twitpic FAQ
        private static final String IMGLY_UPLOAD_URL = "http://img.ly/api/2/upload.json";
        private      static      final      String      TWITTER_VERIFY_CREDENTIALS      =
"https://api.twitter.com/1/account/verify_credentials.json";

        public ImgLyOAuthUploader (OAuthAuthorization auth)
        {
            this.auth = auth;
        }

        @Override
        public String upload (File image) throws TwitterException
        {
```

```
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image)
                    });
        }

        @Override
        public String upload (File image, String message) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image),
                    new HttpParameter ("message", message)
                    });
        }

        @Override
        public String upload (String imageFileName, InputStream imageBody) throws
TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    });
        }

        @Override
        public String upload (String imageFileName, InputStream imageBody, String message)
throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    new HttpParameter ("message", message)
                    });
        }

        private String upload (HttpParameter[] additionalParams) throws TwitterException
        {
            // step 1 - generate HTTP request headers
            String                  verifyCredentialsAuthorizationHeader                  =
generateVerifyCredentialsAuthorizationHeader ();

            Map<String, String> headers = new HashMap<String, String>();
            headers.put ("X-Auth-Service-Provider", TWITTER_VERIFY_CREDENTIALS);
            headers.put                              ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

            // step 2 - generate HTTP parameters
            HttpParameter[] params = {
                    };
            params = appendHttpParameters(params, additionalParams);

            // step 3 - upload the file
            HttpClientWrapper client = new HttpClientWrapper ();
            HttpResponse httpResponse = client.post (IMGLY_UPLOAD_URL, params, headers);

            // step 4 - check the response
            int statusCode = httpResponse.getStatusCode ();
            if (statusCode != 200)
                throw new TwitterException ("ImgLy image upload returned invalid status
code", httpResponse);

            String response = httpResponse.asString ();

            try
```

```
            {
                JSONObject json = new JSONObject (response);
                if (! json.isNull ("url"))
                    return json.getString ("url");
            }
            catch (JSONException e)
            {
                throw new TwitterException ("Invalid ImgLy response: " + response, e);
            }

            throw new TwitterException ("Unknown ImgLy response", httpResponse);
        }

        private String generateVerifyCredentialsAuthorizationHeader ()
        {
            List<HttpParameter> oauthSignatureParams = auth.generateOAuthSignatureHttpParams
("GET", TWITTER_VERIFY_CREDENTIALS);
            return           "OAuth           realm=\"http://api.twitter.com/\","           +
OAuthAuthorization.encodeParameters (oauthSignatureParams, ",", true);
        }
    }

    public static class TwitgooOAuthUploader extends ImageUpload
    {
        private OAuthAuthorization auth;

        // uses the secure upload URL, not the one specified in the Twitpic FAQ
        private        static        final        String        TWITGOO_UPLOAD_URL        =
"http://twitgoo.com/api/uploadAndPost";
        private       static       final       String       TWITTER_VERIFY_CREDENTIALS       =
"https://api.twitter.com/1/account/verify_credentials.json";

        public TwitgooOAuthUploader(OAuthAuthorization auth)
        {
            this.auth = auth;
        }

        @Override
        public String upload (File image) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image)
                    });
        }

        @Override
        public String upload (File image, String message) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image),
                    new HttpParameter ("message", message)
                    });
        }

        @Override
        public   String   upload   (String   imageFileName,   InputStream   imageBody)   throws
TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    });
        }
```

```
        @Override
        public String upload (String imageFileName, InputStream imageBody, String message)
throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    new HttpParameter ("message", message)
                    });
        }

        private String upload (HttpParameter[] additionalParams) throws TwitterException
        {
            // step 1 - generate HTTP request headers
            String                  verifyCredentialsAuthorizationHeader                  =
generateVerifyCredentialsAuthorizationHeader ();

            Map<String, String> headers = new HashMap<String, String>();
            headers.put ("X-Auth-Service-Provider", TWITTER_VERIFY_CREDENTIALS);
            headers.put                                ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

            // step 2 - generate HTTP parameters
            HttpParameter[] params = {
                      new HttpParameter("no_twitter_post", "1")
                    };
            params = appendHttpParameters(params, additionalParams);

            // step 3 - upload the file
            HttpClientWrapper client = new HttpClientWrapper ();
            HttpResponse httpResponse = client.post (TWITGOO_UPLOAD_URL, params, headers);

            // step 4 - check the response
            int statusCode = httpResponse.getStatusCode ();
            if (statusCode != 200)
                throw new TwitterException ("Twitgoo image upload returned invalid status
code", httpResponse);

            String response = httpResponse.asString ();
            if(response.contains("<rsp status=\"ok\">")){
               String h = "<mediaurl>";
               int i = response.indexOf(h);
               if(i != -1){
                       int j = response.indexOf("</mediaurl>", i + h.length());
                       if(j != -1){
                               return response.substring(i + h.length(), j);
                       }
               }
            } else if(response.contains("<rsp status=\"fail\">")){
               String h = "msg=\"";
               int i = response.indexOf(h);
               if(i != -1){
                       int j = response.indexOf("\"", i + h.length());
                       if(j != -1){
                               String msg = response.substring(i + h.length(), j);
                                throw new TwitterException ("Invalid Twitgoo response: " +
msg);
                       }
               }
            }

            throw new TwitterException ("Unknown Twitgoo response", httpResponse);
```

```
        }

        private String generateVerifyCredentialsAuthorizationHeader ()
        {
            List<HttpParameter> oauthSignatureParams = auth.generateOAuthSignatureHttpParams
("GET", TWITTER_VERIFY_CREDENTIALS);
            return           "OAuth           realm=\"http://api.twitter.com/\","          +
OAuthAuthorization.encodeParameters (oauthSignatureParams, ",", true);
        }
    }

    private static HttpParameter[] appendHttpParameters(HttpParameter[] src, HttpParameter[]
dst){
        int srcLen = src.length;
        int dstLen = dst.length;
        HttpParameter[] ret = new HttpParameter[srcLen + dstLen];
        for(int i = 0; i < srcLen; i++){
               ret[i] = src[i];
        }
        for(int i = 0; i < dstLen; i++){
               ret[srcLen + i] = dst[i];
        }
        return ret;
=======

    private static HttpParameter[] appendHttpParameters(HttpParameter[] src, HttpParameter[]
dst) {
        int srcLen = src.length;
        int dstLen = dst.length;
        HttpParameter[] ret = new HttpParameter[srcLen + dstLen];
        for (int i = 0; i < srcLen; i++) {
            ret[i] = src[i];
        }
        for (int i = 0; i < dstLen; i++) {
            ret[srcLen + i] = dst[i];
        }
        return ret;
>>>>>>> f0299deabfe7fc79484aceb357a18a5a00e2e918
    }
}
```

```
    }

    // Described at http://dev.twitpic.com/docs/2/upload/
    public static class ImgLyOAuthUploader extends ImageUpload
    {
        private OAuthAuthorization auth;

        // uses the secure upload URL, not the one specified in the Twitpic FAQ
        private static final String IMGLY_UPLOAD_URL = "http://img.ly/api/2/upload.json";
        private      static      final      String      TWITTER_VERIFY_CREDENTIALS      =
"https://api.twitter.com/1/account/verify_credentials.json";

        public ImgLyOAuthUploader (OAuthAuthorization auth)
        {
            this.auth = auth;
        }

        @Override
        public String upload (File image) throws TwitterException
        {
```

```
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image)
                    });
        }

        @Override
        public String upload (File image, String message) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image),
                    new HttpParameter ("message", message)
                    });
        }

        @Override
        public String upload (String imageFileName, InputStream imageBody) throws
TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    });
        }

        @Override
        public String upload (String imageFileName, InputStream imageBody, String message)
throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    new HttpParameter ("message", message)
                    });
        }

        private String upload (HttpParameter[] additionalParams) throws TwitterException
        {
            // step 1 - generate HTTP request headers
            String                  verifyCredentialsAuthorizationHeader                  =
generateVerifyCredentialsAuthorizationHeader ();

            Map<String, String> headers = new HashMap<String, String>();
            headers.put ("X-Auth-Service-Provider", TWITTER_VERIFY_CREDENTIALS);
            headers.put                              ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

            // step 2 - generate HTTP parameters
            HttpParameter[] params = {
                    };
            params = appendHttpParameters(params, additionalParams);

            // step 3 - upload the file
            HttpClientWrapper client = new HttpClientWrapper ();
            HttpResponse httpResponse = client.post (IMGLY_UPLOAD_URL, params, headers);

            // step 4 - check the response
            int statusCode = httpResponse.getStatusCode ();
            if (statusCode != 200)
                throw new TwitterException ("ImgLy image upload returned invalid status
code", httpResponse);

            String response = httpResponse.asString ();

            try
```

```
            {
                JSONObject json = new JSONObject (response);
                if (! json.isNull ("url"))
                    return json.getString ("url");
            }
            catch (JSONException e)
            {
                throw new TwitterException ("Invalid ImgLy response: " + response, e);
            }

            throw new TwitterException ("Unknown ImgLy response", httpResponse);
        }

        private String generateVerifyCredentialsAuthorizationHeader ()
        {
            List<HttpParameter> oauthSignatureParams = auth.generateOAuthSignatureHttpParams
("GET", TWITTER_VERIFY_CREDENTIALS);
            return           "OAuth           realm=\"http://api.twitter.com/\","           +
OAuthAuthorization.encodeParameters (oauthSignatureParams, ",", true);
        }
    }

    public static class TwitgooOAuthUploader extends ImageUpload
    {
        private OAuthAuthorization auth;

        // uses the secure upload URL, not the one specified in the Twitpic FAQ
        private        static        final        String        TWITGOO_UPLOAD_URL        =
"http://twitgoo.com/api/uploadAndPost";
        private       static       final       String       TWITTER_VERIFY_CREDENTIALS       =
"https://api.twitter.com/1/account/verify_credentials.json";

        public TwitgooOAuthUploader(OAuthAuthorization auth)
        {
            this.auth = auth;
        }

        @Override
        public String upload (File image) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image)
                    });
        }

        @Override
        public String upload (File image, String message) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image),
                    new HttpParameter ("message", message)
                    });
        }

        @Override
        public   String   upload   (String   imageFileName,   InputStream   imageBody)   throws
TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    });
        }
```

```
        @Override
        public String upload (String imageFileName, InputStream imageBody, String message)
throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    new HttpParameter ("message", message)
                    });
        }

        private String upload (HttpParameter[] additionalParams) throws TwitterException
        {
            // step 1 - generate HTTP request headers
            String                  verifyCredentialsAuthorizationHeader                  =
generateVerifyCredentialsAuthorizationHeader ();

            Map<String, String> headers = new HashMap<String, String>();
            headers.put ("X-Auth-Service-Provider", TWITTER_VERIFY_CREDENTIALS);
            headers.put                                ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

            // step 2 - generate HTTP parameters
            HttpParameter[] params = {
                      new HttpParameter("no_twitter_post", "1")
                    };
            params = appendHttpParameters(params, additionalParams);

            // step 3 - upload the file
            HttpClientWrapper client = new HttpClientWrapper ();
            HttpResponse httpResponse = client.post (TWITGOO_UPLOAD_URL, params, headers);

            // step 4 - check the response
            int statusCode = httpResponse.getStatusCode ();
            if (statusCode != 200)
                throw new TwitterException ("Twitgoo image upload returned invalid status
code", httpResponse);

            String response = httpResponse.asString ();
            if(response.contains("<rsp status=\"ok\">")){
               String h = "<mediaurl>";
               int i = response.indexOf(h);
               if(i != -1){
                       int j = response.indexOf("</mediaurl>", i + h.length());
                       if(j != -1){
                               return response.substring(i + h.length(), j);
                       }
               }
            } else if(response.contains("<rsp status=\"fail\">")){
               String h = "msg=\"";
               int i = response.indexOf(h);
               if(i != -1){
                       int j = response.indexOf("\"", i + h.length());
                       if(j != -1){
                               String msg = response.substring(i + h.length(), j);
                                throw new TwitterException ("Invalid Twitgoo response: " +
msg);
                       }
               }
            }

            throw new TwitterException ("Unknown Twitgoo response", httpResponse);
```

```
        }

        private String generateVerifyCredentialsAuthorizationHeader ()
        {
            List<HttpParameter> oauthSignatureParams = auth.generateOAuthSignatureHttpParams
("GET", TWITTER_VERIFY_CREDENTIALS);
            return           "OAuth           realm=\"http://api.twitter.com/\","          +
OAuthAuthorization.encodeParameters (oauthSignatureParams, ",", true);
        }
    }

    private static HttpParameter[] appendHttpParameters(HttpParameter[] src, HttpParameter[]
dst) {
        int srcLen = src.length;
        int dstLen = dst.length;
        HttpParameter[] ret = new HttpParameter[srcLen + dstLen];
        for (int i = 0; i < srcLen; i++) {
            ret[i] = src[i];
        }
        for (int i = 0; i < dstLen; i++) {
            ret[srcLen + i] = dst[i];
        }
        return ret;
    }
}
```

Version: 83e861d18761b87b91611dd27a7417620fefdbdb

Parents:

d2ffce58eec309979a5286d29990e468b0e04942

830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd

Merge base:

358c31c7c270febe0ffbdffb318015ccae1038fe

twitter4j/twitter4j-core/src/main/java/twitter4j/util/ImageUpload.java

Chunk 59: (concatenation/Import declaration)

```
import java.io.InputStream;
<<<<<<< HEAD
=======
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd
import java.util.HashMap;
```

```
import java.io.InputStream;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.HashMap;
```

Chunk 60:(versio 2/method interface)

```
    public abstract String upload (File image) throws TwitterException;
<<<<<<< HEAD
    public abstract String upload (String imageFileName, InputStream imageBody) throws
TwitterException;
=======
    public abstract String upload (File image, String message) throws TwitterException;
    public abstract String upload (String imageFileName, InputStream imageBody) throws
TwitterException;
    public abstract String upload (String imageFileName, InputStream imageBody, String
message) throws TwitterException;
>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd


    /** Returns an image uploader to Twitpic. Handles both BasicAuth and OAuth.
```

```
    public abstract String upload(File image) throws TwitterException;

    public abstract String upload(File image, String message) throws TwitterException;

    public abstract String upload(String imageFileName, InputStream imageBody) throws
TwitterException;

    public abstract String upload(String imageFileName, InputStream imageBody, String
message) throws TwitterException;


    /**
     * Returns an image uploader to Twitpic. Handles both BasicAuth and OAuth.
```

Chunk 61: (versio 2/ method invocation, return statement, variable)

```
                    new HttpParameter("verify_url", signedVerifyCredentialsURL),
<<<<<<< HEAD
                    new HttpParameter("media", image)
                    };
            return upload(params);
        }
=======
                    };
            params = appendHttpParameters(params, additionalParams);
>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd
```

```
                    new HttpParameter("verify_url", signedVerifyCredentialsURL),
            };
            params = appendHttpParameters(params, additionalParams);
```

Chunk 62: (new code/ annotation, method declaration, method invocation, variable)

```
                    };
<<<<<<< HEAD
            return upload(params);
        }

        @Override
        public String upload(String imageFileName, InputStream imageBody) throws
TwitterException {
            // step 1 - generate HTTP parameters
            HttpParameter[] params =
                    {
                            new HttpParameter("username", auth.getUserId()),
                            new HttpParameter("password", auth.getPassword()),
                            new HttpParameter("media", imageFileName, imageBody)
                    };
            return upload(params);
        }
=======
            params = appendHttpParameters(params, additionalParams);
>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd

        private String upload(HttpParameter[] params) throws TwitterException {
```

```
            };
            params = appendHttpParameters(params, additionalParams);
```

Chunk 63: (version 2/annotation, commentary, method declaration, method invocation, return statement, variable)

```
        public String upload (File image) throws TwitterException
        {
<<<<<<< HEAD
            // step 2 - generate HTTP parameters
            HttpParameter[] params =
            {
                new HttpParameter ("key", twitpicAPIKey),
                new HttpParameter ("media", image)
            };

            return upload(params);
```

```
=======
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image)
                    });
        }

        @Override
        public String upload (File image, String message) throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", image),
                    new HttpParameter ("message", message),
                    });
>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd
        }
```

```
        public String upload(File image) throws TwitterException {
            return upload(new HttpParameter[]{
                    new HttpParameter("media", image)
            });
        }

        @Override
        public String upload(File image, String message) throws TwitterException {
            return upload(new HttpParameter[]{
                    new HttpParameter("media", image),
                    new HttpParameter("message", message),
            });
        }
```

Chunk 64: (version 2/ annotation, commentary, method declaration, method invocation, method signature, return statement, variable)

```
        {
<<<<<<< HEAD
            // step 2 - generate HTTP parameters
            HttpParameter[] params =
            {
                new HttpParameter ("key", twitpicAPIKey),
                new HttpParameter ("media", imageFileName, imageBody)
            };

            return upload(params);
        }

        private String upload (HttpParameter[] params) throws TwitterException
=======
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody)
                    });
        }

        @Override
        public String upload (String imageFileName, InputStream imageBody, String message)
throws TwitterException
        {
            return upload(new HttpParameter[]{
                    new HttpParameter ("media", imageFileName, imageBody),
                    new HttpParameter ("message", message)
                    });
        }
```

```
        private String upload (HttpParameter[] additionalParams) throws TwitterException
>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd
        {
```

```
        public   String   upload(String   imageFileName,   InputStream   imageBody)   throws
TwitterException {
            return upload(new HttpParameter[]{
                    new HttpParameter("media", imageFileName, imageBody)
            });
        }

        @Override
        public String upload(String imageFileName, InputStream imageBody, String message)
throws TwitterException {
            return upload(new HttpParameter[]{
                    new HttpParameter("media", imageFileName, imageBody),
                    new HttpParameter("message", message)
            });
        }

        private String upload(HttpParameter[] additionalParams) throws TwitterException {
```

Chunk 65: (version 2/ commentary, method invocation, variable)

```
            headers.put                                ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

<<<<<<< HEAD

=======
            // step 2 - generate HTTP parameters
            HttpParameter[] params = {
                new HttpParameter ("key", twitpicAPIKey),
            };
            params = appendHttpParameters(params, additionalParams);

>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd
            // step 3 - upload the file
```

```
            headers.put("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

            // step 2 - generate HTTP parameters
            HttpParameter[] params = {
                    new HttpParameter("key", twitpicAPIKey),
            };
            params = appendHttpParameters(params, additionalParams);

            // step 3 - upload the file
```

Chunk 66: (new code/ method declaration, method invocation, return statement, variable)

```
                    };
<<<<<<< HEAD

            return upload(params);
        }

        @Override
```

```
        public String upload(String imageFileName, InputStream imageBody) throws
TwitterException {
            // step 1 - generate HTTP parameters
            HttpParameter[] params =
                    {
                            new HttpParameter("username", auth.getUserId()),
                            new HttpParameter("password", auth.getPassword()),
                            new HttpParameter("media", imageFileName, imageBody)
                    };

            return upload(params);
        }
=======
            params = appendHttpParameters(params, additionalParams);
>>>>>>> 830fc66c54c1fe93a8f6caed971eb3a3f9c0eefd

        private String upload(HttpParameter[] params) throws TwitterException {
            // step 2 - upload the file
```

```
            };
            params = appendHttpParameters(params, additionalParams);

            // step 2 - upload the file
```

Version: e05d1dc0d749022cd477186e9bb7408857134de7

Parents:

48843586c13e8df941d61737bee287b6041be0cf

2c842822a4eab2b25c66c56dd1a39ccf0659e7a9

Merge base:

6283e3c4c631cf46e08de2e8394850e54191266e

twitter4j/twitter4j-core/src/main/java/twitter4j/StatusStreamImpl.java

Chunk 67: (version 1/if statement)

```
                        listener.onStatus(new StatusJSONImpl(json));
<<<<<<< HEAD
                    }
                    else if (!json.isNull ("direct_message")) {
                        userStreamListener.onDirectMessage     (new     DirectMessageJSONImpl
(json.getJSONObject ("direct_message")));
                    }
                    else if (!json.isNull("delete")) {
=======
                    } else if (!json.isNull("delete")) {
>>>>>>> 2c842822a4eab2b25c66c56dd1a39ccf0659e7a9
                        listener.onDeletionNotice(new StatusDeletionNoticeImpl(json));
```

```
                        listener.onStatus(new StatusJSONImpl(json));
                    } else if (!json.isNull("direct_message")) {
                        userStreamListener.onDirectMessage(new
DirectMessageJSONImpl(json.getJSONObject("direct_message")));
                    } else if (!json.isNull("delete")) {
                        listener.onDeletionNotice(new StatusDeletionNoticeImpl(json));
```

Chunk 68: (version 1/if statement)

```
                            userStreamListener.onUnfollow(source, target);
<<<<<<< HEAD
                        } else if (event.startsWith ("list_"))
                         {
                            UserList      targetObject      =      new      UserListJSONImpl
(json.getJSONObject ("target_object"));

                            if ("list_user_subscribed".equals (event)) {
                                userStreamListener.onUserSubscribedToList  (source,  target,
targetObject);
                            } else if ("list_created".equals (event)) {
                                userStreamListener.onUserCreatedList               (source,
targetObject);
                            } else if ("list_updated".equals (event)) {
                                userStreamListener.onUserUpdatedList               (source,
targetObject);
                            } else if ("list_destroyed".equals (event)) {
                                userStreamListener.onUserDestroyedList             (source,
targetObject);
                            }
                        } else if ("block".equals(event)) {
                          userStreamListener.onBlock (source, target);
                        } else if ("unblock".equals (event)) {
                            userStreamListener.onUnblock (source, target);
```

```
                        } else {
                            Status targetObject = new StatusJSONImpl (json.getJSONObject
("target_object"));

                            if ("favorite".equals (event)) {
                                userStreamListener.onFavorite       (source,       target,
targetObject);
                            } else if ("unfavorite".equals (event)) {
                                userStreamListener.onUnfavorite      (source,      target,
targetObject);
                            } else if ("retweet".equals (event)) {
                                // note: retweet events also show up as statuses
                                userStreamListener.onRetweet        (source,        target,
targetObject);
                            } else {
                                // tmp: just checking what kind of unknown social event
we're receiving on this stream
                                logger.info("Received unknown social event type '" + event
+ "': " + line);
                            }
                         }
                     } else {
=======
                       } else if ("block".equals(event)) {
                         userStreamListener.onBlock(source, target);
                       } else if ("unblock".equals(event)) {
                         userStreamListener.onUnblock(source, target);
                       }
                     } else {
>>>>>>> 2c842822a4eab2b25c66c56dd1a39ccf0659e7a9
                        // tmp: just checking what kind of unknown event we're receiving on
this stream
```

```
                           userStreamListener.onUnfollow(source, target);
                       } else if (event.startsWith("list_")) {
                           UserList targetObject = new UserListJSONImpl (json.getJSONObject
("target_object"));

                            if ("list_user_subscribed".equals (event)) {
                                userStreamListener.onUserSubscribedToList  (source,  target,
targetObject);
                            } else if ("list_created".equals (event)) {
                                userStreamListener.onUserCreatedList               (source,
targetObject);
                            } else if ("list_updated".equals (event)) {
                                userStreamListener.onUserUpdatedList               (source,
targetObject);
                            } else if ("list_destroyed".equals (event)) {
                                userStreamListener.onUserDestroyedList             (source,
targetObject);
                            }
                        } else if ("block".equals(event)) {
                          userStreamListener.onBlock (source, target);
                        } else if ("unblock".equals (event)) {
                            userStreamListener.onUnblock (source, target);
                        } else {
                            Status targetObject = new StatusJSONImpl (json.getJSONObject
("target_object"));

                            if ("favorite".equals (event)) {
```

```
                                userStreamListener.onFavorite       (source,       target,
targetObject);
                            } else if ("unfavorite".equals (event)) {
                                userStreamListener.onUnfavorite      (source,      target,
targetObject);
                            } else if ("retweet".equals (event)) {
                                // note: retweet events also show up as statuses
                                userStreamListener.onRetweet        (source,        target,
targetObject);
                            } else {
                                // tmp: just checking what kind of unknown social event
we're receiving on this stream
                                logger.info("Received unknown social event type '" + event
+ "': " + line);
                            }
                        }
                    } else {
                       // tmp: just checking what kind of unknown event we're receiving on
this stream
```

twitter4j/twitter4j-core/src/main/java/twitter4j/UserStreamAdapter.java

Chunk 69: (concatenation/ method declaration)

```
    }

<<<<<<< HEAD
    public void onUserSubscribedToList(User subscriber, User listOwner, UserList list) {
    }

    public void onUserCreatedList(User listOwner, UserList list) {
    }

    public void onUserUpdatedList(User listOwner, UserList list) {
    }

    public void onUserDestroyedList(User listOwner, UserList list) {
    }

=======
>>>>>>> 2c842822a4eab2b25c66c56dd1a39ccf0659e7a9
    public void onBlock(User source, User target) {
```

```
    }

    public void onUserSubscribedToList(User subscriber, User listOwner, UserList list) {
    }

    public void onUserCreatedList(User listOwner, UserList list) {
    }

    public void onUserUpdatedList(User listOwner, UserList list) {
    }

    public void onUserDestroyedList(User listOwner, UserList list) {
    }

    public void onBlock(User source, User target) {
```

twitter4j/twitter4j-core/src/main/java/twitter4j/UserStreamListener.java

Chunk 70: (concatenation/ commentary, method interface)

```
    /**
<<<<<<< HEAD
     * @param subscriber
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserSubscribedToList(User subscriber, User listOwner, UserList list);

    /**
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserCreatedList(User listOwner, UserList list);

    /**
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserUpdatedList(User listOwner, UserList list);

    /**
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserDestroyedList(User listOwner, UserList list);

    /**
=======
>>>>>>> 2c842822a4eab2b25c66c56dd1a39ccf0659e7a9
     * @param source
```

```
    /**
     * @param subscriber
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserSubscribedToList(User subscriber, User listOwner, UserList list);

    /**
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserCreatedList(User listOwner, UserList list);

    /**
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserUpdatedList(User listOwner, UserList list);
```

```
    /**
     * @param listOwner
     * @param list
     * @since Twitter4J 2.1.3
     */
    void onUserDestroyedList(User listOwner, UserList list);

    /**
     * @param source
```

Version: d4488a43598615aab876c23cc9833ed2a0a1cb95

Parents:

4a57ec5718bf9b866629c6761deb9cef8171eaf7

3f7905002910c3d277810bdfd4331d1ddde05de4

Merge base:

f1d385b080ef6a2740757d7bc506848e97d90cee

twitter4j/twitter4j-core/src/main/java/twitter4j/StatusAdapter.java

Chunk 71: (version1/method declaration)

```
    }

<<<<<<< HEAD

    public void onStatus(Status status) {
    }

    public void onTrackLimitationNotice(int numberOfLimitedStatuses) {
    }

=======
    @Override
    public void onFriendList (int [] friendIds)
    {
    }

    @Override
    public void onStatus (Status status)
    {
    }

    @Override
    public void onTrackLimitationNotice (int numberOfLimitedStatuses)
    {
    }

    @Override
    public void onFavorite (User source, User target, Status targetObject)
    {
    }

    @Override
    public void onFollow (User source, User target)
    {
    }

    @Override
    public void onUnfavorite (User source, User target, Status targetObject)
    {
    }

    @Override
    public void onRetweet (User source, User target, Status targetObject)
    {
    }

    @Override
```

```
    public void onUnfollow (User source, User target)
    {
    }

    @Override
    public void onDirectMessage (DirectMessage directMessage)
    {
    }

    @Override
    public void onUserSubscribedToList (User subscriber, User listOwner, UserList list)
    {
    }

    @Override
    public void onUserCreatedList (User listOwner, UserList list)
    {
    }

    @Override
    public void onUserDestroyedList (User listOwner, UserList list)
    {
    }

    @Override
    public void onUserUpdatedList (User listOwner, UserList list)
    {
    }

    @Override
    public void onBlock (User source, User target)
    {
    }

    @Override
    public void onUnblock (User source, User target)
    {
    }
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
}
```

```
    }

    public void onStatus(Status status) {
    }

    public void onTrackLimitationNotice(int numberOfLimitedStatuses) {
    }

}
```

twitter4j/twitter4j-core/src/main/java/twitter4j/StatusListener.java

Chunk 72: (version 1/method interface)

```
    void onException(Exception ex);
<<<<<<< HEAD
=======

    void onFriendList (int [] friendIds);
```

```
    void onFavorite (User source, User target, Status favoritedStatus);
    void onUnfavorite (User source, User target, Status unfavoritedStatus);

    void onFollow (User source, User target);
    void onUnfollow (User source, User target);

    void onRetweet (User source, User target, Status retweetedStatus);

    void onDirectMessage (DirectMessage directMessage);

    void onUserSubscribedToList (User subscriber, User listOwner, UserList list);
    void onUserCreatedList (User listOwner, UserList list);
    void onUserUpdatedList (User listOwner, UserList list);
    void onUserDestroyedList (User listOwner, UserList list);

    void onBlock (User source, User target);
    void onUnblock (User source, User target);
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
}
```

```
    void onException(Exception ex);
}
```

twitter4j/twitter4j-core/src/main/java/twitter4j/StatusStreamImpl.java

Chunk 73: (version 1/ if statement)

```
                    JSONObject json = new JSONObject(line);
<<<<<<< HEAD
                    if (!json.isNull ("sender")) {
                        userStreamListener.onDirectMessage     (new     DirectMessageJSONImpl
(json));
                    } else if (!json.isNull("text")) {
=======
                    if (!json.isNull("text")) {
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
                        listener.onStatus(new StatusJSONImpl(json));
```

```
                    JSONObject json = new JSONObject(line);
                    if (!json.isNull ("sender")) {
                        userStreamListener.onDirectMessage     (new     DirectMessageJSONImpl
(json));
                    } else if (!json.isNull("text")) {
                        listener.onStatus(new StatusJSONImpl(json));
```

Chunk 74: (new code/ if statement)

```
                        listener.onTrackLimitationNotice(ParseUtil.getInt("track",
json.getJSONObject("limit")));
<<<<<<< HEAD
                    } else if (!json.isNull("friends")) {
                        JSONArray friends = json.getJSONArray("friends");
                        int[] friendIds = new int[friends.length()];
                        for (int i = 0; i < friendIds.length; ++i)
                            friendIds[i] = friends.getInt(i);
                        userStreamListener.onFriendList(friendIds);
                    } else if (!json.isNull("event")) {
                        String event = json.getString("event");
                        User source = new UserJSONImpl(json.getJSONObject("source"));
                        User target = new UserJSONImpl(json.getJSONObject("target"));
```

```
                        if ("favorite".equals(event)) {
                            Status             targetObject             =             new
StatusJSONImpl(json.getJSONObject("target_object"));
                            userStreamListener.onFavorite(source, target, targetObject);
                        } else if ("unfavorite".equals(event)) {
                            Status             targetObject             =             new
StatusJSONImpl(json.getJSONObject("target_object"));
                            userStreamListener.onUnfavorite(source, target, targetObject);
                        } else if ("retweet".equals(event)) {
                            // note: retweet events also show up as statuses
                            Status             targetObject             =             new
StatusJSONImpl(json.getJSONObject("target_object"));
                            userStreamListener.onRetweet(source, target, targetObject);
                        } else if ("follow".equals(event)) {
                            userStreamListener.onFollow(source, target);
                        } else if ("unfollow".equals(event)) {
                            userStreamListener.onUnfollow(source, target);
                        }
                    } else {
=======
                    }
                    else if (!json.isNull ("scrub_geo")) {
                        // Not implemented yet
                        System.out.println ("Geo-tagging deletion notice (not implemented
yet): " + line);
                    }
                    else if (!json.isNull ("friends")) {
                        JSONArray friends = json.getJSONArray ("friends");
                        int [] friendIds = new int [friends.length ()];
                        for (int i = 0; i < friendIds.length; ++i)
                            friendIds[i] = friends.getInt (i);

                        listener.onFriendList (friendIds);
                    }
                    else if (!json.isNull ("event")) {
                        String event = json.getString ("event");
                        User source = new UserJSONImpl (json.getJSONObject ("source"));
                        User target = new UserJSONImpl (json.getJSONObject ("target"));

                        if ("follow".equals (event) || "unfollow".equals (event))
                        {
                            listener.onFollow (source, target);
                        }
                        else if ("unfollow".equals (event))
                        {
                            listener.onUnfollow (source, target);
                        }
                        else if (event.startsWith ("list_"))
                        {
                            UserList targetObject = new UserListJSONImpl (json.getJSONObject
("target_object"));

                            if ("list_user_subscribed".equals (event)) {
                                listener.onUserSubscribedToList       (source,       target,
targetObject);
                            } else if ("list_created".equals (event)) {
                                listener.onUserCreatedList (source, targetObject);
                            } else if ("list_updated".equals (event)) {
                                listener.onUserUpdatedList (source, targetObject);
                            } else if ("list_destroyed".equals (event)) {
                                listener.onUserDestroyedList (source, targetObject);
                            }
```

```
                        }
                        else if ("block".equals (event))
                        {
                            listener.onBlock (source, target);
                        }
                        else if ("unblock".equals (event))
                        {
                            listener.onUnblock (source, target);
                        }
                        else
                        {
                            Status targetObject = new StatusJSONImpl (json.getJSONObject
("target_object"));

                            if ("favorite".equals (event)) {
                                listener.onFavorite (source, target, targetObject);
                            } else if ("unfavorite".equals (event)) {
                                listener.onUnfavorite (source, target, targetObject);
                            } else if ("retweet".equals (event)) {
                                // note: retweet events also show up as statuses
                                listener.onRetweet (source, target, targetObject);
                            } else {
                                // tmp: just checking what kind of unknown social event
we're receiving on this stream
                                System.out.println ("Received unknown social event type '" +
event + "': " + line);
                            }
                        }
                    }
                    else
                    {
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
                        // tmp: just checking what kind of unknown event we're receiving on
this stream
```

```
                        listener.onTrackLimitationNotice(ParseUtil.getInt("track",
json.getJSONObject("limit")));
                    } else if (!json.isNull ("scrub_geo")) {
                            // Not implemented yet
                            System.out.println    ("Geo-tagging    deletion    notice    (not
implemented yet): " + line);
                    } else if (!json.isNull("friends")) {
                        JSONArray friends = json.getJSONArray("friends");
                        int[] friendIds = new int[friends.length()];
                        for (int i = 0; i < friendIds.length; ++i)
                            friendIds[i] = friends.getInt(i);
                        userStreamListener.onFriendList(friendIds);
                    } else if (!json.isNull("event")) {
                        String event = json.getString("event");
                        User source = new UserJSONImpl(json.getJSONObject("source"));
                        User target = new UserJSONImpl(json.getJSONObject("target"));

                        if ("favorite".equals(event)) {
                            Status              targetObject              =              new
StatusJSONImpl(json.getJSONObject("target_object"));
                            userStreamListener.onFavorite(source, target, targetObject);
                        } else if ("unfavorite".equals(event)) {
                            Status              targetObject              =              new
StatusJSONImpl(json.getJSONObject("target_object"));
                            userStreamListener.onUnfavorite(source, target, targetObject);
                        } else if ("retweet".equals(event)) {
```

```
                            // note: retweet events also show up as statuses
                            Status              targetObject              =              new
StatusJSONImpl(json.getJSONObject("target_object"));
                            userStreamListener.onRetweet(source, target, targetObject);
                        } else if ("follow".equals(event)) {
                            userStreamListener.onFollow(source, target);
                        } else if ("unfollow".equals(event)) {
                            userStreamListener.onUnfollow(source, target);
                        } else if (event.startsWith ("list_"))
                         {
                             UserList      targetObject      =      new      UserListJSONImpl
(json.getJSONObject ("target_object"));

                             if ("list_user_subscribed".equals (event)) {
                                 userStreamListener.onUserSubscribedToList  (source,  target,
targetObject);
                             } else if ("list_created".equals (event)) {
                                 userStreamListener.onUserCreatedList               (source,
targetObject);
                             } else if ("list_updated".equals (event)) {
                                 userStreamListener.onUserUpdatedList               (source,
targetObject);
                             } else if ("list_destroyed".equals (event)) {
                                 userStreamListener.onUserDestroyedList             (source,
targetObject);
                             }
                         } else if ("block".equals(event)) {
                           userStreamListener.onBlock (source, target);
                         } else if ("unblock".equals (event)) {
                             userStreamListener.onUnblock (source, target);
                         } else {
                             Status  targetObject  =  new  StatusJSONImpl  (json.getJSONObject
("target_object"));

                             if ("favorite".equals (event)) {
                                 userStreamListener.onFavorite        (source,        target,
targetObject);
                             } else if ("unfavorite".equals (event)) {
                                 userStreamListener.onUnfavorite       (source,       target,
targetObject);
                             } else if ("retweet".equals (event)) {
                                 // note: retweet events also show up as statuses
                                 userStreamListener.onRetweet        (source,        target,
targetObject);
                             } else {
                                 // tmp: just checking what kind of unknown social event
we're receiving on this stream
                                 logger.info("Received unknown social event type '" + event
+ "': " + line);
                             }
                         }
                     } else {
                        // tmp: just checking what kind of unknown event we're receiving on
this stream
```

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterStream.java

Chunk 75: (version 2/commentary, method invocation, method signature)

```
    }

<<<<<<< HEAD
```

```
    public void user() {
        ensureBasicEnabled(); // for now, the user stream will switch to OAuth at some point
in the future
        startHandler(new StreamHandlingThread(true) {
            public UserStream getStream() throws TwitterException {
=======
    public void user () {
        ensureAuthorizationEnabled ();
        startHandler(new StreamHandlingThread() {
            public StatusStream getStream() throws TwitterException {
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
                return getUserStream();
```

```
    }

    public void user() {
        ensureAuthorizationEnabled ();
        startHandler(new StreamHandlingThread(true) {
            public UserStream getStream() throws TwitterException {
                return getUserStream();
```

Chunk 76: (version 1/if statement, method invocation, method signature)

```
    }

<<<<<<< HEAD
    public UserStream getUserStream() throws TwitterException {
        ensureBasicEnabled();
        if (!(statusListener instanceof UserStreamListener)) {
            logger.warn("Use of UserStreamListener is suggested.");
        }
=======
    public StatusStream getUserStream() throws TwitterException {
        ensureAuthorizationEnabled ();
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
        try {
```

```
    }

    public UserStream getUserStream() throws TwitterException {
                ensureAuthorizationEnabled ();
        if (!(statusListener instanceof UserStreamListener)) {
            logger.warn("Use of UserStreamListener is suggested.");
        }
        try {
```

twitter4j/twitter4j-core/src/main/java/twitter4j/util/ImageUpload.java

Chunk 77: (version 2/ class signature, commentary, method declaration, variable)

```
import twitter4j.internal.org.json.JSONObject;

<<<<<<< HEAD
public abstract class ImageUpload {
    public abstract String upload(File image) throws TwitterException;

    /**
     * Returns an image uploader to Twitpic. Only handles BasicAuth right now
     */
    public static ImageUpload getTwitpicUploader(Twitter twitter) throws TwitterException {
        return getTwitpicUploader(twitter.getAuthorization());
```

```
    }

    /**
     * Returns an image uploader to Twitpic. Only handles BasicAuth right now
     */
    public static ImageUpload getTwitpicUploader(Authorization auth) {
        ensureBasicEnabled(auth);
        return getTwitpicUploader((BasicAuthorization) auth);
=======
public abstract class ImageUpload
{
    public static String DEFAULT_TWITPIC_API_KEY = null;

    public abstract String upload (File image) throws TwitterException;


    /** Returns an image uploader to Twitpic. Handles both BasicAuth and OAuth.
     *  Note: When using OAuth, the Twitpic API Key needs to be specified, either with the
field ImageUpload.DEFAULT_TWITPIC_API_KEY,
     *    or using the getTwitpicUploader (String twitpicAPIKey, OAuthAuthorization auth)
method
     */
    public static ImageUpload getTwitpicUploader (Twitter twitter) throws TwitterException
    {
        Authorization auth = twitter.getAuthorization ();
        if (auth instanceof OAuthAuthorization)
            return getTwitpicUploader (DEFAULT_TWITPIC_API_KEY, (OAuthAuthorization) auth);

        ensureBasicEnabled (auth);
        return getTwitpicUploader ((BasicAuthorization) auth);
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
    }
```

```
import twitter4j.internal.org.json.JSONObject;

public abstract class ImageUpload
{
    public static String DEFAULT_TWITPIC_API_KEY = null;

    public abstract String upload (File image) throws TwitterException;


    /** Returns an image uploader to Twitpic. Handles both BasicAuth and OAuth.
     *  Note: When using OAuth, the Twitpic API Key needs to be specified, either with the
field ImageUpload.DEFAULT_TWITPIC_API_KEY,
     *    or using the getTwitpicUploader (String twitpicAPIKey, OAuthAuthorization auth)
method
     */
    public static ImageUpload getTwitpicUploader (Twitter twitter) throws TwitterException
    {
        Authorization auth = twitter.getAuthorization ();
        if (auth instanceof OAuthAuthorization)
            return getTwitpicUploader (DEFAULT_TWITPIC_API_KEY, (OAuthAuthorization) auth);

        ensureBasicEnabled (auth);
        return getTwitpicUploader ((BasicAuthorization) auth);
    }
```

Chunk 78: (Combination/ commentary, method declaration)

```
    }
```

```
<<<<<<< HEAD


    /**
     * Returns an image uploader to YFrog. Handles both BasicAuth and OAuth
     */
    public static ImageUpload getYFrogUploader(Twitter twitter) throws TwitterException {
        Authorization auth = twitter.getAuthorization();
        if (auth instanceof OAuthAuthorization)
            return      getYFrogUploader(twitter.getScreenName(),       (OAuthAuthorization)
twitter.getAuthorization());

        ensureBasicEnabled(auth);
        return getYFrogUploader((BasicAuthorization) auth);
    }

    /**
     * Returns a BasicAuth image uploader to YFrog
     */
    public static ImageUpload getYFrogUploader(String userId, String password) {
        return getYFrogUploader(new BasicAuthorization(userId, password));
    }

    /**
     * Returns a BasicAuth image uploader to YFrog
     */
    public static ImageUpload getYFrogUploader(BasicAuthorization auth) {
        return new YFrogBasicAuthUploader(auth);
=======

    /** Returns an OAuth image uploader to Twitpic */
    public static ImageUpload getTwitpicUploader (String twitpicAPIKey, OAuthAuthorization
auth)
    {
        return new TwitpicOAuthUploader (twitpicAPIKey, auth);
    }


    /** Returns an image uploader to YFrog. Handles both BasicAuth and OAuth */
    public static ImageUpload getYFrogUploader (Twitter twitter) throws TwitterException
    {
        Authorization auth = twitter.getAuthorization ();
        if (auth instanceof OAuthAuthorization)
            return getYFrogUploader (twitter.getScreenName (), (OAuthAuthorization) auth);

        ensureBasicEnabled (auth);
        return getYFrogUploader ((BasicAuthorization) auth);
    }

    /** Returns a BasicAuth image uploader to YFrog */
    public static ImageUpload getYFrogUploader (BasicAuthorization auth)
    {
        return new YFrogBasicAuthUploader (auth);
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
    }
```

```
    }


    /** Returns an image uploader to YFrog. Handles both BasicAuth and OAuth */
    public static ImageUpload getYFrogUploader (Twitter twitter) throws TwitterException
```

```
    {
        Authorization auth = twitter.getAuthorization ();
        if (auth instanceof OAuthAuthorization)
            return getYFrogUploader (twitter.getScreenName (), (OAuthAuthorization) auth);

        ensureBasicEnabled (auth);
        return getYFrogUploader ((BasicAuthorization) auth);
    }

    /** Returns a BasicAuth image uploader to YFrog */
    public static ImageUpload getYFrogUploader (BasicAuthorization auth)
    {
        return new YFrogBasicAuthUploader (auth);
    }
```

Chunk 79: v2/class declaration, class signature, commentary)

```
    }
<<<<<<< HEAD

    private static class TwitpicBasicAuthUploader extends ImageUpload {
=======

    // Described at http://dev.twitpic.com/docs/2/upload/
    private static class TwitpicOAuthUploader extends ImageUpload
    {
        private String twitpicAPIKey;
        private OAuthAuthorization auth;

        // uses the secure upload URL, not the one specified in the Twitpic FAQ
        private        static        final        String        TWITPIC_UPLOAD_URL        =
"https://twitpic.com/api/2/upload.json";
        private      static      final      String      TWITTER_VERIFY_CREDENTIALS      =
"https://api.twitter.com/1/account/verify_credentials.json";

        public TwitpicOAuthUploader (String twitpicAPIKey, OAuthAuthorization auth)
        {
            if (twitpicAPIKey == null || "".equals (twitpicAPIKey))
                throw new IllegalArgumentException ("The Twitpic API Key supplied to the
OAuth image uploader can't be null or empty");

            this.twitpicAPIKey = twitpicAPIKey;
            this.auth = auth;
        }

        @Override
        public String upload (File image) throws TwitterException
        {
            // step 1 - generate HTTP request headers
            String                  verifyCredentialsAuthorizationHeader                =
generateVerifyCredentialsAuthorizationHeader ();

            Map<String, String> headers = new HashMap<String, String> ();
            headers.put ("X-Auth-Service-Provider", TWITTER_VERIFY_CREDENTIALS);
            headers.put                              ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

            // step 2 - generate HTTP parameters
            HttpParameter[] params =
            {
                new HttpParameter ("key", twitpicAPIKey),
                new HttpParameter ("media", image)
```

```
            };

            // step 3 - upload the file
            HttpClientWrapper client = new HttpClientWrapper ();
            HttpResponse httpResponse = client.post (TWITPIC_UPLOAD_URL, params, headers);

            // step 4 - check the response
            int statusCode = httpResponse.getStatusCode ();
            if (statusCode != 200)
                throw new TwitterException ("Twitpic image upload returned invalid status
code", httpResponse);

            String response = httpResponse.asString ();

            try
            {
                JSONObject json = new JSONObject (response);
                if (! json.isNull ("url"))
                    return json.getString ("url");
            }
            catch (JSONException e)
            {
                throw new TwitterException ("Invalid Twitpic response: " + response, e);
            }

            throw new TwitterException ("Unknown Twitpic response", httpResponse);
        }

        private String generateVerifyCredentialsAuthorizationHeader ()
        {
            List<HttpParameter> oauthSignatureParams = auth.generateOAuthSignatureHttpParams
("GET", TWITTER_VERIFY_CREDENTIALS);
            return           "OAuth           realm=\"http://api.twitter.com/\","           +
OAuthAuthorization.encodeParameters (oauthSignatureParams, ",", true);
        }
    }

    private static class TwitpicBasicAuthUploader extends ImageUpload
    {
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
        private BasicAuthorization auth;
```

```
    }

    // Described at http://dev.twitpic.com/docs/2/upload/
    private static class TwitpicOAuthUploader extends ImageUpload
    {
        private String twitpicAPIKey;
        private OAuthAuthorization auth;

        // uses the secure upload URL, not the one specified in the Twitpic FAQ
        private        static        final        String        TWITPIC_UPLOAD_URL        =
"https://twitpic.com/api/2/upload.json";
        private      static      final      String      TWITTER_VERIFY_CREDENTIALS      =
"https://api.twitter.com/1/account/verify_credentials.json";

        public TwitpicOAuthUploader (String twitpicAPIKey, OAuthAuthorization auth)
        {
            if (twitpicAPIKey == null || "".equals (twitpicAPIKey))
                throw new IllegalArgumentException ("The Twitpic API Key supplied to the
OAuth image uploader can't be null or empty");
```

```
            this.twitpicAPIKey = twitpicAPIKey;
            this.auth = auth;
        }

        @Override
        public String upload (File image) throws TwitterException
        {
            // step 1 - generate HTTP request headers
            String                  verifyCredentialsAuthorizationHeader                  =
generateVerifyCredentialsAuthorizationHeader ();

            Map<String, String> headers = new HashMap<String, String> ();
            headers.put ("X-Auth-Service-Provider", TWITTER_VERIFY_CREDENTIALS);
            headers.put                                ("X-Verify-Credentials-Authorization",
verifyCredentialsAuthorizationHeader);

            // step 2 - generate HTTP parameters
            HttpParameter[] params =
            {
                new HttpParameter ("key", twitpicAPIKey),
                new HttpParameter ("media", image)
            };

            // step 3 - upload the file
            HttpClientWrapper client = new HttpClientWrapper ();
            HttpResponse httpResponse = client.post (TWITPIC_UPLOAD_URL, params, headers);

            // step 4 - check the response
            int statusCode = httpResponse.getStatusCode ();
            if (statusCode != 200)
                throw new TwitterException ("Twitpic image upload returned invalid status
code", httpResponse);

            String response = httpResponse.asString ();

            try
            {
                JSONObject json = new JSONObject (response);
                if (! json.isNull ("url"))
                    return json.getString ("url");
            }
            catch (JSONException e)
            {
                throw new TwitterException ("Invalid Twitpic response: " + response, e);
            }

            throw new TwitterException ("Unknown Twitpic response", httpResponse);
        }

        private String generateVerifyCredentialsAuthorizationHeader ()
        {
            List<HttpParameter> oauthSignatureParams = auth.generateOAuthSignatureHttpParams
("GET", TWITTER_VERIFY_CREDENTIALS);
            return          "OAuth          realm=\"http://api.twitter.com/\","          +
OAuthAuthorization.encodeParameters (oauthSignatureParams, ",", true);
        }
    }

    private static class TwitpicBasicAuthUploader extends ImageUpload
    {
        private BasicAuthorization auth;
```

twitter4j/twitter4j-examples/src/main/java/twitter4j/examples/PrintUserStream.java

Chunk 80: (version2/ commentary)

```
    private TwitterStream twitterStream;
<<<<<<< HEAD

    // used for getting status and user info, for favoriting, following and unfollowing
events
=======

    // used for getting user info
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
    private Twitter twitter;
```

```
    private TwitterStream twitterStream;

    // used for getting user info
    private Twitter twitter;
```

Chunk 81: (new code/method declaration)

```
    private int currentUserId;
<<<<<<< HEAD

    public PrintUserStream(String[] args) {
        Configuration                    conf                    =                    new
PropertyConfiguration(getClass().getResourceAsStream("twitter4j.properties"));

        twitterStream     =     new     TwitterStreamFactory().getInstance(conf.getUser(),
conf.getPassword());

        twitter                                    =                                    new
TwitterFactory().getOAuthAuthorizedInstance(conf.getOAuthConsumerKey(),
conf.getOAuthConsumerSecret(),
                new                                AccessToken(conf.getOAuthAccessToken(),
conf.getOAuthAccessTokenSecret()));

        try {
            currentUser = twitter.verifyCredentials();
            currentUserId = currentUser.getId();
        }
        catch (TwitterException e) {
            System.out.println("Unexpected exception caught while trying to retrieve the
current user: " + e);
            e.printStackTrace();
            System.exit(-1);
        }

        Timer t = new Timer(5 * 60 * 1000, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("");
=======

    public PrintUserStream (String [] args)
    {
        Configuration conf = new PropertyConfiguration (getClass ().getResourceAsStream
("twitter4j.properties"));
```

```
        OAuthAuthorization auth = new OAuthAuthorization (ConfigurationContext.getInstance
(), conf.getOAuthConsumerKey (), conf.getOAuthConsumerSecret (),
                  new          AccessToken          (conf.getOAuthAccessToken          (),
conf.getOAuthAccessTokenSecret ()));

        twitterStream = new TwitterStreamFactory ().getInstance (auth);
        twitter = new TwitterFactory().getInstance (auth);

        try
        {
            User currentUser = twitter.verifyCredentials ();
            currentUserId = currentUser.getId ();
        }
        catch (TwitterException e)
        {
            System.out.println ("Unexpected exception caught while trying to retrieve the
current user: " + e);
            e.printStackTrace();
        }

        Timer t = new Timer (5 * 60 * 1000, new ActionListener ()
        {
            @Override
            public void actionPerformed (ActionEvent e)
            {
                System.out.println ("");
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
            }
```

```
    private int currentUserId;

    public PrintUserStream (String [] args) {
        Configuration conf = new PropertyConfiguration (getClass ().getResourceAsStream
("twitter4j.properties"));

        OAuthAuthorization auth = new OAuthAuthorization (ConfigurationContext.getInstance
(), conf.getOAuthConsumerKey (), conf.getOAuthConsumerSecret (),
                  new          AccessToken          (conf.getOAuthAccessToken          (),
conf.getOAuthAccessTokenSecret ()));

        twitterStream = new TwitterStreamFactory ().getInstance (auth);
        twitter = new TwitterFactory().getInstance (auth);

        try {
            User currentUser = twitter.verifyCredentials ();
            currentUserId = currentUser.getId ();
        }
        catch (TwitterException e) {
            System.out.println ("Unexpected exception caught while trying to retrieve the
current user: " + e);
            e.printStackTrace();
        }

        Timer t = new Timer(5 * 60 * 1000, new ActionListener() {
            public void actionPerformed (ActionEvent e)
            {
                System.out.println ("");
            }
```

Case 82: (combination/method declaration, variable)

```
    }

<<<<<<< HEAD
    private Map<Integer, SoftReference<User>> friends;

    public void onFriendList(int[] friendIds) {
        System.out.println("Received friends list - Following " + friendIds.length + "
people");

        friends = new HashMap<Integer, SoftReference<User>>();
        for (int id : friendIds)
            friends.put(id, null);

        friends.put(currentUser.getId(), new SoftReference<User>(currentUser));
    }

    private User friend(int id) {
        User friend = null;

        SoftReference<User> ref = friends.get(id);

        if (ref != null)
            friend = ref.get();

        if (friend == null) {
            try {
                friend = twitter.showUser(id);
                friends.put(id, new SoftReference<User>(friend));
            }
            catch (TwitterException e) {
                e.printStackTrace();
            }
        }

        if (friend == null) {
            System.out.println("User id " + id + " lookup failed");
            return new NullUser();
        }

        return friend;
    }

    public void onStatus(Status status) {
        int replyTo = status.getInReplyToUserId();
        if (replyTo > 0 && !friends.containsKey(replyTo))
            System.out.print("[Out of band] "); // I've temporarily labeled "out of bands"
messages that are sent to people you don't follow

        User user = status.getUser();

        System.out.println(user.getName() + " [" + user.getScreenName() + "] : " +
status.getText());
=======
    private Set<Integer> friends;

    @Override
    public void onFriendList (int [] friendIds)
    {
        System.out.println ("Received friends list - Following " + friendIds.length + "
people");
```

```
        friends = new HashSet<Integer> (friendIds.length);
        for (int id : friendIds)
            friends.add (id);
    }

    public void onStatus (Status status)
    {
        int replyTo = status.getInReplyToUserId ();
        if (replyTo > 0 && !friends.contains (replyTo) && currentUserId != replyTo)
            System.out.print ("[Out of band] "); // I've temporarily labeled "out of band"
messages that are sent to people you don't follow

        User user = status.getUser ();

        System.out.println (user.getName () + " [" + user.getScreenName () + "] : " +
status.getText ());
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
    }
```

```
    }

    private Set<Integer> friends;

    public void onFriendList (int [] friendIds) {
        System.out.println ("Received friends list - Following " + friendIds.length + "
people");

        friends = new HashSet<Integer> (friendIds.length);
        for (int id : friendIds)
            friends.add (id);
    }

    public void onStatus (Status status) {
        int replyTo = status.getInReplyToUserId ();
        if (replyTo > 0 && !friends.contains (replyTo) && currentUserId != replyTo)
            System.out.print ("[Out of band] "); // I've temporarily labeled "out of band"
messages that are sent to people you don't follow

        User user = status.getUser ();

        System.out.println (user.getName () + " [" + user.getScreenName () + "] : " +
status.getText ());
    }
```

Chunk 83: (version 2 /method declaration)

```
    }

<<<<<<< HEAD
    public void onDeletionNotice(StatusDeletionNotice notice) {
        if (notice == null) {
            System.out.println("Deletion notice is null!"); // there's a problem in the
stream, should be done before notification
            return;
        }

        User user = friend(notice.getUserId());
        System.out.println(user.getName() + " [" + user.getScreenName() + "] deleted the
tweet "
                + notice.getStatusId());
```

```
    }

    public void onTrackLimitationNotice(int numberOfLimitedStatuses) {
        System.out.println("track limitation: " + numberOfLimitedStatuses);
=======
    public void onDeletionNotice (StatusDeletionNotice notice)
    {
        User user = friend (notice.getUserId ());
        if (user == null)
            return;
        System.out.println (user.getName () + " [" + user.getScreenName () + "] deleted the
tweet "
                + notice.getStatusId ());
    }

    private User friend (int userId)
    {
        try
        {
            return twitter.showUser (userId);
        }
        catch (TwitterException e)
        {
            System.out.println ("Unexpected exception caught while trying to show user " +
userId + ": " + e);
            e.printStackTrace();
        }

        return null;
    }

    public void onTrackLimitationNotice (int numberOfLimitedStatuses)
    {
        System.out.println ("track limitation: " + numberOfLimitedStatuses);
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
    }
```

```
    }

    public void onDeletionNotice (StatusDeletionNotice notice) {
        User user = friend (notice.getUserId ());
        if (user == null)
            return;
        System.out.println (user.getName () + " [" + user.getScreenName () + "] deleted the
tweet "
                + notice.getStatusId ());
    }

    private User friend(int userId) {
        try {
            return twitter.showUser(userId);
        } catch (TwitterException e) {
            System.out.println("Unexpected exception caught while trying to show user " +
userId + ": " + e);
            e.printStackTrace();
        }

        return null;
    }

    public void onTrackLimitationNotice (int numberOfLimitedStatuses) {
```

```
        System.out.println ("track limitation: " + numberOfLimitedStatuses);
    }
```

Chunk 84: (version 2 /annotation, method declaration)

```
    }

<<<<<<< HEAD
    public void onFavorite(int source, int target, long targetObject) {
        User user = friend(source);
        Status rt = status(targetObject);

        System.out.print(user.getName() + " [" + user.getScreenName() + "] favorited ");
        if (rt == null)
            System.out.println("a protected tweet");
        else
            System.out.println(rt.getUser().getName()        +        "'s        ["        +
rt.getUser().getScreenName() + "] tweet: " + rt.getText());
    }

    public void onUnfavorite(int source, int target, long targetObject) {
        User user = friend(source);
        Status rt = status(targetObject);

        System.out.print(user.getName() + " [" + user.getScreenName() + "] unfavorited ");
        if (rt == null)
            System.out.println("a protected tweet");
        else
            System.out.println(rt.getUser().getName()        +        "'s        ["        +
rt.getUser().getScreenName() + "] tweet: " + rt.getText());
    }

    public void onFollow(int source, int target) {
        User user = friend(source);
        User friend = friend(target);

        System.out.println(user.getName()  +  "  ["  +  user.getScreenName()  +  "]  started
following "
                + friend.getName() + " [" + friend.getScreenName() + "]");

        if (source == currentUserId && friend != null)
            friends.put(target, new SoftReference<User>(friend));
    }

    public void onUnfollow(int source, int target) {
        User user = friend(source);
        User friend = friend(target);

        System.out.println(user.getName() + " [" + user.getScreenName() + "] unfollowed "
                + friend.getName() + " [" + friend.getScreenName() + "]");

        if (source == currentUserId)
            friends.remove(target);
    }

    public void onRetweet(int source, int target, long targetObject) {

    }

    private Status status(long id) {
        try {
            return twitter.showStatus(id);
```

```
        }
        catch (TwitterException e) {
            if (e.getStatusCode() != 403) // forbidden
                e.printStackTrace();
        }

        return null;
    }

    // Preventing the null users in most situations, only used in when there's problems in
the stream

    private static class NullUser implements User {
        public Date getCreatedAt() {
            return null;
        }

        public String getDescription() {
            return null;
        }

        public int getFavouritesCount() {
            return 0;
        }

        public int getFollowersCount() {
            return 0;
        }

        public int getFriendsCount() {
            return 0;
        }

        public int getId() {
            return 0;
        }

        public String getLang() {
            return null;
        }

        public String getLocation() {
            return null;
        }

        public String getName() {
            return null;
        }

        public String getProfileBackgroundColor() {
            return null;
        }

        public String getProfileBackgroundImageUrl() {
            return null;
        }

        public URL getProfileImageURL() {
            return null;
        }

        public String getProfileLinkColor() {
```

```
            return null;
        }

        public String getProfileSidebarBorderColor() {
            return null;
        }

        public String getProfileSidebarFillColor() {
            return null;
        }

        public String getProfileTextColor() {
            return null;
        }

        public String getScreenName() {
            return null;
        }

        public Status getStatus() {
            return null;
        }

        public Date getStatusCreatedAt() {
            return null;
        }

        public long getStatusId() {
            return 0;
        }

        public String getStatusInReplyToScreenName() {
            return null;
        }

        public long getStatusInReplyToStatusId() {
            return 0;
        }

        public int getStatusInReplyToUserId() {
            return 0;
        }

        public String getStatusSource() {
            return null;
        }

        public String getStatusText() {
            return null;
        }

        public int getStatusesCount() {
            return 0;
        }

        public String getTimeZone() {
            return null;
        }

        public URL getURL() {
            return null;
        }
```

```
        public int getUtcOffset() {
            return 0;
        }

        public boolean isContributorsEnabled() {
            return false;
        }

        public boolean isGeoEnabled() {
            return false;
        }

        public boolean isProfileBackgroundTiled() {
            return false;
        }

        public boolean isProtected() {
            return false;
        }

        public boolean isStatusFavorited() {
            return false;
        }

        public boolean isStatusTruncated() {
            return false;
        }

        public boolean isVerified() {
            return false;
        }

        public int compareTo(User o) {
            return 0;
        }

        public RateLimitStatus getRateLimitStatus() {
            return null;
        }

=======
    @Override
    public void onFavorite (User source, User target, Status favoritedStatus)
    {
        System.out.println (source.getName () + " [" + source.getScreenName () + "]
favorited "
                + target.getName () + "'s [" + target.getScreenName () + "] tweet: " +
favoritedStatus.getText ());
    }

    @Override
    public void onUnfavorite (User source, User target, Status unfavoritedStatus)
    {
        System.out.println (source.getName () + " [" + source.getScreenName () + "]
unfavorited "
                + target.getName () + "'s [" + target.getScreenName () + "] tweet: " +
unfavoritedStatus.getText ());
    }

    @Override
    public void onFollow (User source, User target)
```

```
    {
        System.out.println (source.getName () + " [" + source.getScreenName () + "] started
following "
                + target.getName () + " [" + target.getScreenName () +"]");
    }

    @Override
    public void onUnfollow (User source, User target)
    {
        System.out.println  (source.getName  ()  +  "  ["  +  source.getScreenName  ()  +  "]
unfollowed "
                + target.getName () + " [" + target.getScreenName () +"]");

        if (source.getId () == currentUserId)
            friends.remove (target);
    }

    @Override
    public void onUserSubscribedToList (User subscriber, User listOwner, UserList list)
    {
        System.out.println (subscriber.getName () + " [" + subscriber.getScreenName () + "]
subscribed to "
                + listOwner.getName () + "'s [" + listOwner.getScreenName () +"] list: " +
list.getName ()
                + " [" + list.getFullName () + "]");
    }

    @Override
    public void onUserCreatedList (User listOwner, UserList list)
    {
        System.out.println (listOwner.getName () + " [" + listOwner.getScreenName () + "]
created list: " + list.getName ()
                + " [" + list.getFullName () + "]");
    }

    @Override
    public void onUserUpdatedList (User listOwner, UserList list)
    {
        System.out.println (listOwner.getName () + " [" + listOwner.getScreenName () + "]
updated list: " + list.getName ()
                + " [" + list.getFullName () + "]");
    }

    @Override
    public void onUserDestroyedList (User listOwner, UserList list)
    {
        System.out.println (listOwner.getName () + " [" + listOwner.getScreenName () + "]
destroyed list: " + list.getName ()
                + " [" + list.getFullName () + "]");
    }

    @Override
    public void onRetweet (User source, User target, Status retweetedStatus)
    {
    }

    @Override
    public void onBlock (User source, User target)
    {
        System.out.println (source.getName () + " [" + source.getScreenName () + "] blocked
"
                + target.getName () + " [" + target.getScreenName () +"]");
```

```
    }

    @Override
    public void onUnblock (User source, User target)
    {
        System.out.println  (source.getName  ()  +  "  ["  +  source.getScreenName  ()  +  "]
unblocked "
                + target.getName () + " [" + target.getScreenName () +"]");
>>>>>>> 3f7905002910c3d277810bdfd4331d1ddde05de4
    }
```

```
    }

    public void onFavorite (User source, User target, Status favoritedStatus) {
        System.out.println  (source.getName  ()  +  "  ["  +  source.getScreenName  ()  +  "]
favorited "
                + target.getName  ()  +  "'s  ["  +  target.getScreenName  ()  +  "]  tweet:  "  +
favoritedStatus.getText ());
    }

    public void onUnfavorite (User source, User target, Status unfavoritedStatus) {
        System.out.println  (source.getName  ()  +  "  ["  +  source.getScreenName  ()  +  "]
unfavorited "
                + target.getName  ()  +  "'s  ["  +  target.getScreenName  ()  +  "]  tweet:  "  +
unfavoritedStatus.getText ());
    }

    public void onFollow (User source, User target) {
        System.out.println (source.getName () + " [" + source.getScreenName () + "] started
following "
                + target.getName () + " [" + target.getScreenName () +"]");
    }

    public void onUnfollow (User source, User target) {
        System.out.println  (source.getName  ()  +  "  ["  +  source.getScreenName  ()  +  "]
unfollowed "
                + target.getName () + " [" + target.getScreenName () +"]");

        if (source.getId () == currentUserId)
            friends.remove (target);
    }

    public void onUserSubscribedToList (User subscriber, User listOwner, UserList list) {
        System.out.println (subscriber.getName () + " [" + subscriber.getScreenName () + "]
subscribed to "
                + listOwner.getName () + "'s [" + listOwner.getScreenName () +"] list: " +
list.getName ()
                + " [" + list.getFullName () + "]");
    }

    public void onUserCreatedList (User listOwner, UserList list) {
        System.out.println (listOwner.getName () + " [" + listOwner.getScreenName () + "]
created list: " + list.getName ()
                + " [" + list.getFullName () + "]");
    }

    public void onUserUpdatedList (User listOwner, UserList list) {
        System.out.println (listOwner.getName () + " [" + listOwner.getScreenName () + "]
updated list: " + list.getName ()
                + " [" + list.getFullName () + "]");
    }
```

```
    public void onUserDestroyedList (User listOwner, UserList list) {
        System.out.println (listOwner.getName () + " [" + listOwner.getScreenName () + "]
destroyed list: " + list.getName ()
                + " [" + list.getFullName () + "]");
    }

    public void onRetweet (User source, User target, Status retweetedStatus) {
    }

    public void onBlock (User source, User target) {
        System.out.println (source.getName () + " [" + source.getScreenName () + "] blocked
"
                + target.getName () + " [" + target.getScreenName () +"]");
    }

    public void onUnblock (User source, User target) {
        System.out.println  (source.getName  ()  +  "  ["  +  source.getScreenName  ()  +  "]
unblocked "
                + target.getName () + " [" + target.getScreenName () +"]");
    }
}
```

Version: 36e6aaa0f6ad4af6e20f3209016daf8514b613fd

Parents:

6283e3c4c631cf46e08de2e8394850e54191266e

7f303e002665f20bc83aa714a88fd6714294964a

Merge base:

306ba836690fc8fb30384d4b351eb5b77cef1818

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterStream.java

Chunk 85: (version 1/ class signature, variable)

```
    private static final int HTTP_ERROR_WAIT_CAP = 240 * 1000;

<<<<<<< HEAD
    private static final int NO_WAIT = 0;

    abstract class StreamHandlingThread extends Thread {
        private StatusStream stream = null;
        private UserStreamListener userStreamListener;
        private final boolean handleUserStream;
=======
  abstract class StreamHandlingThread extends Thread {
        StatusStream stream = null;
>>>>>>> 7f303e002665f20bc83aa714a88fd6714294964a
        private static final String NAME = "Twitter Stream Handling Thread";
```

```
    private static final int HTTP_ERROR_WAIT_CAP = 240 * 1000;

    private static final int NO_WAIT = 0;

    abstract class StreamHandlingThread extends Thread {
        private StatusStream stream = null;
        private UserStreamListener userStreamListener;
        private final boolean handleUserStream;
        private static final String NAME = "Twitter Stream Handling Thread";
```

Version: 404f639c96735007410ce2f9700aca76173ab671

Parents:

8fb225a96e2ab0247a412e3dfec29ace135bc3db

f1d385b080ef6a2740757d7bc506848e97d90cee

Merge base:

4f76f294c89f46115bb719086b6dbb2e287a6e43

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterStream.java

Chunk 86: (version 1/if statement)

```
                    if (!closed) {
<<<<<<< HEAD
                        if (NO_WAIT == timeToSleep) {
=======
                        if (0 == timeToSleep) {
>>>>>>> f1d385b080ef6a2740757d7bc506848e97d90cee
                            if (te.getStatusCode() > 200) {
```

```
                    if (!closed) {
                        if (NO_WAIT == timeToSleep) {
                            if (te.getStatusCode() > 200) {
```

Version: 7f303e002665f20bc83aa714a88fd6714294964a

Parents:

28288d5dc9b1ffad7cba77d1a106a9dc8e8cbf85

306ba836690fc8fb30384d4b351eb5b77cef1818

Merge base:

b1084ac30a65d0ce8235a11943a6dd39d50927c7

twitter4j/twitter4j-core/src/main/java/twitter4j/TwitterStream.java

Chunk 87: (version 1/ method invocation, return statement, try statement)

```
    public StatusStream getFirehoseStream(int count) throws TwitterException {
<<<<<<< HEAD
        return getCountStream("statuses/firehose.json", count);
=======
        ensureBasicEnabled();
        try {
            return       new       StatusStreamImpl(http.post(conf.getStreamBaseURL()       +
"statuses/firehose.json"
                    , new HttpParameter[]{new HttpParameter("count"
                            , String.valueOf(count))}, auth));
        } catch (IOException e) {
            throw new TwitterException(e);
        }
>>>>>>> 306ba836690fc8fb30384d4b351eb5b77cef1818
    }
```

```
    public StatusStream getFirehoseStream(int count) throws TwitterException {
        return getCountStream("statuses/firehose.json", count);
    }
```

Chunk 88: (version 1/commentary, method invocation, method signature, return statement, try statement)

```
    public StatusStream getLinksStream(int count) throws TwitterException {
<<<<<<< HEAD
        return getCountStream("statuses/links.json", count);
    }

    /**
     * Starts listening on a tweet stream.
     *
     * @param relativeUrl The relative url of the feed, for example "statuses/firehose.json"
for the firehose.
     * @param count Indicates the number of previous statuses to stream before transitioning
to the live stream.
     */
    public void stream(final String relativeUrl, final int count) {
      startHandler(new StreamHandlingThread() {
        public StatusStream getStream() throws TwitterException {
          return getCountStream(relativeUrl, count);
=======
        ensureBasicEnabled();
        try {
            return       new       StatusStreamImpl(http.post(conf.getStreamBaseURL()       +
"statuses/links.json"
                    , new HttpParameter[]{new HttpParameter("count"
```

```
                            , String.valueOf(count))}, auth));
        } catch (IOException e) {
            throw new TwitterException(e);
>>>>>>> 306ba836690fc8fb30384d4b351eb5b77cef1818
        }
```

```
    public StatusStream getLinksStream(int count) throws TwitterException {
        return getCountStream("statuses/links.json", count);
    }

    /**
     * Starts listening on a tweet stream.
     *
     * @param relativeUrl The relative url of the feed, for example "statuses/firehose.json"
for the firehose.
     * @param count Indicates the number of previous statuses to stream before transitioning
to the live stream.
     */
    public void stream(final String relativeUrl, final int count) {
      startHandler(new StreamHandlingThread() {
          public StatusStream getStream() throws TwitterException {
              return getCountStream(relativeUrl, count);
          }
      });
```

Version: 63bf15d15f2533480e43f6b96c44c8fbd593c7a1

Parents:

1cb542632a38ebe77c1dda8b7b54f6ad020f5324

64b00b1fe2f1556ddd1c415fa584b632edd188a0

Merge base:

0a7b28d920f9dc407bd41ca7b3d91f7f7f74a54238

twitter4j/twitter4j-examples/src/main/java/twitter4j/examples/PrintFilterStream.java

Chunk 89: (version 1/method invocation, variable)

```
        }
<<<<<<< HEAD
        trackArray = track.split(",");
=======
>>>>>>> 64b00b1fe2f1556ddd1c415fa584b632edd188a0
    }
```

```
        }
        trackArray = track.split(",");
    }
```

Version: f921017bf7100a8b92057934fc8139e2d4bc2642

Parents:

a76accd1ac400637ed10f344aa299ff7cc1fc43b

e7e7be0c110e70b011f5b098fbc5b134230a9f95

Merge base:

ca81de674fcfc0f491dbd32c4ace1ba94cdcd748

No conflict!

Version: 040b6bf9bf7d7e105eed6ada67e73af0de0e25bf

Parents:

9b23ba9096b720f282cdb1a93d28f0cd0f5717f8

d9e361a4be947ad95c8f57c9f24860611ffa7178

Merge base:

c4d9b86795b4a23e5e7ac0f0644316e3d3f87ec2

twitter4j/src/test/java/twitter4j/conf/ConfigurationTest.java

Chunk 90: (version 1/commentary, method invocation,variable)

```
        deleteFile("./twitter4j.properties");
<<<<<<< HEAD


        // configuration for two different countries and default
        writeFile("./twitter4j.properties", "restBaseURL=http://somewhere.com/"
                + "\n" + "http.useSSL=false"
                + "\n" + "user=one"
                + "\n" + "china.restBaseURL=http://somewhere.cn/"
                + "\n" + "china.user=two"
                + "\n" + "japan.restBaseURL=http://yusuke.homeip.net/"
                + "\n" + "japan.user=three"
        );
        conf = new PropertyConfiguration();
        assertEquals("one", conf.getUser());
        conf = new PropertyConfiguration("/china");
        assertEquals("two", conf.getUser());
        conf = new PropertyConfiguration("/japan");
        assertEquals("three", conf.getUser());


        writeFile("./twitter4j.properties", "restBaseURL=http://somewhere.com/"
                + "\n" + "http.useSSL=false"
                + "\n" + "user=one"
                + "\n" + "password=pasword-one"
                + "\n" + "china.restBaseURL=http://somewhere.cn/"
                + "\n" + "china.user1.user=two"
                + "\n" + "china.user1.password=pasword-two"
                + "\n" + "china.user2.user=three"
                + "\n" + "china.user2.password=pasword-three"
        );
        conf = new PropertyConfiguration();
        assertEquals("one", conf.getUser());
        conf = new PropertyConfiguration("/china/user1");
```

```
        assertEquals("two", conf.getUser());
        assertEquals("pasword-two", conf.getPassword());
        conf = new PropertyConfiguration("/china/user2");
        assertEquals("three", conf.getUser());
        assertEquals("pasword-three", conf.getPassword());

        deleteFile("./twitter4j.properties");
=======
>>>>>>> d9e361a4be947ad95c8f57c9f24860611ffa7178
    }
```

```
        deleteFile("./twitter4j.properties");

        // configuration for two different countries and default
        writeFile("./twitter4j.properties", "restBaseURL=http://somewhere.com/"
                + "\n" + "http.useSSL=false"
                + "\n" + "user=one"
                + "\n" + "china.restBaseURL=http://somewhere.cn/"
                + "\n" + "china.user=two"
                + "\n" + "japan.restBaseURL=http://yusuke.homeip.net/"
                + "\n" + "japan.user=three"
        );
        conf = new PropertyConfiguration();
        assertEquals("one", conf.getUser());
        conf = new PropertyConfiguration("/china");
        assertEquals("two", conf.getUser());
        conf = new PropertyConfiguration("/japan");
        assertEquals("three", conf.getUser());


        writeFile("./twitter4j.properties", "restBaseURL=http://somewhere.com/"
                + "\n" + "http.useSSL=false"
                + "\n" + "user=one"
                + "\n" + "password=pasword-one"
                + "\n" + "china.restBaseURL=http://somewhere.cn/"
                + "\n" + "china.user1.user=two"
                + "\n" + "china.user1.password=pasword-two"
                + "\n" + "china.user2.user=three"
                + "\n" + "china.user2.password=pasword-three"
        );
        conf = new PropertyConfiguration();
        assertEquals("one", conf.getUser());
        conf = new PropertyConfiguration("/china/user1");
        assertEquals("two", conf.getUser());
        assertEquals("pasword-two", conf.getPassword());
        conf = new PropertyConfiguration("/china/user2");
        assertEquals("three", conf.getUser());
        assertEquals("pasword-three", conf.getPassword());

        deleteFile("./twitter4j.properties");
    }
```

Version: 5b85d7b14dd925d9155112e4aead24aa14e9cef5

Parents:

2adc8385fc93ac5a63bbd991b7c80972491a05fb

f47e377e91dc83da0a65654cc7a3cb0e547de3e0

Merge base:

64bfc121ff8a357473929e723293b29a37fc3dd6

twitter4j/src/main/java/twitter4j/http/AccessToken.java

Chunk 91: (version 1/blank)

```
    private int userId;

<<<<<<< HEAD
=======

>>>>>>> f47e377e91dc83da0a65654cc7a3cb0e547de3e0
    AccessToken(Response res) throws TwitterException {
```

```
    private int userId;

    AccessToken(Response res) throws TwitterException {
```

Chunk 92: (version 1/commentary)

```
    }
<<<<<<< HEAD

    /**
     *
     * @return screen name
     * @since Twitter4J 2.0.4
     */
=======

>>>>>>> f47e377e91dc83da0a65654cc7a3cb0e547de3e0
       public String getScreenName() {
```

```
    }

    /**
     *
     * @return screen name
     * @since Twitter4J 2.0.4
     */

       public String getScreenName() {
```

Chunk 93: (version 1/commentary)

```
       }

<<<<<<< HEAD
    /**
     *
     * @return user id
     * @since Twitter4J 2.0.4
     */
```

```
=======
>>>>>>> f47e377e91dc83da0a65654cc7a3cb0e547de3e0
        public int getUserId() {
```

```
        }

    /**
     *
     * @return user id
     * @since Twitter4J 2.0.4
     */

        public int getUserId() {
```