BITS, Pilani – KK Birla Goa Campus
Department of CS&IS

CS/IS F213 Object Oriented Programming
CS/IS C313 Object Oriented Programming and Design

Lab-9 *Dt.29.10.2014*

The pre-lab work for the lab is as follows.

1. Read **Ch 2: Your First Unit Tests, Ch 3: Writing Tests in Junit, Ch 6: Using Mock Objects** from Pragmatic Unit Testing in Java with Junit, Andy Hunt and Dave Thomas, The Pragmatic BookShelf, 2003.
2. We will use graph algorithm code from http://algs4.cs.princeton.edu/41undirected/.
   a) Familiarize yourself with background concepts and code skeleton of the following Java classes.
   BreadthFirstPaths.java, Bag.java, Graph.java, SelfEdgeException.java, In.java, Queue.java *and* ***Stack.java***
   b) Also read the format of graph as specified in **tinyG.txt** file.

NOTE: Use JUnit3 for this lab. You can add JUnit3 by doing
Project --> Properties --> Java BuildPath --> Libraries --> Add Library --> JUnit --> JUnit3

The in-lab work for nineth lab is as follows.

Question1
You are given with the implementation of Single-Source Shortest Path algorithm for a graph via Breadth first Search traversal algorithm. You don't need to worry about the algorithm. You just need to look at the structure of the Java project for which you can refer the Java documents available in the folder **code_download**.
It includes following seven classes **BreadthFirstPaths.java, Bag.java, Graph.java, SelfEdgeException.java, In.java, Queue.java** and **Stack.java**. And one input test case **tinyG.txt** which you can use as input file to your code.

Build a JUnit test cases for following cases

A) *addEdge()* method is present in **Graph.java** class. This method is used to add the edge to the given graph. Before calling this method you need to generate graph. To generate graph, use the **tinyG.txt** input file. You can see the main method of **Graph.java** to generate the graph.

   Write test cases for the following situations:
   A.1) Try to add self edge (program should throw **SelfEdgeException**)
   A.2) Try to add edge whose vertex is not present in graph (program should throw **IndexOutOfBoundsException**)

B) *getAdjecencyList()* gives the adjacency list of the created graph. Write a test case to test the adjacency list. Consider the following conditions for the same:
   B.1) Check after graph construction

B.2) Add multiple edges
B.3) Check the adjacency list

C) Write a JUnit TestSuite to bundle unit test cases of the *getAdjecencyList()* and *addEdge()* and then execute it.

The post-lab work for nineth lab is as follows.

Question-1 (continued)

D) Write a parametric test case for *hasPathTo()* method of **BreadthFirstPaths.java**. The constructor *BreadthFirstPaths(Graph G, int s)* takes graph and source vertex as parameters. Write a parametric test case for *hasPathTo()* methd. The parametric test must iterate over graph in **tinyG.txt** and five different source vertices, namely 3, 4, 5, 10 and 12. You must check to see if there is a path from each of these five source vertices to vertices 4 and 5.

Question2

In a university, a student gets scholarship based on academic performance and family income. We are developing a system that will compute credits based on student CGPA and family income. Student scholarships shall be issued based on the computed credits. To get credits, a method *getOverallCredit()* has been defined which is not implemented as of now. But method *isEligibleForScholarship()* makes use of *getOverallCredit()* method. To test *isEligibleForScholarship()* method, we need to use mock objects so that *isEligibleForScholarship()* can be tested without actually implementing *getOverallCredit()* method. Code is provided in **mockObjects** package. Write a tester class named **ScholarshipServiceTest** to test the above scenario.

References

E-copy of Pragmatic Unit Testing available at http://www.infotest.by/documents/Pragmatic_unit_testing.pdf

Junit Tests
http://www.tutorialspoint.com/junit/
http://www.whiteboxtest.com/Unit-Testing-With-JUnit3.php
Parametric testing idea http://www.coderanch.com/t/95776/Testing/Parameterizing-test-cases

EasyMock (useful for Question 2)
http://easymock.org/