Assignment 3

1 Objective

Practice Test Driven Development

2 Marking

This assignment is worth 10% of your final grade.

The submission deadline is Feb 5, 2017, 11:59pm on Github.

You are encouraged to work in pairs.

Late submission policy: No late submissions are accepted.

3 How to submit your work

1. Please sign in to git using your `mail.utoronto.ca` account. Use this invitiation link

   `https://classroom.github.com/group-assignment-invitations/6cf671c1f138418175bc40957bffa5fb`

   and clone the starter code following `GitHub` instructions.
2. Make sure to add team members names and student IDs in the Readme.md.
3. To submit your work, add, commit and push your changes to your repository.

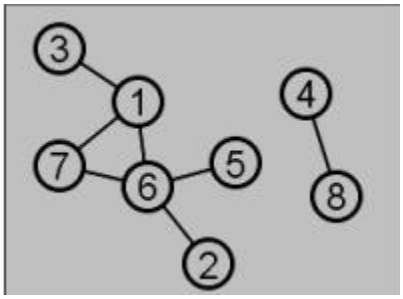
   Do **not** commit the files and directories generated by Eclipse, such as `bin, doc, .project`, etc. Marks will be deducted if you submit these. Feel free to update `.gitignore` as necessary.
4. The date and time of your last commit must be on or before 5 Feb 2017, 11:59pm.

4 `Graph` ADT

In this exercise, you will write an implementation of the `GraphInterface` called `Graph`. A graph is an ordered pair (V, E) where V is a set of vertices, and E is the set of edges.

Each vertex will contain a unique piece of information (which can belong to any Java type, therefore our ADT is generic).

You will work out your implementation so the provided unit tests are satisfied in their entirety.



5 Specifications for `Graph.java`

- The set of edges, denoted E simply contains connections between two vertices.
- Our graph is undirected, so if our graph contains the edge $(1, 6)$, it also contains the edge $(6, 1)$.
- If an edge already exists between two vertices, no additional edges can be added between them.
- A vertex cannot have an edge connecting to itself.
- A connected component of a graph is a subgraph such that for any pair of vertices that belong to the set of vertices of that subgraph, there exist a path connecting these two vertices.
- A graph can contain more than one connected component. For instance the graph shown in previous page contains two connected components.
- Your implementation of the `Graph` class must contain a method with the following signature:

```
public Set<T> DFSVisit(T startVertex)
// Uses depth-first search algorithm to visit as much vertices as
// possible starting from startVertex.
// In the process, keeps track of all vertices reachable from startVertex
// by adding them to a Set<T> variable.
// Once done, returns the Set<T> that we build up.
     // This is precisely the connected component that contains the vertex startVertex.
```

- The purpose of the method DFSVisit is to provide a tool that finds individual connected components of the graph.
- The method `DFSVisit` must use depth first search algorithm, which you can find in Wikipedia or other online resources. The Java code though must be entirely yours!
- In addition to the required methods, you may want to create helper methods at your will.
- Also, your implementation of the `Graph` class must contain a method `connectedComponents` with the following signature:

```
public ArrayList<Set<T>> connectedComponents()
// Returns a list of connected components of the graph
// For each vertex that does not belong to the connected components
// already on the list, call DFSVisit to obtain a set of vertices connected to the current vertex
// Add the set to the list
```

- You are free to implement the required classes your way, however you have to make sure that your code satisfies the provided unit tests.

6 Specifications for `GraphIsFullException.java` and `VertexExistsException.java`

Two exception classes `GraphIsFullException` and `VertexExistsException.java` must be checked exceptions, and need only two members: a no-argument constructor, and a one-argument constructor that takes a `String` message. Each constructor should simply call the corresponding constructor of the parent class.

7 Checklist

Have you...

- tested your code on the lab computers using **Java 1.8**?
- committed and pushed to your repo?
- tested your solution, made any necessary changes, and re-committed if necessary?