Testing Methodology
Assignment 2
Unit testing using JUnit

COSC 565/475: Software Engineering I

Fall Semester 2015

Deadline: 24th of November

What is JUnit?

JUnit is a framework for testing parts, or units of your code. In general, these units are considered to be the methods of each class. JUnit can help you to make sure that each of your classes work as expected. In unit testing you will usually write one test class for each of the classes that you want to test. Your test class will often include a test method for each method implemented by the class being tested. But keep in mind that this is not always feasible, or necessary. A test case will, and should often touch on more than a single method. These tests should be written to make it likely that when all the tests pass, the code is functioning as required.

JUnit with Eclipse

- Run Eclipse IDE. We will create a new workplace project
    - so click File -> New -> Project,
    - then choose Java and click Next.
    - Type in a project name -- for example, ProjectWithJUnit. Click Finish.
    - The new project will be generated in your IDE. Let's configure our Eclipse IDE, so it will add the JUnit library to the build path.
    - Click on Project -> Properties, select Java Build Path, Libraries, click Add External JARs and browse to directory where your JUnit is stored. Pick *junit.jar* and click Open.
    - You will see that JUnit will appear on your screen in the list of libraries. By clicking Okay you will force Eclipse to rebuild all build paths.
    - To create such a test, right-click on the ProjectWithJUnit title,
    - select **New -> Other**, expand the "Java" selection, and choose **JUnit**.
    - On the right column of the dialog, choose **Test Case**, then click **Next**.

Example

```
public class HelloWorld {
        public String say() {       return("Hello World!");    }
}
```

```
import junit.framework.TestCase;
public class TestThatWeGetHelloWorldPrompt extends TestCase {
   public TestThatWeGetHelloWorldPrompt(
       String name) {
       super(name);
```

```
    }
    public void testSay() {
        HelloWorld hi = new HelloWorld();
        assertEquals("Hello World!", hi.say());
    }
    public static void main(String[] args) {
        junit.textui.TestRunner.run(
            TestThatWeGetHelloWorldPrompt.class);
    }
}
```

Basic Information

Before you begin, we would like to call your attention to the following conventions:

- A Test Case Class is named [classname]Test.java, where classname is the name of the class that is tested.
- A Test Case Method is a method of the Test Case Class which is used to test one or more of the methods of the target class. Test Case Methods are are annotated with @Test to indicate them to JUnit. Cases without @Test will not be noticed by JUnit.

JUnit assertions are used to assert that a condition must be true at some point in the test method. JUnit has many types of assertions. The following is a selection of the most commonly used assertions:

_ **assertEquals**(expected, actual): Assert that expected value is equal to the actual value. The expected and actual value can be of any type, for example integer, double, byte, string, char or any Java object. If the expected and actual values are of type double or oat, you should add a third parameter indicating the delta. It represents the maximum difference between expected and actual value for which both numbers are still considered equal.
_ **assertTrue**(condition): Asserts that the Boolean condition is True.
_ **assertFalse**(condition): Asserts that the Boolean condition is False.
_ **assertNull**(object): Asserts that an object is null.
_ **assertNotNull**(object): Asserts that an object is not null.
_ **assertSame**(expected object, actual object): Asserts that two variables refer to the same object.
_ **assertNotSame**(expected object, actual object): Asserts that two variables do not refer to the same object.

Whenever an assertion fails, an AssertionError is thrown, which is caught by the JUnit framework and presented as a red bar, indicating test failure. Assert statements accept an extra message parameter before the other parameters.

Loading the Project

Download the file vending.rar from blackboard

In Eclipse, choose File -> New -> Java Project. Give it a name (“Lab1", for instance) and click Finish.

Running the Test Cases

When you run a test class, JUnit will run each method annotated with @Test separately and show a green bar if all of them pass, and a red bar if any of them fail. It is important that anything happening in a test method is independent from the other test methods, otherwise you risk getting weird results.

Assignment Tasks

Money and Currency

- You have been given a template for the Currency and Money classes, with JavaDoc comments explaining what each method should do. There is also a Bank and Account class, but we will come back to that in the following section. All the methods of Currency.java and Money.java are empty.
- First, write test cases for the methods of each class, and then fill in the methods with code that will make your test cases pass. The template test methods are a guideline you can use for constructing your tests. Unless you have a good idea for how to otherwise structure the tests, you should simply fill in those template test methods. Bear in mind that the teaching assistants have their own test cases for these methods, so write some good tests yourself, to make it likely that your code will pass our tests as well.

Bank

The Bank and Account classes were written by a bad programmer. When you are confident your Money and Currency classes work as intended, write test cases for the Bank and Account classes and find the bugs. Again, the specification is provided in the JavaDoc comments.

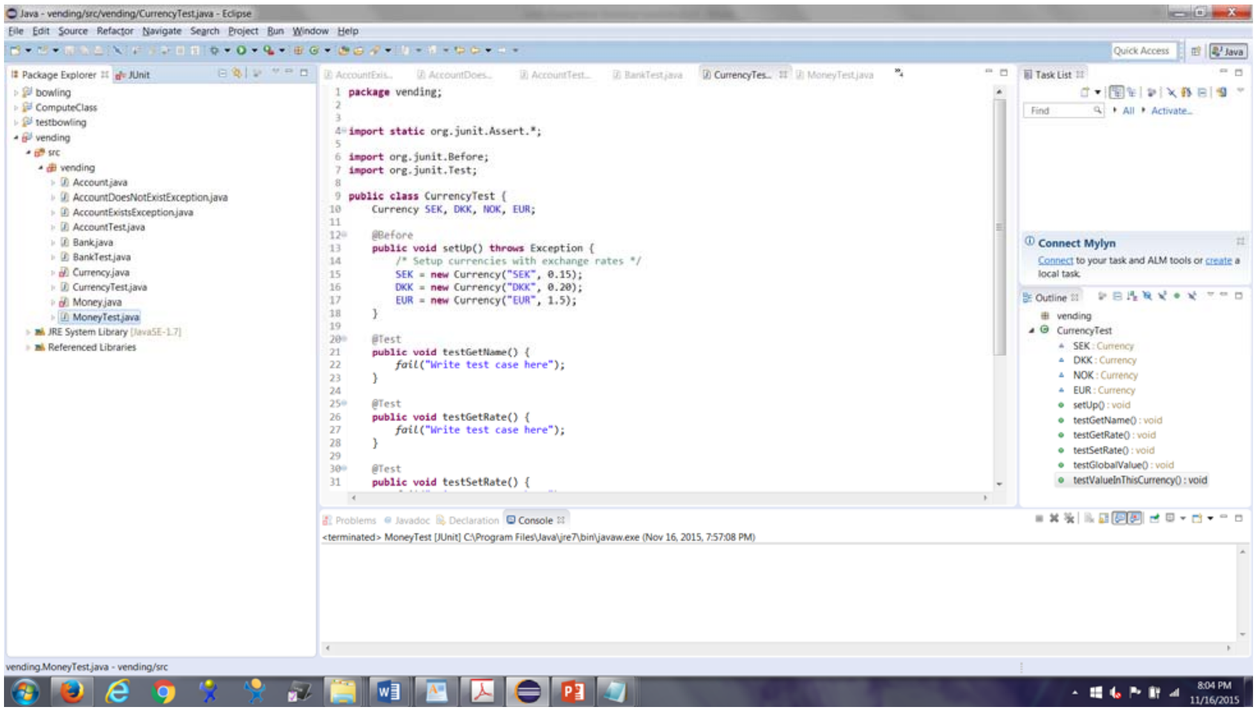
- Write the code for the body of the test case methods of MoneyTest.java and CurrencyTest.java.
- Complete the methods body in **Money.java and Currency.java** classes, and comment where necessary. Again make sure you follow the specification for each method and implement the required functionality.
- If you run your test cases for MoneyTest.java and CurrencyTest.java at this point, they should be all pass.
- By following the Bank.java and Account.java specification in the JavaDoc comments, write your test cases in the BankTest.java, CurrencyTest.java, MoneyTest.java, and AccountTest.java files. Note which of your test cases fail, by commenting in the corresponding test methods.
- Fix the bugs and verify that BankTest.java and AccountTest.java passes successfully.

Extra Credit: [submit a word file to identify extra credit questions]

- Uses **5 different assert statements and write 10 more unit test cases for** BankTest.java, CurrencyTest.java, MoneyTest.java, and AccountTest.java files. The most common assert statement is assertEquals() which takes at least two arguments. Comment the extra credit assert statements to identify the new test cases.
- Find the bugs in Bank.java and Account.java based on the JUnit failures from the previous step, and note it by placing a comment in the code where the bug was spotted. Explain how you found the bug with your unit tests.

NOTE

Working independently. Your answers to the questions on this assignment will be individually marked, and must be your own work. You will be assigned 0 marks for this entire assignment, if any of your answers to individual questions bears a close resemblance to another student's submission, or to something previously published on the internet or elsewhere



```java
package vending;


import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class CurrencyTest {
    Currency SEK, DKK, NOK, EUR;

    @Before
    public void setUp() throws Exception {
        /* Setup currencies with exchange rates */
        SEK = new Currency("SEK", 0.15);
        DKK = new Currency("DKK", 0.20);
        EUR = new Currency("EUR", 1.5);
    }

    @Test
    public void testGetName() {
        fail("Write test case here");
    }

    @Test
    public void testGetRate() {
        fail("Write test case here");
    }

    @Test
    public void testSetRate() {
```

```
<terminated> MoneyTest [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (Nov 16, 2015, 7:57:08 PM)
```

Screen Shot of the classes in Eclipse

Submission Instructions

- Archive your project folder in zip format and name it FirstName.LastNam.zip where the first name and last name refers to your name.
- Submit it on Blackboard.