COMP 3111: Introduction to Software Engineering

Phase 0: Individual Assignment (10%) – Writing test cases for a simple Twitter application

Due date: Friday, 1st March 2013 (11:59pm)

Project Description

1. General

TwitterClient is a Twitter desktop client application. It has many basic twitter functions such as reading home timelines, viewing user profiles and tweets, displaying tweet comments and showing the following and follower list of a user.

The TwitterClient project is written in Java language and uses *The Standard Widget Toolkit* (SWT) as its user-interface framework.

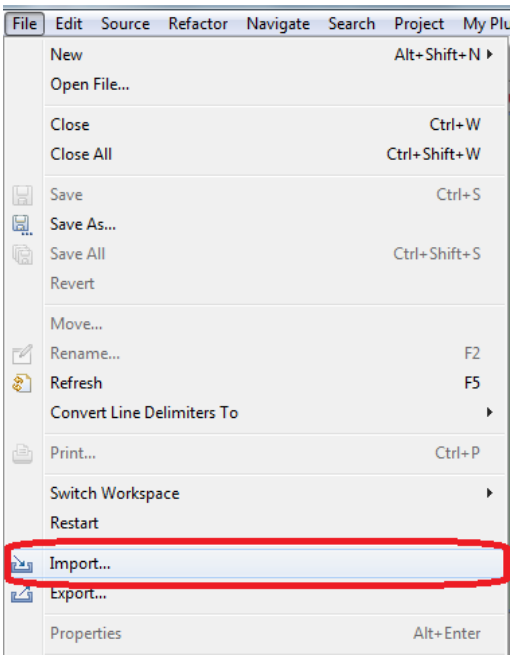
During this course, students are expected to develop new or enhance existing functionalities of this application using the knowledge they have learnt from the course.

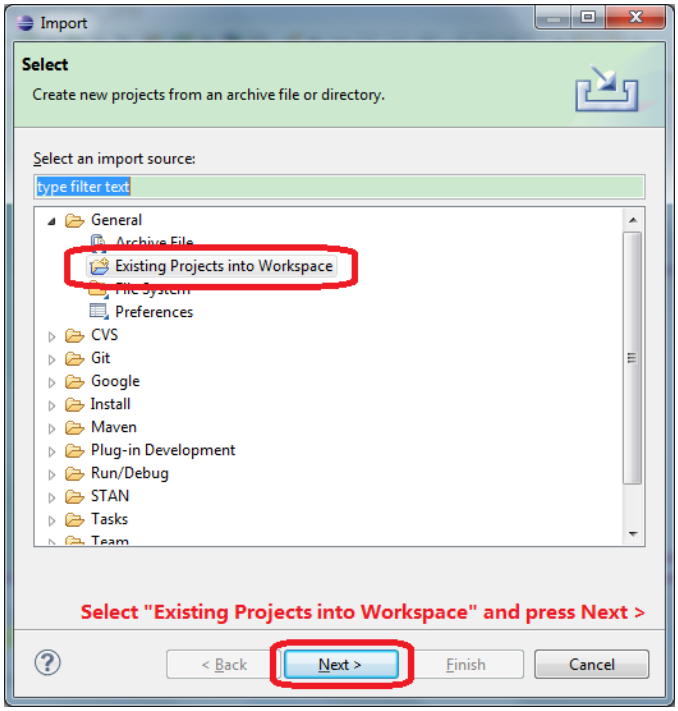
2. Download, Import, and Build

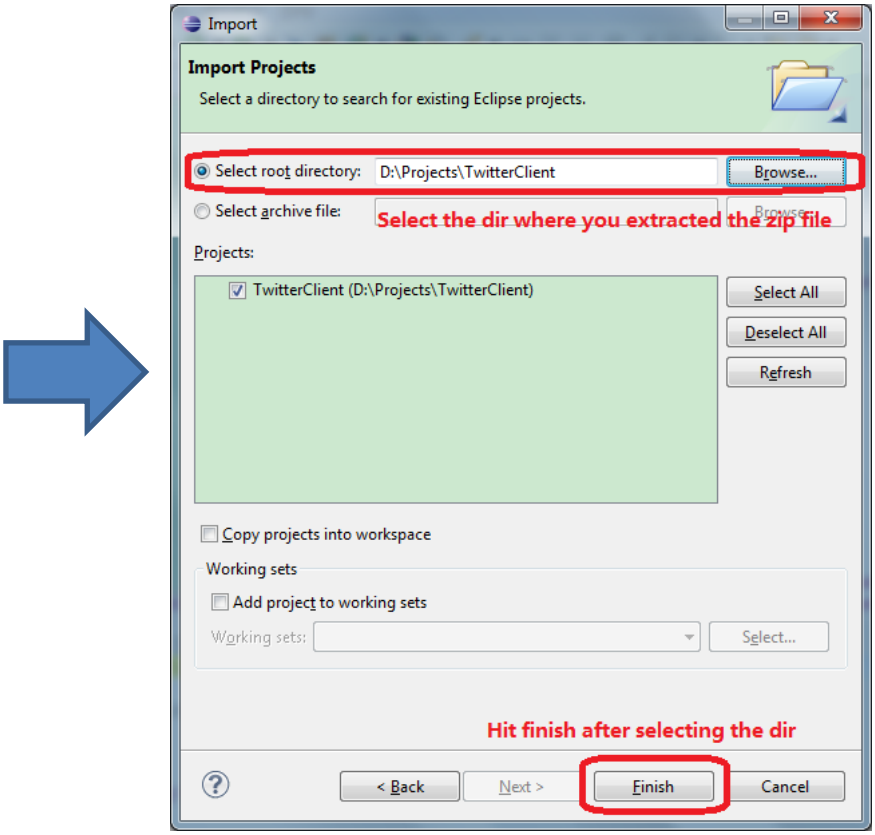
You can download the initial project source code from our course web page: http://course.cse.ust.hk/comp3111/projects/P0/TwitterClient.zip. After downloading the zip file, please unzip it to a location you prefer.

2.1. Building and Executing Project in Eclipse

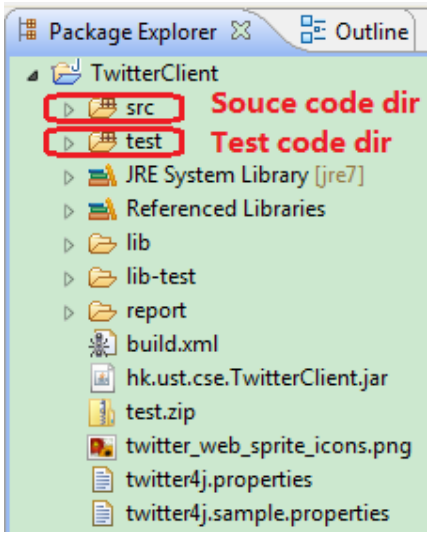
To build and execute the project in eclipse, import the project into eclipse by pressing: *File > Import > General > Existing Projects into Workspace > Next > Select root directory > Finish*


File
Edit
Source
Refactor
Navigate
Search
Project
New
Alt+Shift+N
Open File...
Close
Ctrl+W
Close All
Ctrl+Shift+W
Save
Ctrl+S
Save As...
Save All
Ctrl+Shift+S
Revert
Move...
Rename...
F2
Refresh
F5
Convert Line Delimiters To
Print...
Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties
Alt+Enter


Import
Select
Create new projects from an archive file or directory.
Select an import source:
type filter text
General
Existing Projects into Workspace
Preferences
CVS
Git
Google
Install
Maven
Plug-in Development
Run/Debug
STAN
Tasks
Select "Existing Projects into Workspace" and press Next >
< Back
Next >
Finish
Cancel



After finish importing, there will be two important directories: 1) the *src* directory contains the main source code files of the project; 2) the *test* directory contains the test files of the project.



Obtaining OAuth Keys from Twitter

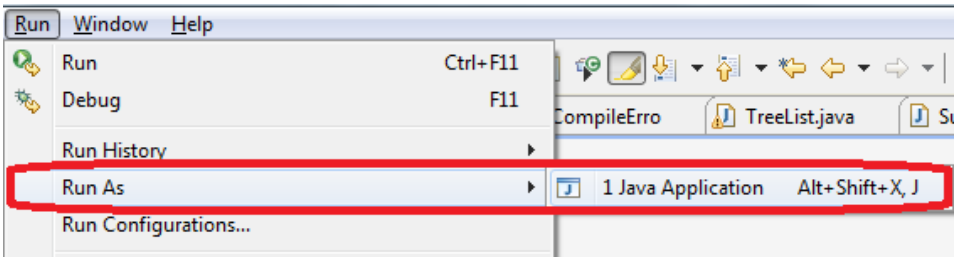
Before executing the project, you first need to obtain four OAuth keys from Twitter, namely *consumerKey*, *consumerSecret*, *accessToken*, *accessTokenSecret*. A detail guide on how to obtain the four oauth keys is available at: https://course.cse.ust.hk/comp3111/projects/P0H/P0H-Sign_Twitter.pdf

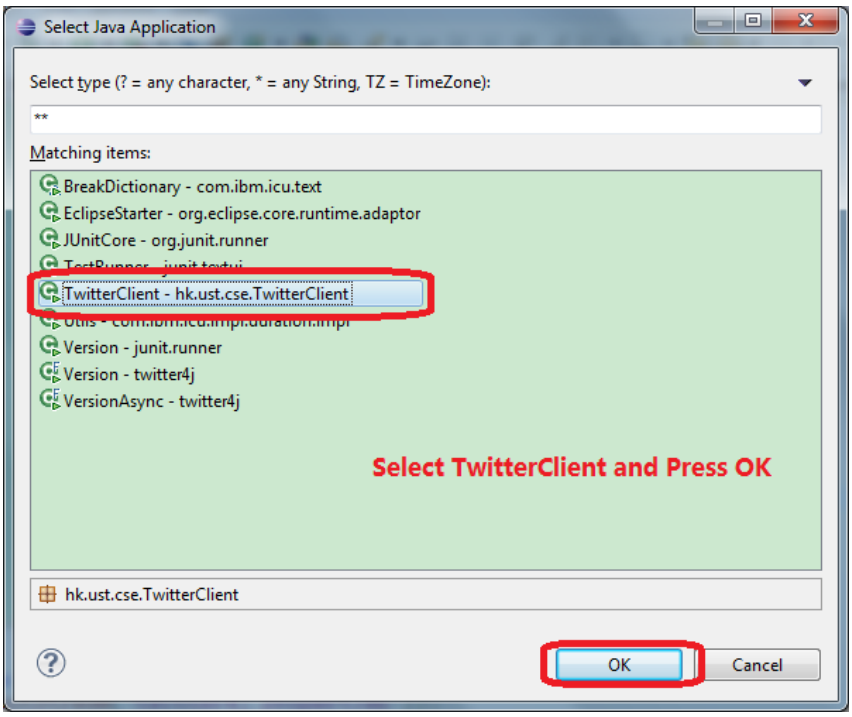
After obtaining the four keys, put them into *twitter4j.properties* file as below:

```
debug=false
oauth.consumerKey=vDhn1IQY0BPTjpk8KMqhbw
oauth.consumerSecret=kXpDqjB5XX1RkJcug1IQkVV6P3lTTm5ZbXDNt3zcQ
oauth.accessToken=1137243308-7MmjiVnA9Gf1LEFqJnCKPrjaocho5A9MWWA60Sc
oauth.accessTokenSecret=1IeMgxTIR4pSbBZIqMw68fAjGjKrEeDirXPjZ3g
```

(Please do not use the keys on the picture, obtain your own from Twitter.)

After putting your oauth keys into twitter4j.properties file, everything is ready. You can now execute the project as follows: double click *TwitterClient.java* to open this file from the *src* directory. Then, select menu item *Run* > *Run As* > *Java Application* > Select *TwitterClient - hk.ust.cse.TwitterClient* > Press *OK*



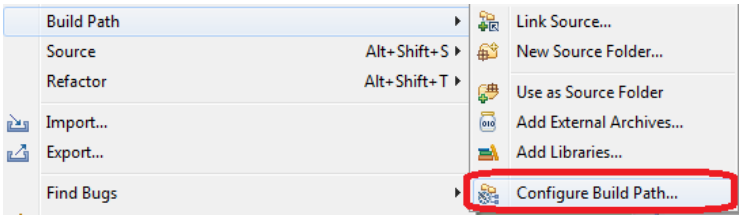


Note that, depending on your network speed, the first run may take 1-2 minutes due to authentication with Twitter. Please be patient.

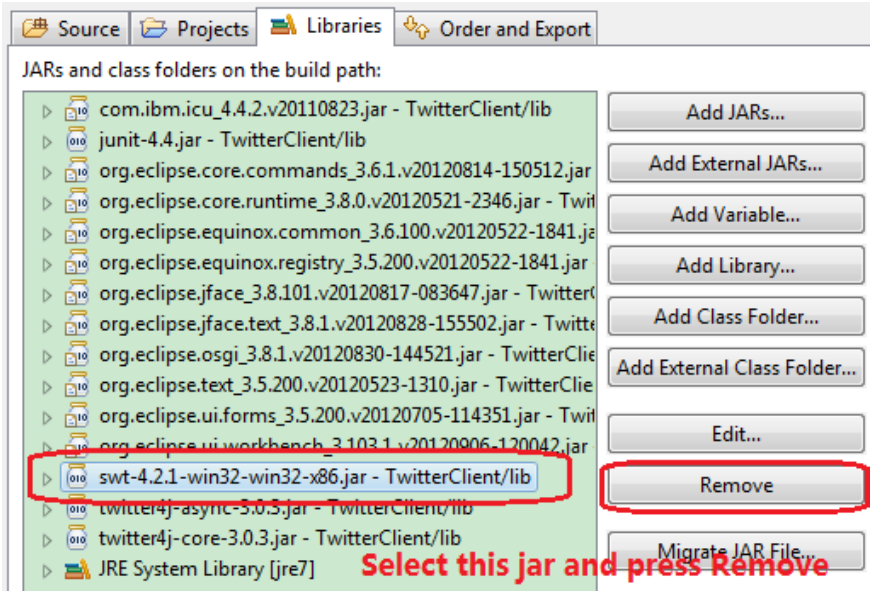
Executing from a 64-bit Machine

In case your machine is running a 64-bit operating system, when you try to execute the project, an error message will be displayed: Wrong SWT library for your platform, please configure your project build path to use SWT library: ./lib/ swt-4.2.1-win32-win32-x86_64.jar

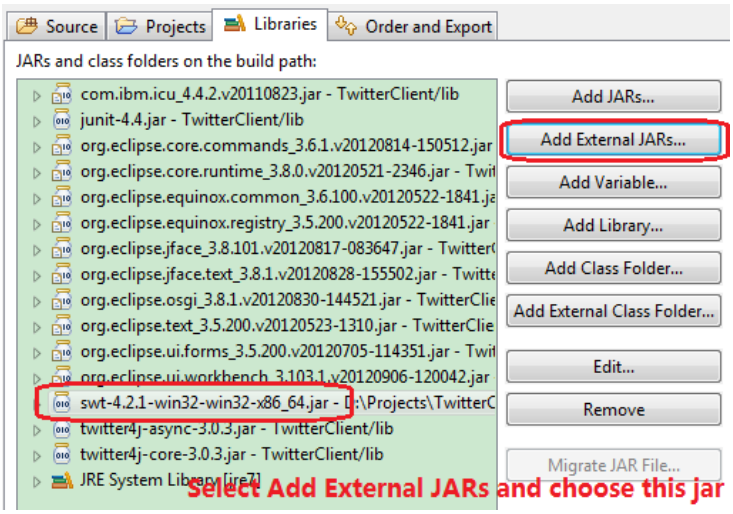
To resolve this problem, right click the project in the *Project Explore* window, select *Build Path* > *Configure Build Path…*



In the *Library* tab, select *swt-4.2.1-win32-win32-x86.jar* and press *Remove*.



Then, press *Add External Jars…*, select *YOUR_PROJECT_DIR/lib/swt-4.2.1-win32-win32-x86_64.jar* in the open file dialog and press *OK*.



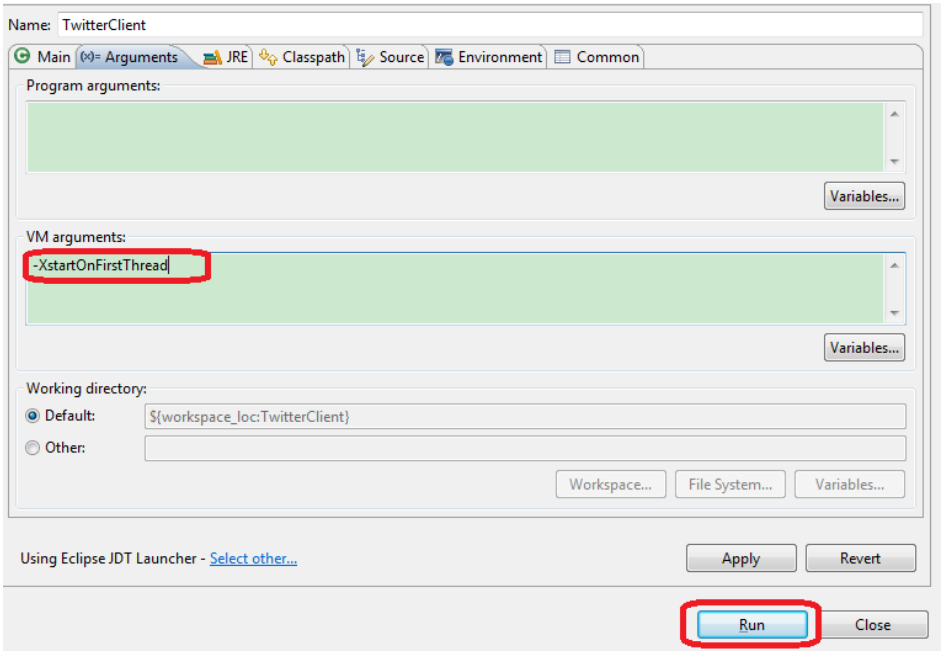
After this step, the original SWT library *swt-4.2.1-win32-win32-x86.jar* is replaced by its 64-bit counterpart, *swt-4.2.1-win32-win32-x86_64.jar*. Now, you can re-execute the project as previously described.

Executing from Mac OS

In case you are using a Mac, you will need to configure the project in two more places.

1) Change the build path of your project as described above, but this time, add *swt-4.2.1-cocoa-macosx-x86.jar* (or *swt-4.2.1-cocoa-macosx-x86_64.jar* if it is a 64-bit Mac) instead of *swt-4.2.1-win32-win32-x86_64.jar* to the build path.

2) Go to menu item *Run* > *Run Configurations…* > Select *Arguments* tab > *Enter -XstartOnFirstThread* in the *VM arguments* text box> Press *Run* button

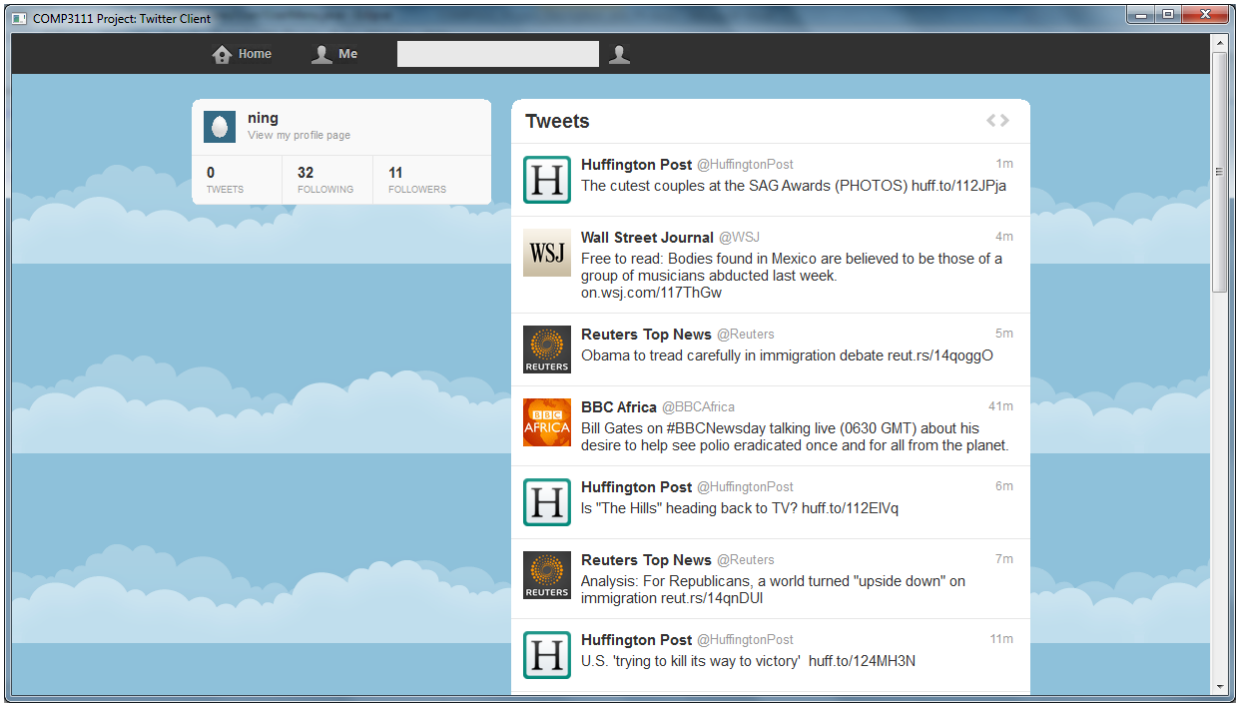


2.2.Building and Executing Project with Ant

Besides building and executing the project in eclipse, we also provide an *ant* script for building the project. To use the ant script, please first install apache-ant on your machine. An installation guide is available in http://ant.apache.org/manual/index.html. Then, you can open a command line window, cd to the project directory, and execute command: *ant build.*

```
D:\Projects\TwitterClient>ant build
```

This ant command will automatically compile the project, output the complied files in a .jar file and execute it (You still need to obtain the oauth keys from Twitter and put them into twitter4j.properties file first.). After successful execution of this command, you should see the interface of TwitterClient similar to below:



Assignment P0 Description

1. Rules

✓ This is an **individual** assignment. All submitted work should be written by yourself individually.

- ✓ Any work submitted to the online testing system will be recorded automatically. Students who copy from others or let others copy from them will be considered **cheating**.

2. Assignment Description

In software engineering, test coverage (e.g. line coverage, branch coverage) achieved by test cases is an important metric for determining the test adequacy of a project. A project with high test coverage is usually considered less bug-prone than a project with low test coverage. In assignment P0, You are asked to write JUnit test cases for the TwitterClient project to achieve as high test coverage as possible.

3. Writing Test Cases

In the *test* directory of the TwitterClient project, some sample test cases are already written for you. For example, file *test.hk.ust.cse.TwitterClient.UtilsTest.java* contains test methods for source code *hk.ust.cse.TwitterClient.Utils.java*.

All test cases written by you should be placed in the project's *test* directory. It is also recommended that, each test file is related to only one source file. For example, test *test.hk.ust.cse.TwitterClient.UtilsTest.java* should only contain test methods for source *hk.ust.cse.TwitterClient.Utils.java*; and test *test.hk.ust.cse.TwitterClient.Views.ControlBarTest.java* should only contain test methods for source *hk.ust.cse.TwitterClient.Views.ControlBar.java*, etc.

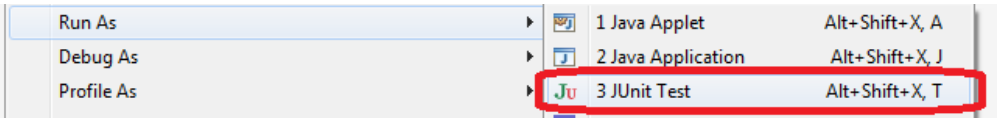
In assignment P0, you are required to write JUnit test cases for the TwitterClient project to achieve as high test coverage (both line coverage and branch coverage) as possible. P0's marks will be given according to the final line and branch coverage achieved by your submitted test cases. There is no limitation on how many tests you can write. And you are not required to modify any project sources (i.e. java files in the *src* directory) during P0.

4. Executing Test Cases Locally

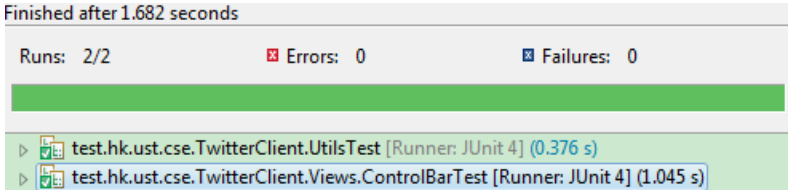
Although marks will be given according to the test coverage achieved by tests submitted online (see Section 5), we strongly recommend that you try out your test cases locally on your own machine before submitting them. There are more than 200 students in this course, and it is likely that our online testing server cannot handle that many requests. Therefore, please try out your written test cases on your own machine first and only submit them to the online testing system when you feel confident with them.

4.1. Executing Test Cases in Eclipse:

In the *Package Explore* window, 1) right click the *test* directory of the TwitterClient project; 2) select *Run As* 3) select *JUnit Test*



This will execute all test methods in all test files under *test* directory. After execution, result will be displayed in a JUnit window within eclipse:



4.2. Obtaining Test Coverage Locally:

We have prepared an *ant* script for obtaining the test coverage from test cases in your own machine. To obtain the test coverage information, open a command line window, cd to your project directory and execute command: *ant coverage* (Note that, you should have already installed apache-ant on your machine as described in *Section 2. Download, Import, and Build*.)

After successful execution of this ant command, a coverage report will be generated at *YOUR_PROJECT_DIRECOTRY/report/cobertura/html/index.html*. Both the line coverage and the branch coverage information will be available in this file.

Since the resource of our online testing system is limited, please be considerate and only submit your test files to the online system after you have tested them on your own machine and want to record the current coverage into the online system.

5. Submitting Test Cases to Online System

Finally, to obtain marks for your P0 assignment, you will need to upload your written test cases for the project onto our online testing system:
http://143.89.191.20:3111/onlinetesting/

5.1. Simple Guide on Using the Online Testing System

1) **Please read the highlighted instructions on the Online Testing System webpage before uploading your test files.**

2) **Student ID and Secret Code**

Enter your student ID, and a secret code. Please choose a secret code that is only recognizable by yourself. For your first upload, the system will NOT check your secret code. But starting from the second time and onwards, you will always need to use the SAME secret code as the first time. **So please keep the secret code you entered at the first time safely**!

3) Uploading Test File

The online testing system only accepts test cases compressed into ONE zip file. To do so, simply compress the *test* directory into a zip file. Please DO NOT compress any source code files from the *src* directory. A sample zip file is available at: http://143.89.191.20:3111/onlinetesting/sample.zip

Alternatively, we also provide an ant script for creating the zip file. Simply open a command line window, cd to your project directory and execute command: *ant zip*.

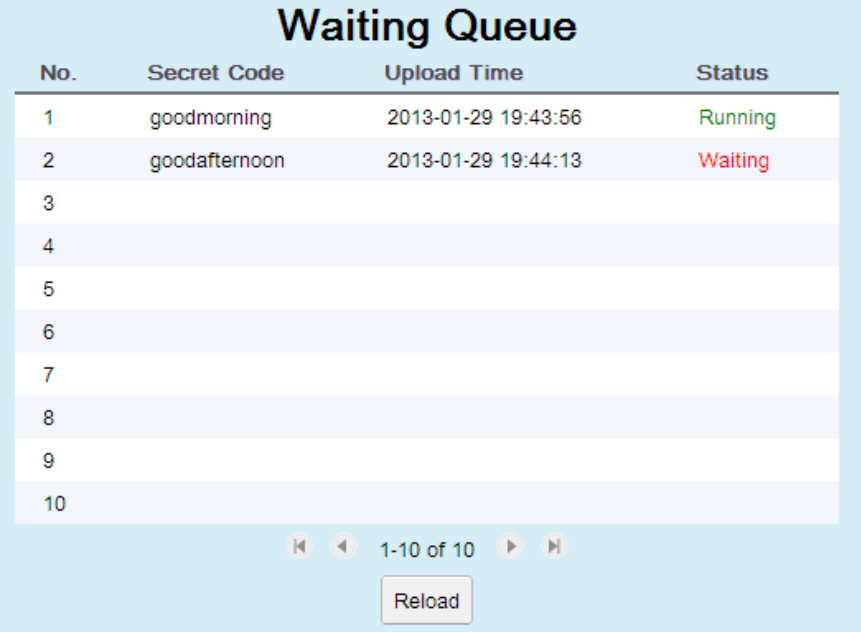
After successful execution of this ant command, a *test.zip* file will be outputted into the project directory and ready to be uploaded.

Note that, if you have uploaded test cases for multiple times, Your P0 mark is decided according to the coverage result of the LAST successful run. No marks will be given if your tests failed to run.

Since our server resource is limited, please be considerate, and only upload test files when you have tested them on your own machine.

4) Waiting in Queue

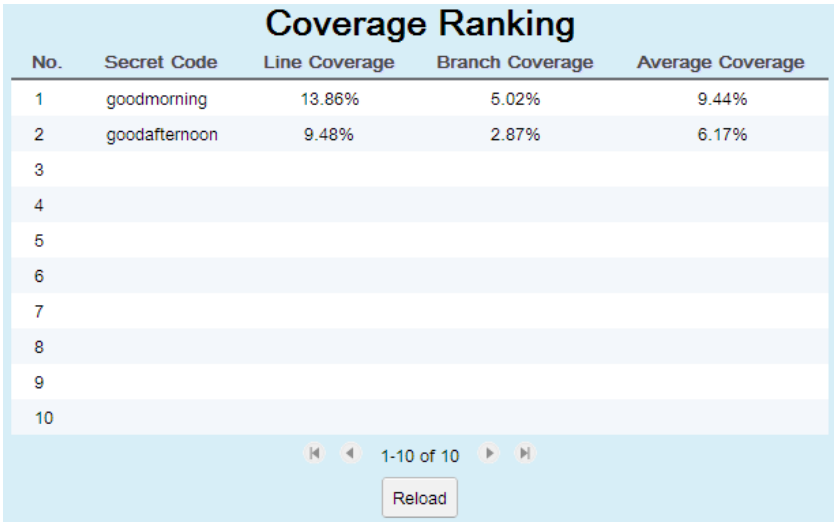
After uploading a test file to the system, it will appear in the ***Waiting Queue***, waiting to be executed by the system. There might be some delay (1-2 minutes) before your test is shown on the waiting queue, please be patient and do NOT upload repeatedly.



If you upload a new test file before your previously uploaded file has been executed, your previous file will be removed from the waiting queue. So each student can only have no more than one file in the waiting queue.

5) Coverage Ranking

The ***Coverage Ranking*** shows the current coverage achieved by each student (only the secret code will be displayed, but not student id) in descending order. You can find your current coverage with your secret code. P0 marks will be given according to the coverage displayed in this ranking.

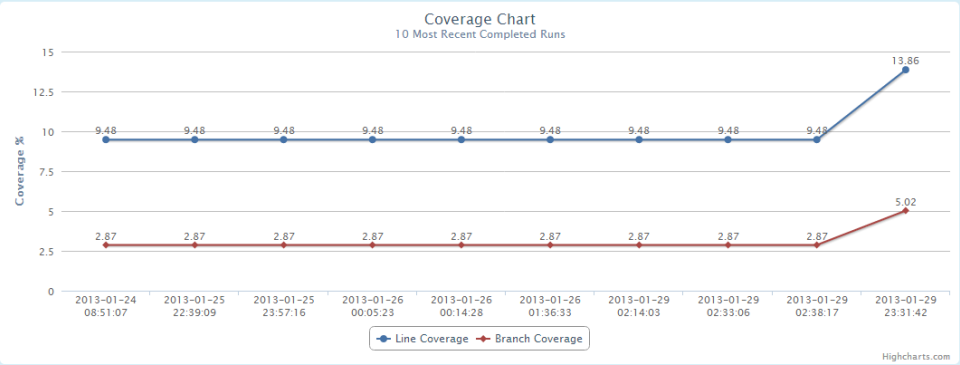


6) More Detail Results

By entering your student id, secret code and press *Show My Results*, you can view the detail information of all your uploaded tests as shown below:

Coverage Results

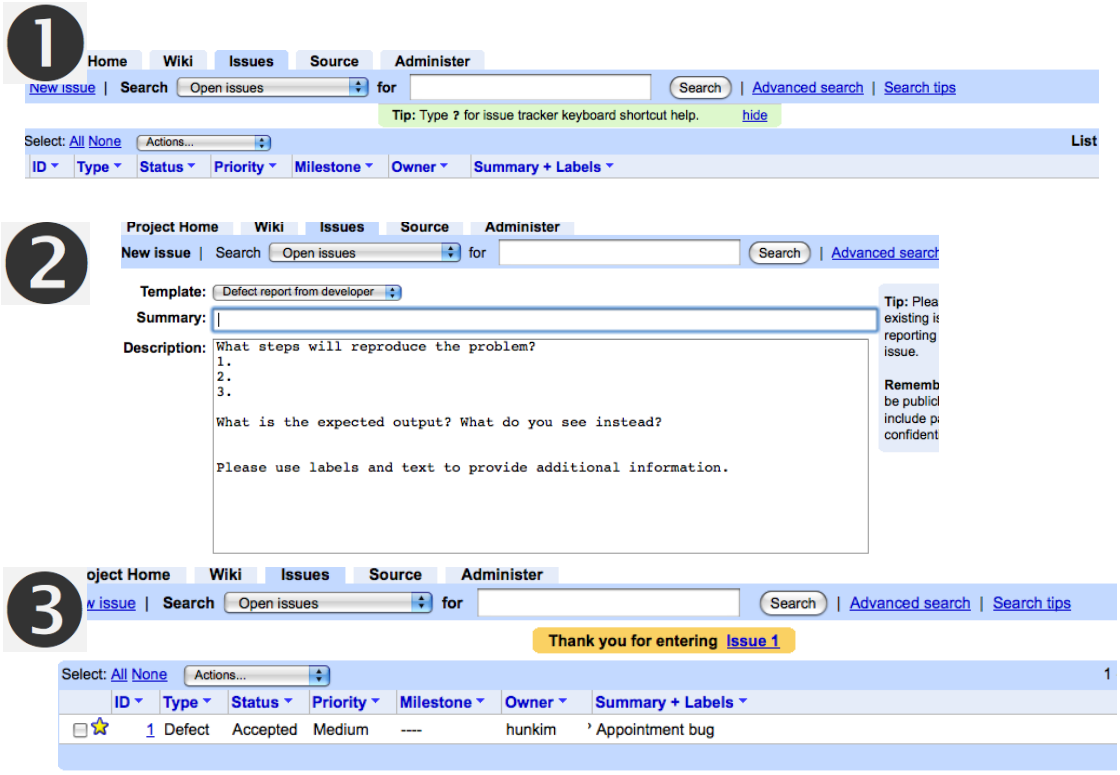
No.	Status	Upload Time	Line Coverage	Branch Coverage	JUnit Report	Standard Output	Error Output
1	COMPLETED	2013-01-22 23:40:54	0.00%	0.00%	JUnit Report	View	View
2	FAILED	2013-01-23 00:51:52	N/A	N/A	JUnit Report	View	View
3	FAILED	2013-01-23 01:05:35	N/A	N/A	JUnit Report	View	View
4	COMPLETED	2013-01-23 01:11:24	0.00%	0.00%	JUnit Report	View	View
5	FAILED	2013-01-23 01:16:28	N/A	N/A	JUnit Report	View	View
6	COMPLETED	2013-01-23 01:49:26	0.26%	0.00%	JUnit Report	View	View
7	FAILED	2013-01-24 03:35:35	0.00%	0.00%	JUnit Report	View	View
8	COMPLETED	2013-01-24 03:46:29	9.48%	2.87%	JUnit Report	View	View
9	COMPLETED	2013-01-24 03:49:40	9.48%	2.87%	JUnit Report	View	View
10	COMPLETED	2013-01-24 08:51:07	9.48%	2.87%	JUnit Report	View	View
11	FAILED	2013-01-25 22:14:57	N/A	N/A	JUnit Report	View	View
12	FAILED	2013-01-25 22:29:04	N/A	N/A	JUnit Report	View	View
13	COMPLETED	2013-01-25 22:39:09	9.48%	2.87%	JUnit Report	View	View
14	FAILED	2013-01-25 22:40:26	N/A	N/A	JUnit Report	View	View
15	FAILED	2013-01-25 22:42:07	N/A	N/A	JUnit Report	View	View
16	TIMEOUT	2013-01-25 22:43:53	N/A	N/A	JUnit Report	View	View
17	TIMEOUT	2013-01-25 23:48:55	N/A	N/A	JUnit Report	View	View
18	COMPLETED	2013-01-25 23:57:16	9.48%	2.87%	JUnit Report	View	View
19	FAILED	2013-01-25 23:58:08	N/A	N/A	JUnit Report	View	View
20	COMPLETED	2013-01-26 00:05:23	9.48%	2.87%	JUnit Report	View	View



6. Reporting bugs (Bonus)

If you can report bugs from the initial code release, each bug finding can help you get an extra 0.2 points. No marks will be given if it is not an actual bug or it has already been reported by other students before. You can issue a bug record under: http://code.google.com/p/3111h/.

Please follow the procedures in the following figure to file a bug report.



Procedures for reporting a bug

7. Further Questions

If you have further questions regarding the system, you can contact: ning@cse.ust.hk