Laboratory 2: JUnit

Name	:	______________________________
Student Number	:	______________________________
Laboratory Time	:	______________________________

Objectives

- Creating JUnit Test Cases in Eclipse
- Creating JUnit Test Suites in Eclipse
- Running JUnit Test Cases and Suites in Eclipse

Preamble

JUnit is a unit-testing framework for Java. It provides a common and reusable structure that is required for developing automate and repeatable unit tests for Java classes. JUnit provides a base class called TestCase that can be extended to create series of tests for your classes, an assertion library that can be used to evaluate the results of the tests, and test drivers, both command line and GUI based, called TestRunner to run the test cases you create.

The recent version of the Eclipse JDT already has JUnit Plug-in built in to make creating and running test cases more convenient. The plug-in includes a wizard for assisting in creating testing a test case and test suite, and an environment for running them.

In this lab, you will learn how to set up a project for creating JUnit tests. Then you will create test cases and a test suite, and run them.

Grading Checklist

By the end of the laboratory session, you need to demonstrate to the TA that you can do the following tasks. The TA will check off the items below that you have completed and collect this cover page from you.

- ☐ JUnit library is in the project's build path
- ☐ Test case for GUIEnvironment class has been created and asserts added to init method body
- ☐ The test case for GUIEnvironment runs successfully (may find errors in GUIEnvironment)
- ☐ Test case for Processor class has been created and assert added to calculate method body
- ☐ The test case for Processor runs successfully (may find errors in Processor)
- ☐ Test suite created and works

TA Initials: ____

Instructions for the Laboratory

Task 1: Set up a new project and create JUnit Test Cases for the GUIEnvironment class

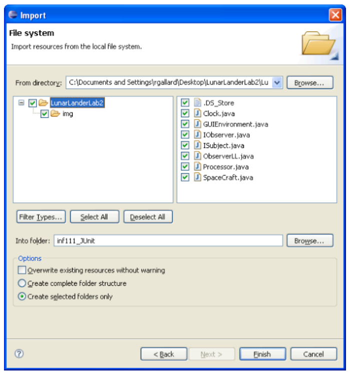
For this task, you will set up a new project and include the provided classes in the project. Then you will create a test case to test the GUIEnvironment class. In JUnit convention, a test class is created for every application class, and every non-trivial method is tested.

a) Download and uncompress LunarLanderLab2.zip, which contains GUIEnvironment.java and other Java files as in lab1.

b) Create a new Java project in Eclipse called inf111_JUnit.

c) It is a good practice to separate the application source code from test case code.

   1. Import GUIEnvironment.java and other files in the zip file into a default (unnamed) package. You can do this by right clicking on the project and then selecting Import → General → File System. You need to select the directory where you have Java files residing. You can use Eclipse's file filtering capabilities here.



Import

File system

Import resources from the local file system.

From directory: C:\Documents and Settings\rgallard\Desktop\LunarLanderLab2\Lu

- [✓] LunarLanderLab2
  - [✓] img

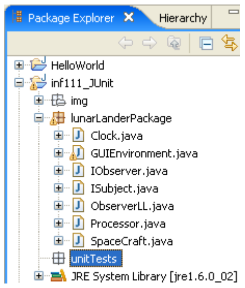
- [✓] .DS_Store
- [✓] Clock.java
- [✓] GUIEnvironment.java
- [✓] IObserver.java
- [✓] ISubject.java
- [✓] ObserverLL.java
- [✓] Processor.java
- [✓] SpaceCraft.java

Into folder: inf111_JUnit

Options

- [ ] Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

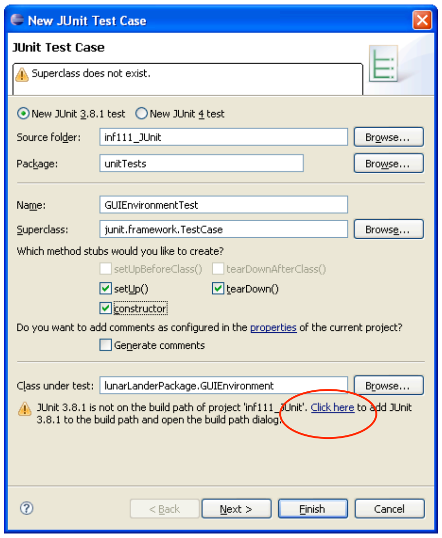
   2. Create a new package and name it lunarLanderPackage (note the lowercase 'l')
   3. Use the refactoring feature to move the java files from the default package into the lunarLanderPackage that you just created.
   4. Create a new package for test source code by right clicking on the project and selecting new package and name the package unitTests.



```
HelloWorld
inf111_JUnit
  img
  lunarLanderPackage
    Clock.java
    GUIEnvironment.java
    IObserver.java
    ISubject.java
    ObserverLL.java
    Processor.java
    SpaceCraft.java
  unitTests
  JRE System Library [jre1.6.0_02]
```

d) Create a JUnit Test Case Class.

   1. First, to bring up the New JUnit Test Case Wizard, select the unitTests package. Then, select File → New → JUnit Test Case.
   2. Add JUnit library to the build path by clicking on the link at the bottom on the dialog box that pops up when you are creating the test case.


New JUnit Test Case

JUnit Test Case

Superclass does not exist.

(●) New JUnit 3.8.1 test ( ) New JUnit 4 test

Source folder: inf111_JUnit

Package: unitTests

Name: GUIEnvironmentTest

Superclass: junit.framework.TestCase

Which method stubs would you like to create?

- [ ] setUpBeforeClass()
- [ ] tearDownAfterClass()
- [✓] setUp()
- [✓] tearDown()
- [✓] constructor

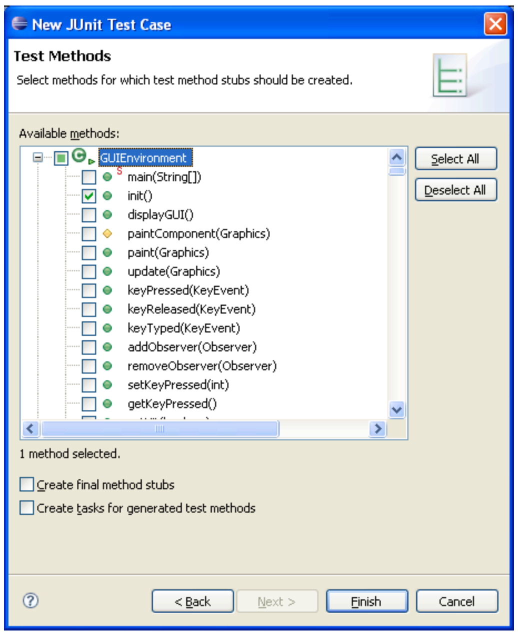
Do you want to add comments as configured in the properties of the current project?

- [ ] Generate comments

Class under test: lunarLanderPackage.GUIEnvironment

JUnit 3.8.1 is not on the build path of project 'inf111_JUnit'. Click here to add JUnit 3.8.1 to the build path and open the build path dialog.

3. Name the class by putting the name GUIEnvironmentTest in the name test box.
4. Select setUp() and tearDown() boxes to automatically generate skeleton of these methods. The setUp() and tearDown() methods are run before and after each test case is run.
5. For "Class under test", enter GUIEnvironment. Eclipse should find GUIEnvironment class in the lunarLanderPackage and should select it (Browse..). Then press Next.


New JUnit Test Case

Test Methods

Select methods for which test method stubs should be created.

Available methods:

GUIEnvironment
- [ ] main(String[])
- [✓] init()
- [ ] displayGUI()
- [ ] paintComponent(Graphics)
- [ ] paint(Graphics)
- [ ] update(Graphics)
- [ ] keyPressed(KeyEvent)
- [ ] keyReleased(KeyEvent)
- [ ] keyTyped(KeyEvent)
- [ ] addObserver(Observer)
- [ ] removeObserver(Observer)
- [ ] setKeyPressed(int)
- [ ] getKeyPressed()

1 method selected.

- [ ] Create final method stubs
- [ ] Create tasks for generated test methods

6. Now you can select methods for which test method stubs will be created. Select init().
7. Press Finish. A new class, GUIEnvironmentTest, that extends junit.framework.TestCase is created with generated method stubs.

e) Implement a test case function in GUIEnvironmentTest to test the GUIEnvironment class.
   1. Create a new class variable, lunarLanderEnvironment as a private variable of the type GUIEnvironment.
      ```
      GUIEnvironment lunarLanderEnvironment;
      ```
   2. Implement the setUp() method to initialize variables.
      ```
      lunarLanderEnvironment = new GUIEnvironment();
      ```
   3. Implement the tearDown() method to clear the variable.
      ```
      lunarLanderEnvironment = null;
      ```
   4. Implement the testInit() to check that GUIEnvironment initialization method is working correctly.
      ```
      lunarLanderEnvironment.init();
      assertFalse(lunarLanderEnvironment.getHit());
      assertNotNull(lunarLanderEnvironment.getSpaceCraft());
      ```
   5. Add in testInit() two assertions to validate that Clock and Sensor (properties of GUIEnvironment) are not null.

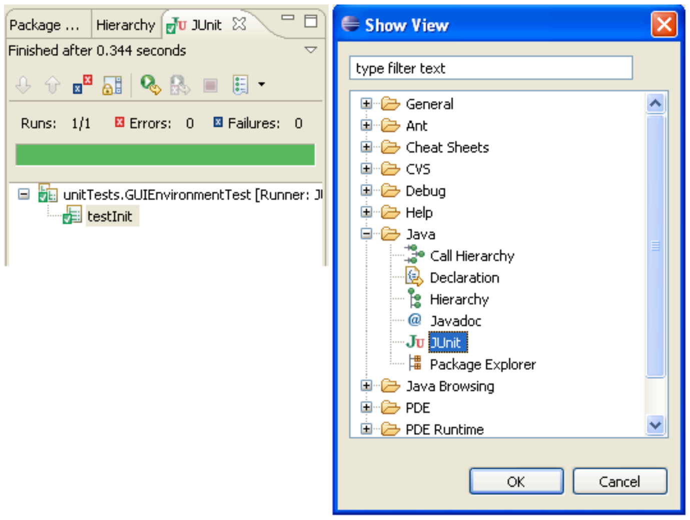
If you get a compilation error, be sure to import the required packages, including junit.framework.Assert package.

You can find more information on the assert API in JUnit JavaDoc at http://www.junit.org/junit/javadoc/3.8.1/index.htm

Task 2: Running GUIEnvironmentTest as a JUnit test case

In this task, you will run the test case in GUIEnvironmentTest as a JUnit Test Case.

a) Right click at GUIEnvironmentTest, and select Run → JUnit Test.
b) You should see the result of your test case in JUnit view. If the view does not appear, show the JUnit view by selecting Window → Show View → Other → Java → JUnit.



```
Finished after 0.344 seconds
Runs: 1/1   Errors: 0   Failures: 0
unitTests.GUIEnvironmentTest [Runner: JU
    testInit
```

Task 3: Create JUnit Test Case for the Processor class
(You may want to skip this task if you are running behind on time)

a) Now repeat Task 1, step d) to create a test case for the calculate method in the Processor class. Implement a test case function in ProcessorTest to test the Processor class.

1. Create three new class variables, lunarLanderEnvironment as a private variable of the type GUIEnvironment, spaceCraft as a private variable of the type SpaceCraft and processor a private variable of the type Processor.
   ```
   GUIEnvironment lunarLanderEnvironment;
   SpaceCraft spaceCraft;
   Processor processor;
   ```

2. Implement the setUp() method to initialize variables.
   ```
   processor = new Processor();
   lunarLanderEnvironment = new GUIEnvironment();
   lunarLanderEnvironment.init();
   spaceCraft = lunarLanderEnvironment.getSpaceCraft();

   JFrame app = new JFrame();
   app.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
   app.getContentPane().add (lunarLanderEnvironment);
   app.setSize (800,800);
   app.setBackground (Color.WHITE);
   app.setLocation (0,0);
   app.addKeyListener (lunarLanderEnvironment);
   app.setVisible (true);
   ```

3. Implement the tearDown() method to clear the variable.
   ```
   lunarLanderEnvironment = null;
   spaceCraft = null;
   processor = null;
   ```

4. Implement the testCalculate() to check that the calculate method is working correctly.
   ```
   int rotation;
   rotation = spaceCraft.getRotation();
   //simulates right key pressed
   lunarLanderEnvironment.setKeyPressed(KeyEvent.VK_RIGHT
   );
   processor.calculate(spaceCraft,
   lunarLanderEnvironment);
   assertEquals(rotation + spaceCraft.getRotationRate(),
   spaceCraft.getRotation());
   ```

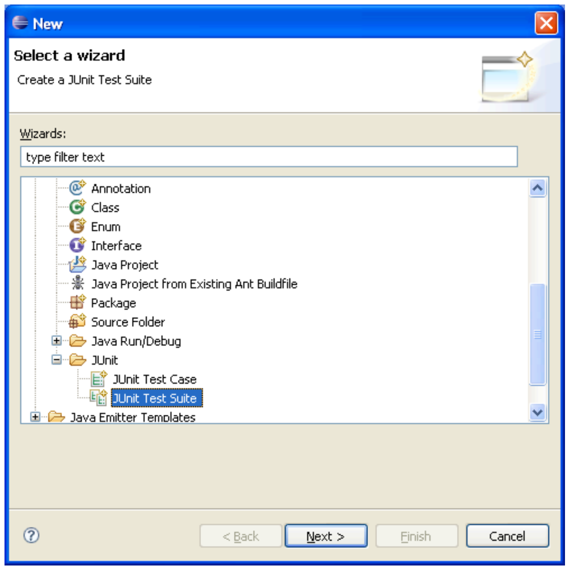
5. Run ProcessorTest as a JUnit Test Case (See Task 2).

If you get a compilation error, be sure to import the required packages, including junit.framework.Assert package.

Task 4: Creating a Test Case Suite

In this task, you will create a test case suite for the test cases created in Task 1 and Task3. A test case suite allows more convenient test case execution and will allow you to run all your test cases at once.

a) Create a Test Suite by selecting the unitTests package, and then selecting File→ New → Other → Java → JUnit Test Suite.


New

Select a wizard

Create a JUnit Test Suite

Wizards:

type filter text

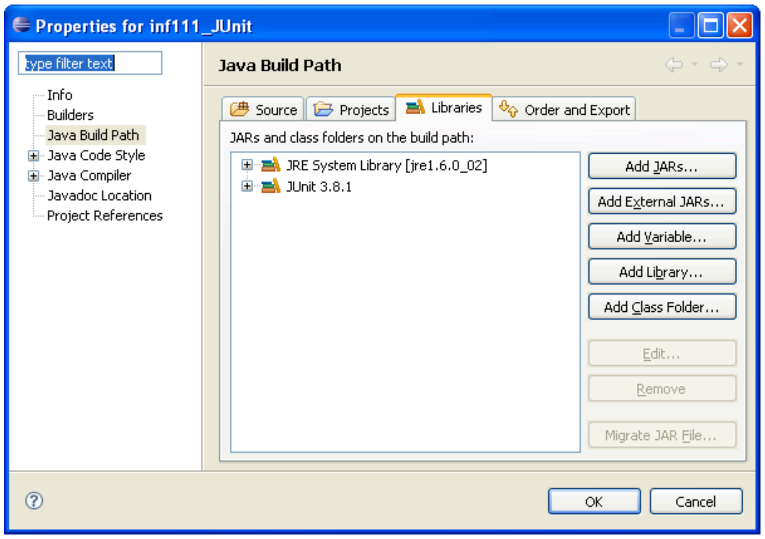
b) In the next screen, name the Test Suite "LunarLanderAllTests", and include your test cases created in the Task 1 and Task 3 (GUIEnvironmentTest and also ProcessorTest if you had time) into the test suite by selecting the check boxes in front of the appropriate classes. Then click Finish.

c) Run the test suite by selecting the Test Suite class, and then Right Click → Run → JUnit Test

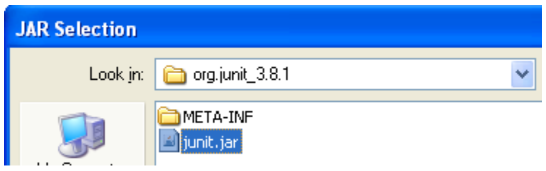
Appendix

You can also manually add the JUnit library, junit.jar, to the project's build path using the following steps.

a) Right click the project and select Properties. The properties dialog box as shown below should appear.



b) Select Java Build Path on the left panel and bring the Libraries tab forward.
c) Click on "Add External JARs…" button.
d) In the JAR Selection dialog box, find the plugins directory under Eclipse's installation directory (C:\Opt\eclipse). Then locate the JUnit folder. The current version of JUnit should be org.junit_3.8.1. From that directory, select junit.jar, as in the figure below.


JAR Selection
Look in:
org.junit_3.8.1
META-INF
junit.jar