Homework #1: A Social Network

Due September 14th (Thursday) at 11:59 p.m.

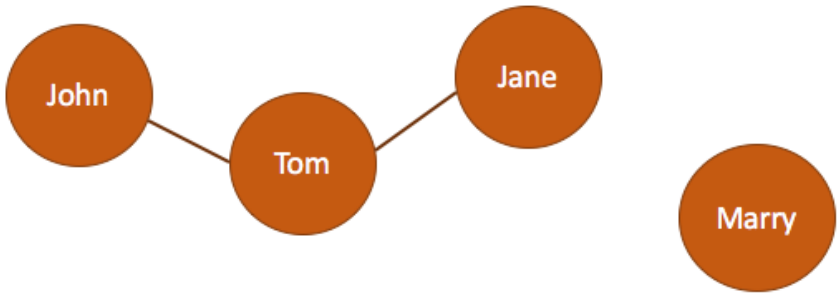
This is a warm-up homework assignment. The goal of this assignment is to remind yourself Java object-oriented programming. For this homework assignment, you will implement a simple graph class that represent a social network.

Instructions

Implement and test a `FriendGraph` class that represents friendships in a social network and can compute the distance between two people in the graph. You should model the friendship network as an undirected graph, where each person is connected to zero or more people.

Note that your class should be in package named cse4006. Thus it should be possible to import cse4006.FriendGraph or simply import cse4006 to use the class.

Suppose you have the following social network:



Your implementation of FriendGraph class (and Person class) should be able to represent the above social network in the following way:

```
FriendGraph graph = new FriendGraph();
Person john = new Person("John");
Person tom = new Person("Tom");
Person jane = new Person("Jane");
Person marry = new Person("Marry");
graph.addPerson(john);
graph.addPerson(tom);
```

```
graph.addPerson(jane);
graph.addPerson(marry);
graph.addFriendship("John", "Tom");
graph.addFriendship("Tom", "Jane");
System.out.println(graph.getDistance("John", "Tom")); // should print 1
System.out.println(graph.getDistance("John", "Jane")); // should print 2
System.out.println(graph.getDistance("John", "John")); // should print 0
System.out.println(graph.getDistance("Marry", "Marry")); // should print -1
```

Evaluation

For the evaluation, we will pay attention to the followings:

- Your graph should have a `getDistance` method which takes two names (as `Strings`) as arguments and returns the shortest distance (an `int`) between the people with those names, or `-1` if the two people are not connected.
- **Do not use any standard libraries, including `java.util.*`.**
- Use proper access modifiers (`public`, `private`, etc.) for your fields and methods. If a field/method can be private, it should be private.
- Do not use any `static` fields or methods except for the `main` method.
- Follow the Java code conventions, especially for naming and commenting. Hint: use **Ctrl + Shift + F** (in Eclipse, Intellij has similar functionality) to auto-format your code!
- Add short descriptive comments (`/** ... */`) to all `public` methods.

Additional hints/assumptions:

- For your implementation of `getDistance`, you may want to review BFS (breadth-first search).
- You may create any number of files to complete this task (for example, helper classes containing custom data structure implementations).
- You may assume that there is some arbitrary upper limit (for example, 50) on the number of nodes or connections to a node. (you can set this limit to be whatever you want.)
- You may assume that each person has a unique name.
- You may assume that there are no self-loops (connections from a person to himself) in the graph. But there may be loops other than self-loops, so you should be able to handle them.
- You may handle incorrect inputs however you want (it's okay to silently fail or crash, or raise an exception)

Submission Guidelines:

- You should submit your homework to the department gitlab repository. The details of the guidelines will be announced soon.
- Your submitted source code should work correctly with Java 8.
- Your submitted git repository should contain build.xml and should be buildable by ant build system (apache-ant, http://ant.apache.org/).
- Running ant should create hw0.jar file in the project root directory.
- Your implemented classes should be in package **cse4006**. That is, if hw0.jar is in classpath, it should be possible to import cse4006.FriendGraph and access the class.