Homework 4: Subversion

Name : _____

Student Number : _____

Laboratory Time : _____

Objectives

- Set up a Subversion repository on UNIX
- Use Eclipse as a Subversion client

Preamble

Subversion (SVN) is a free/open-source version control system. That is, Subversion manages files and directories, and the changes made to them, over time. This allows you to recover older versions of your data, or examine the history of how your data changed.

Subclipse is a Subversion plug-in for Eclipse. It aims to provide all Subversion functionality to the Eclipse development environment. It's an open source project as well, and available at the project homepage, http://subclipse.tigris.org/

In this laboratory, you will create a SVN repository on UNIX. You will also install the Subclipse plug-in. You will then access your SVN repository from the Eclipse workbench using Subclipse. You will then have a working copy of the files.

Grading Checklist (30 points)

- ☐ Configure the UNIX Shell
- ☐ Connect to SVN Repository from Eclipse
- ☐ Check in a Project
- ☐ Check out a Project and revert changes

TA Initials: ____

By the end of the laboratory, you will need to complete the tasks listed below. You will need to do some preparation in advance to ensure that you can complete all of them in the time available. When you have completed all of the tasks, show the TA your work to receive your grade. It's best if you complete all of the tasks and show your results to the TA all at once.

Instructions for the Laboratory

Task 1: Configure Your UNIX Shell

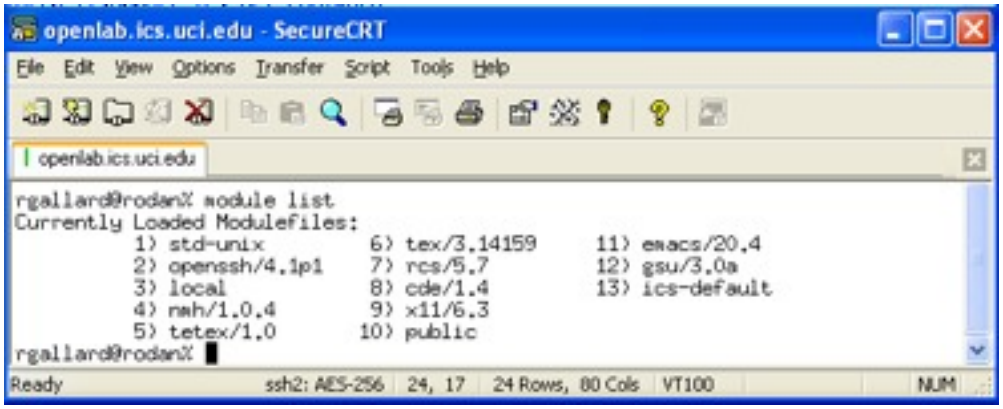
For this task, you will verify that SVN is available in your UNIX environment. If SVN is not available, you will add it. SVN needs to be available so that you can create a SVN repository in your home directory and connect to it using Eclipse in later tasks.

a) Log in to your UNIX account
   1. Open SecureCRT 5.2

   2. Select openlab.ics.uci.edu and click Connect. If a "New Host Key" window appears, click the "Accept and Save" button. Then, it will show a "SecureCRT Window" indicating that the "Access is denied". Click "OK".
   3. Enter your username and click "OK". Then, enter your password and click "OK". You will see the Unix prompt.

b) Type in the command "module list"

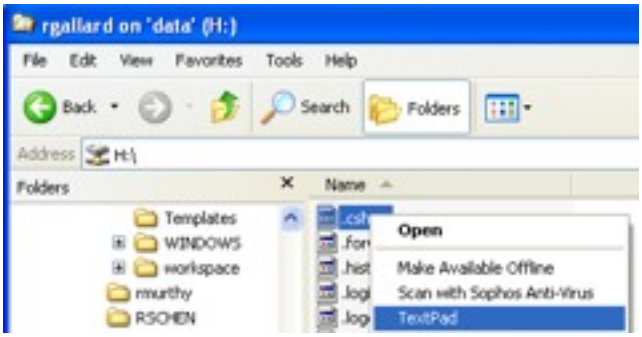
This command will produce a list of modules, or packages of programs, that are available to you in your UNIX environment. The output should look something like the following.



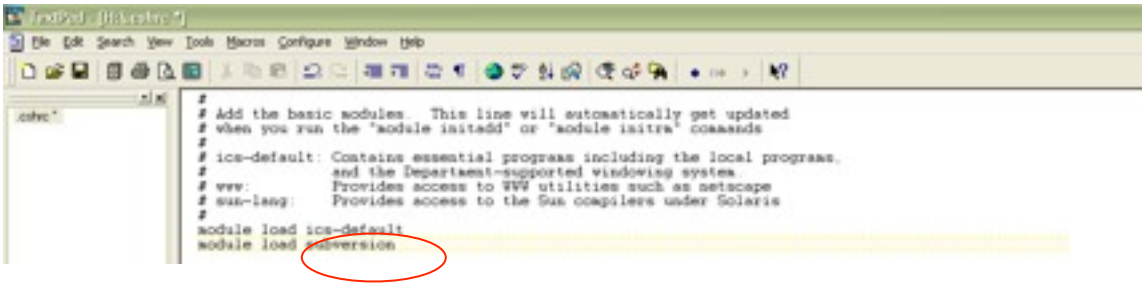
If the output includes Subversion, as in item similar to "`subversion/1.5.2`", then you are done with this task. Otherwise, go to step c).

c) Add the line "module load subversion" to your .cshrc file (notice that there is a period at the beginning of that file name). The following steps will show you how to do this change using Text Pad. If you want to use a UNIX editor such as vi, you can use it.

1. Go to your H: drive using Windows Explorer and find the file .cshrc. Right-click on it and open it using Text Pad.

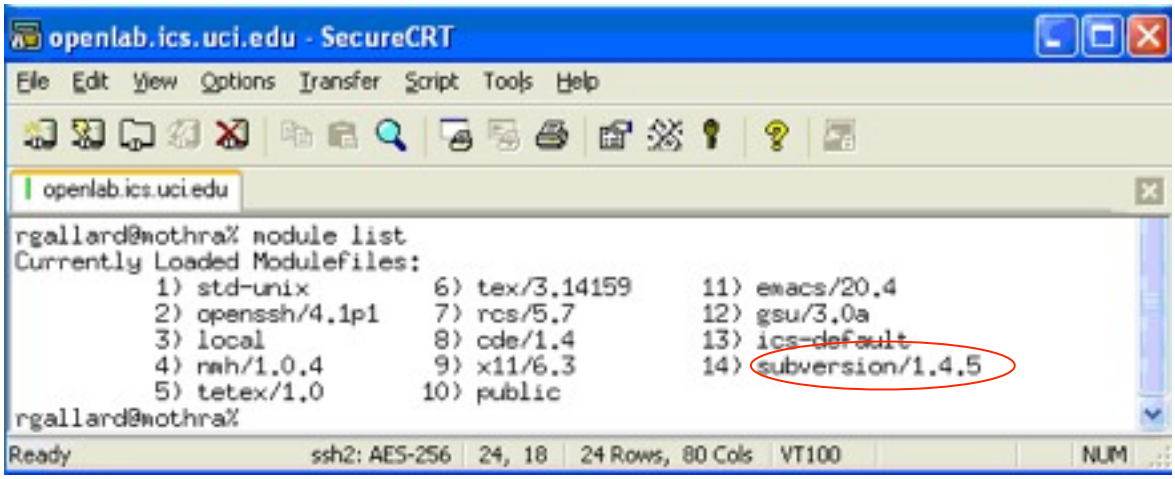


2. You should add the line "module load subversion" close to the other module load commands in the .cshrc file.



3. Make sure you save the file as a UNIX FILE FORMAT. Go to the Save As.. option and indicate: Save as Type: All Files (*.*) and File format: UNIX. If you are asked if you want to replace the file, click "Yes."

d) Log out from SecureCRT using the "exit" command and repeat steps (a) and (b) to verify that the edit was completed correctly. You should see the subversion/1.4.5 module listed.



If you want to know more about how modules work, visit http://www.ics.uci.edu/computing/unix/modules/

Task 2: Add a Directory for Your Subversion Repository

This task involves creating a directory for your Subversion repository in your UNIX Account.

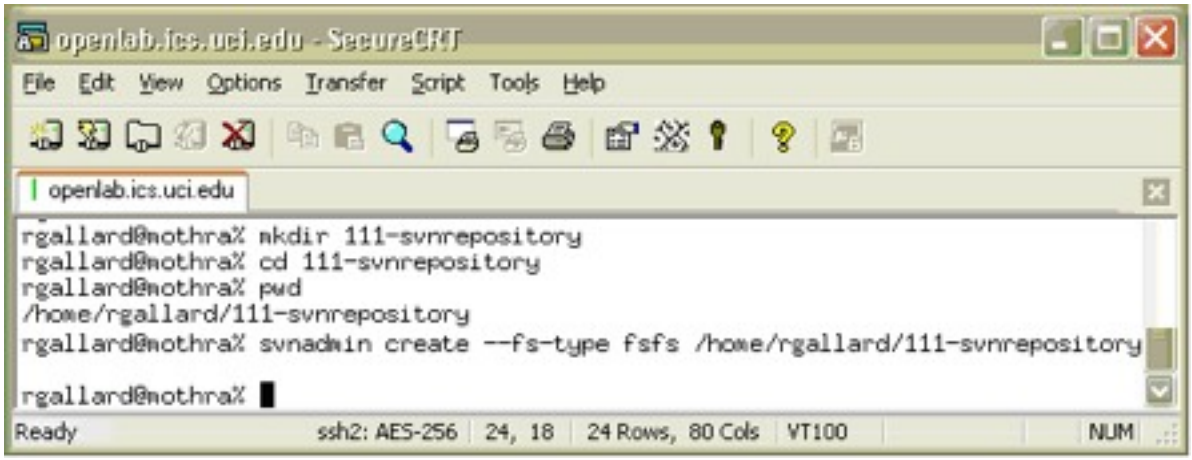
a) Log in to your UNIX account.

b) Create a directory called "111-svnrepository" (or 121-svnrepository, if you prefer). The command to do this is "`mkdir 111-svnrepository`". **Do not place the directory in a subdirectory within your account.**

c) Enter that directory using the command "cd 111-svnrepository"

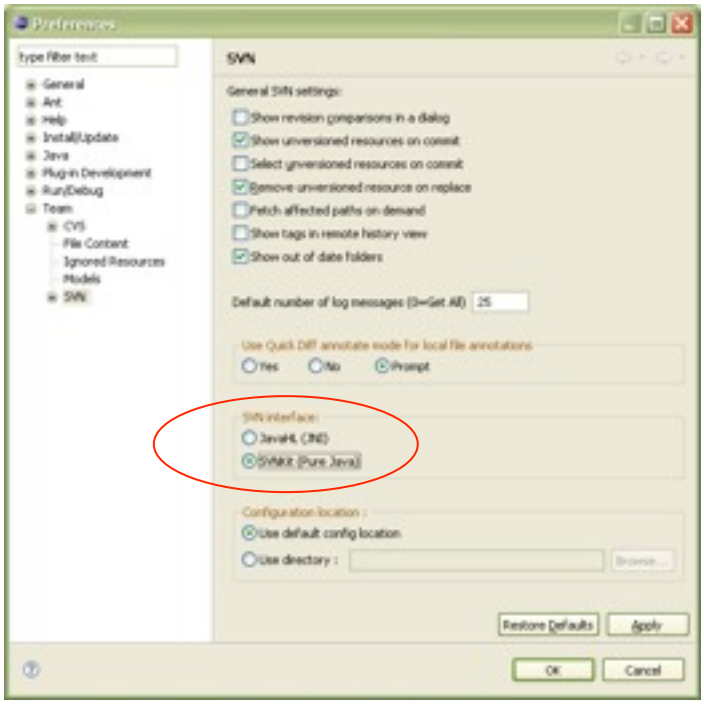
d) Take note of the full path to this directory. Type the command "`pwd`" and write down the result.

e) Initialize the SVN repository: Type in the command "`svnadmin create --fs-type fsfs <path>`". Be sure to substitute the string that you wrote down in step (d) instead of <path>.



Task3: Verify that the Subclipse Plug-in uses SVNKit

a) Go to Window -> Preferences. Expand Team and click SVN. Select "SVNKit (Pure Java)" in the SVN interface information. Click "Apply" and "OK."

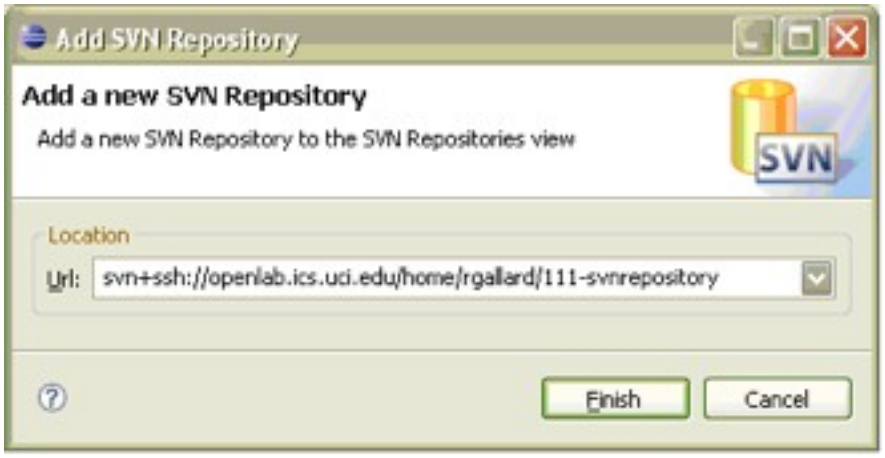


Task 4: Connect to Your SVN Repository from Eclipse

a) In Eclipse, go to the SVN Perspective: Select Window -> Open Perspective -> Other -> SVN Repository Exploring.

b) Create a new Repository Location: In the SVN Repositories tab, right-click and select New -> Repository Location.

c) Indicate the location of the repository. Type in "`svn+ssh://openlab.ics.uci.edu/<path>`". Be sure to substitute the string that you wrote down in step 2d instead of <path>.



d) Enter your password and click OK.

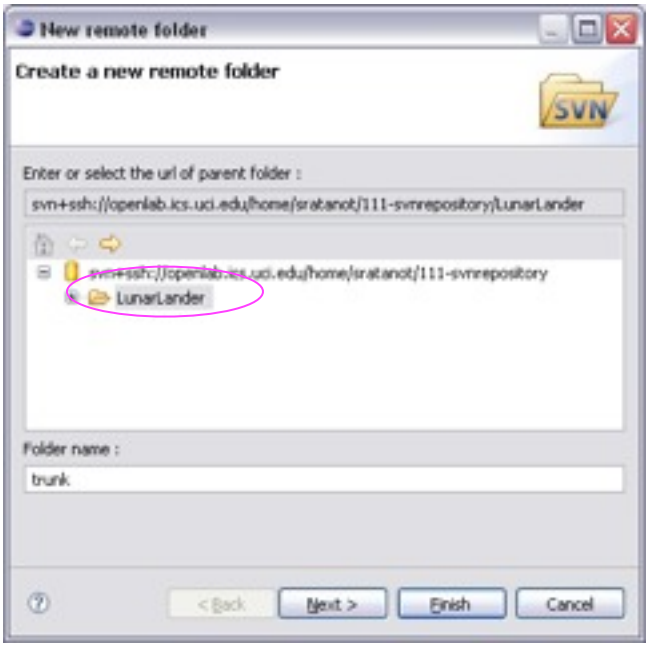
e) You can configure the Author name in SVN. Make sure that **your user name** appears in Author Name. Click OK.

f) You should see the repository you just created.

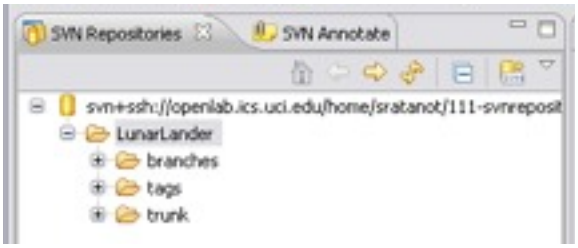
g) Right-click on the repository and select New -> New remote folder.

h) In the New remote folder window, create the folder "LunarLander" and press "Finish."

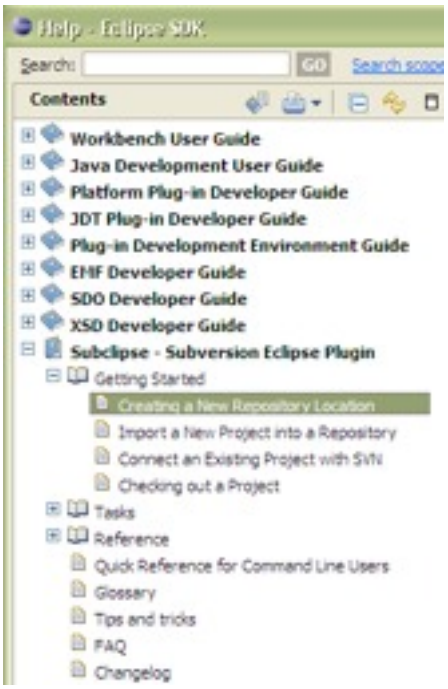
i) Create the following sub folders inside "LunarLander": **"trunk", "branches," and "tags"**. In the New remote folder window, make sure that you select "LunarLander" as the parent folder.



j) Right-click on the repository and select "Refresh." You should see the following structure.



For additional help on setting up a SVN repository, see the online help in Eclipse.



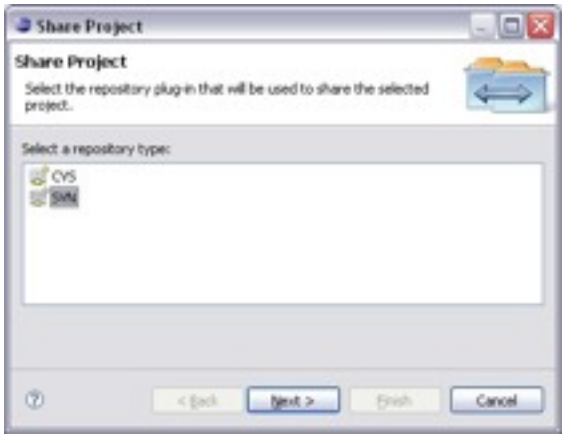
Task 5: Check In a Project

a) Go to the Java Perspective: select Window -> Open Perspective -> Java. Go to the Package Explorer.

b) Look for the project Lunar Lander or Homework1 that we created in Laboratory 1. If the project from Laboratory 1 is not listed in the Package Explorer, open the project from Laboratory 0. If you do not have it, create a project using the Lunar Lander code on the course web site.

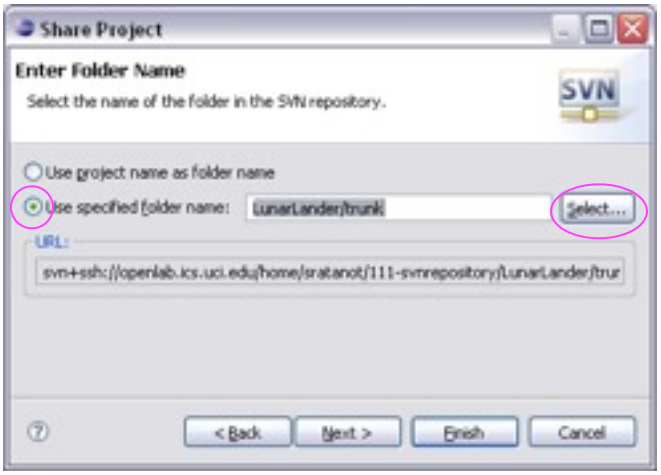
c) Check in your project:
   1. Right-click on the project (Lunar Lander) and select the Team -> Share Project… menu item.

2. Select the SVN repository plug-in and click "Next."

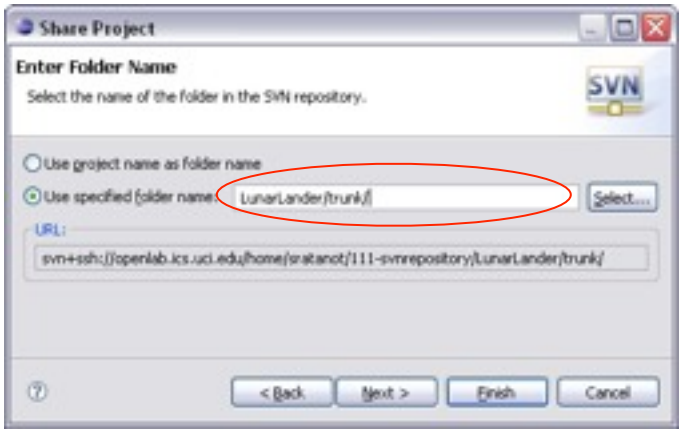


3. Select "Use existing repository location" and click "Next".

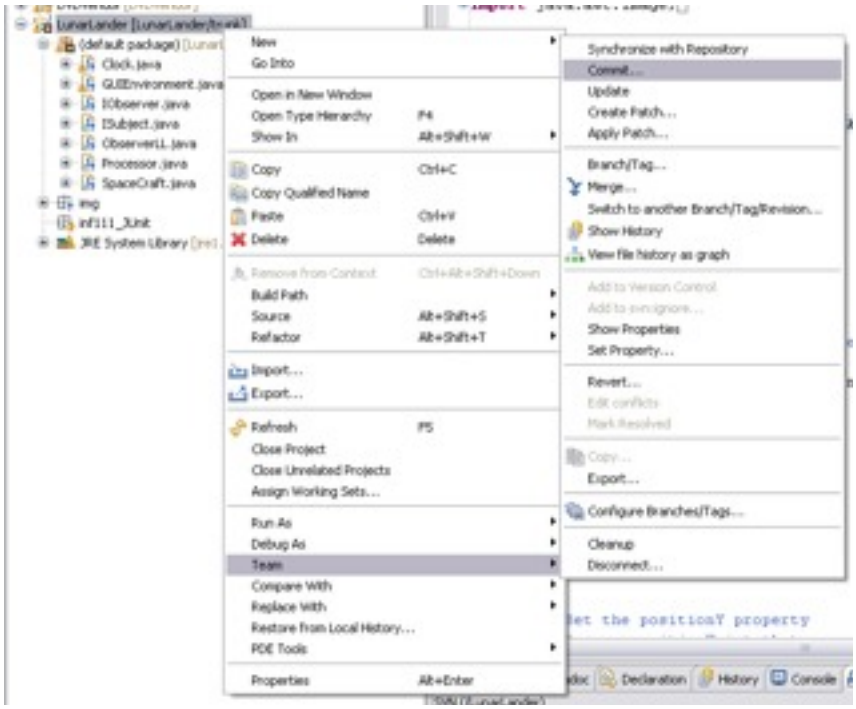
4. In the Enter Folder Name Window, select "Use specified folder name:" and click "Select." Select the "trunk" folder and click OK.



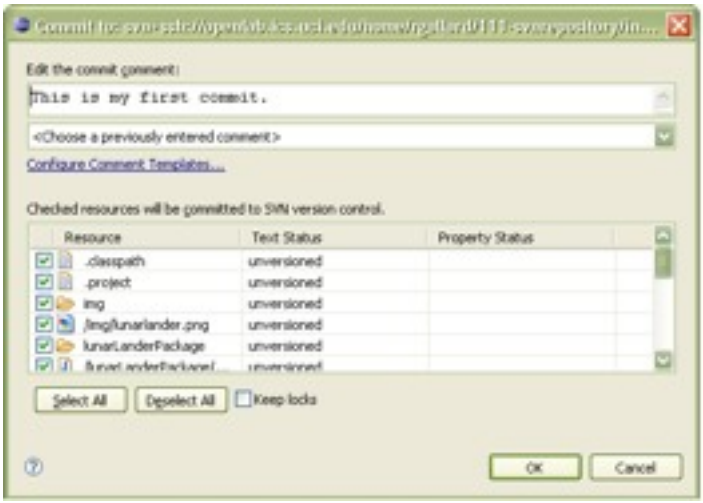
5. Check the project name in the folder name. It should be "LunarLander/trunk/" Click "Next."



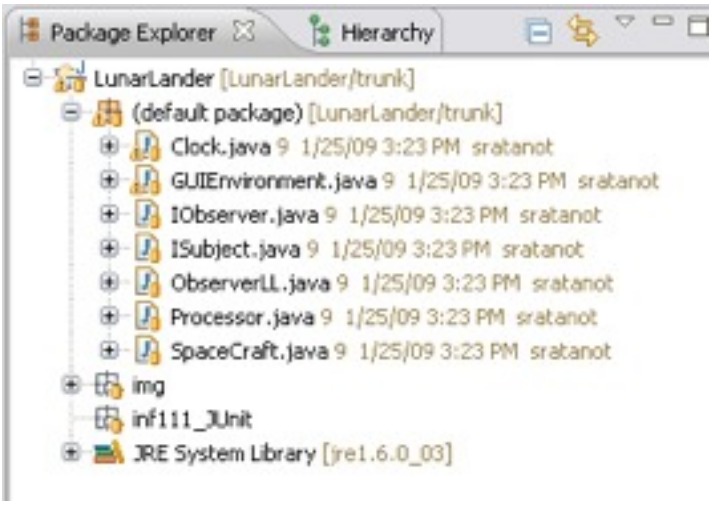
6. You will get a confirmation that the folder already exists in the repository and it will be checked out in your workspace. Select "Yes" to continue
7. You might also receive a dialog asking to open a "Team Synchronization" perspective. Select "No"
8. Right-click on the project (Lunar Lander) and select the Team -> Commit… menu item.



9. Edit the commit comment to "This is my first Commit" and make sure that all files are checked.
10. Click OK to commit the files to the repository.



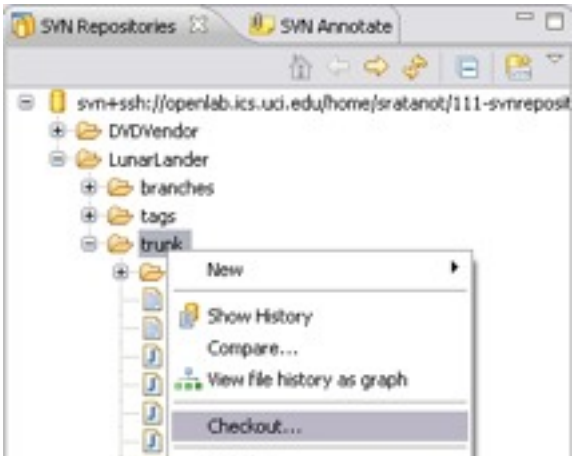
11. You will see your project in the Package Explorer similar to the following screenshot:



Task 6 Check Out a Project and Revert Changes

a) Close the project you just committed (Lunar Lander): Click on the project name and then select Project -> Close Project.

b) Check out the project you just committed (Lunar Lander) from your SVN repository
   1. Go to the SVN Repositories perspective
   2. Right-click on the repository and click "Refresh"
   3. Expand the "LunarLander" folder and select the folder "trunk"
   4. Right-click the project and select the "Checkout" menu item



5. You will see the "Checkout from SVN" Window. Click "Finish." (Note: If you do not want to overwrite your project in your workspace, change the name of the Project.)

6. A "Confirm Overwrite" dialog box will appear if you use the same project name. Click "Ok"
7. Go to the Java Perspective (Window -> Open Perspective -> Java). Make a minor change to the GUIEnvironment.java file: Change lines 23 and 24.
   Before changing:
   ```
   int wRec=70;
   int hRec=70;
   ```
   After changing:
   ```
   int wRec=80;
   int hRec=80;
   ```
8. Save the file you changed (GUIEnvironment.java).

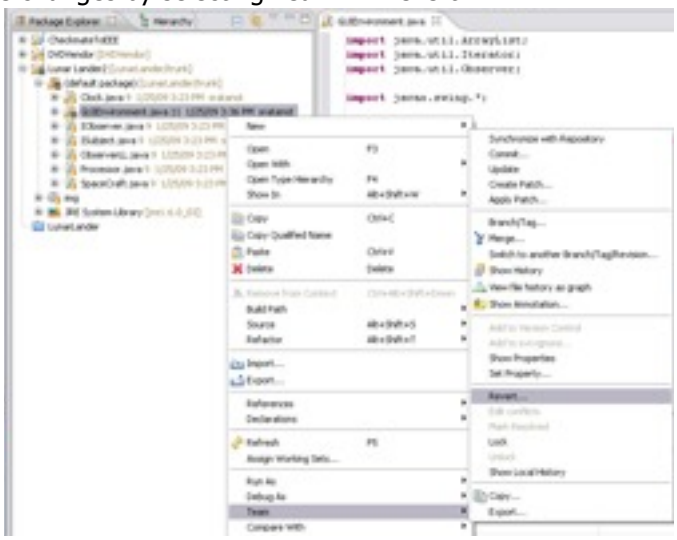
c) Check In the modified project and verify your changes
1. Select the project you are working on.
2. Right-click on the project and select the Team ->Commit… menu item
3. A "Commit to" dialog box will appear. Write a comment "I changed the value of wRec and hRec properties" and click "OK" (note that only the modified file is going to be uploaded to the repository).
4. Go to the SVN Repositories perspective and select the project you are working on
5. Right-click on the repository and select "Refresh". Find the file you modified (GUIEnvironment.java) and you will see the changes you made in this file. You will also see that this file has a different number version from the other files.
6. Check In the modified project and verify your changes

d) Make another change and revert the change
1. Make a minor change again to the GUIEnvironment.java file: Change line 26 and 27.
   Before changing:
   ```
   boolean hit=false;
   boolean landed=false;
   ```
   After changing:
   ```
   boolean hit=true;
   ```

```
boolean landed=true;
```

Save the file.

2. Revert the changes by selecting Team -> Revert



Take Home

1. Create another folder at the root of your repository called "SelfCheckOut". Then create the following folders under "SelfCheckOut": "trunk", "branches," and "tag". Submit a screenshot of your "SVN Repository Exploring" perspective showing the new folders. (15 points)

2. Check in your project for Self Check-Out system created from the previous assignment to the "trunk" folder in the same manner you did with the LunarLander project in the lab. Submit a screenshot of your package explorer showing the shared projects. Expand the "src" folder to show all packages. (15 points)

3. Create three working copies of the Self Check-Out system (one for each of Tiago, Mehryar, and Susan) and perform the following six events, while noting in the table below the different version numbers displayed by the subversion client. (30 points)

The three sandboxes can be created by checking out the "SelfCheckOut/trunk" to different project names, e.g. Mehryar_SelfCheckOut. Assign one sandbox each to Tiago, Mehryar, and Susan, respectively.

The six events are as follows.

Event 1. Mehryar, Tiago, and Susan check out a copy of a version of SelfCheckOut.

Event 2. Mehryar makes changes to line 58 of BIC.java

From:

```
throw (new InvalidBICException("BIC length must be 5"))
```

To:

```
throw (new InvalidBICException("BIC is not 5 digits long."))
```

and checks in the file to the repository.

Event 3. Tiago makes changes to line 58 of BIC.java

From:

```
throw (new InvalidBICException("BIC length must be 5"))
```

To:

```
throw (new InvalidBICException("BIC length must be exactly 5"))
```

and checks in the file to the repository.

Event 4. Tiago makes changes to line 84 of BIC.java

From:

```
return (myBulkItemCode.charAt(myBulkItemCode.length() - 1)) - 48;
```

To:

```
return (myBulkItemCode.charAt(myBulkItemCode.length() - 1)) % 48;
```

and checks in the file to the repository.

Event 5. Tiago updates BIC.java

Event 6. Mehryar makes changes to line 92 of BIC.java

From:

```
private boolean checkLength(String code) {
```

To:

protected boolean checkLength(String code) {

He receives errors and reverts the changes.

i) Enter the appropriate current version numbers of the file "edu.uci.ics.inf111.SelfCheckOut.App.BIC.java" in the repository and each user's working copy, after the event.

ii) Provide a description indicating if the Subversion operations are successful, and how conflicts are resolved. Also, if there is a discrepancy between the version numbers you saw in the screenshot and the one you expected, give a brief explanation of this discrepancy.

iii) Submit a screenshot showing the revision number of files in the package "edu.uci.ics.inf111.SelfCheckOut.App " at the end of each event

Event	Repository	Rosalva's Working Copy	Tiago's Working Copy	Susan's Working Copy
1				
Description:				

Screenshot1:

Event	Repository	Rosalva's Working Copy	Tiago's Working Copy	Susan's Working Copy
2				
Description:				

Screenshot2:

Event	Repository	Rosalva's Working Copy	Tiago's Working Copy	Susan's Working Copy
3				
Description:				

Screenshot3:

Event	Repository	Rosalva's Working Copy	Tiago's Working Copy	Susan's Working Copy
4				
Description:				

Screenshot4:

Event	Repository	Rosalva's Working Copy	Tiago's Working Copy	Susan's Working Copy
5				
Description:				

Screenshot5:

Event	Repository	Rosalva's Working Copy	Tiago's Working Copy	Susan's Working Copy
6				
Description:				

Screenshot6: