

Homework #1: A Friendship Graph

Due Tuesday, January 20nd at 11:59 p.m.

The goals of this assignment are to familiarize you with our course infrastructure and to let you practice object-oriented programming with Java. To complete this homework you will implement a simple graph class that could represent a network of friends.

Your learning goals for this assignment are to:

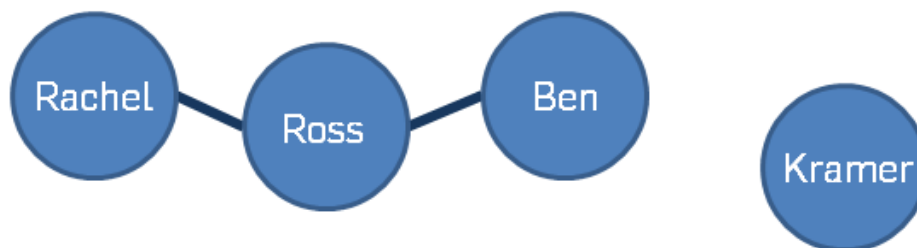
- Use Git and GitHub to revise and share (with course staff) your work.
- Become familiar with a Java development environment, Eclipse.
- Write a first Java program.
- Practice Java style and coding conventions, using Checkstyle to check and enforce the conventions.
- Practice using general build automation and testing tools, Gradle and Travis-CI.

Instructions

To begin your work, clone your course repository from GitHub and import the homework project into Eclipse from the `homework/1` directory in your repository.

Implement and test a `FriendshipGraph` class that represents friendships in a social network and can compute the distance between two people in the graph. You should model the social network as an *undirected* graph where each person is connected to zero or more people, but your underlying graph implementation should be *directed*.

For example, suppose you have the following social network:



Your solution must work with the following client implementation (located in `Main.java`).

```
FriendshipGraph graph = new FriendshipGraph();
Person rachel = new Person("Rachel");
Person ross = new Person("Ross");
Person ben = new Person("Ben");
Person kramer = new Person("Kramer");
graph.addVertex(rachel);
graph.addVertex(ross);
graph.addVertex(ben);
graph.addVertex(kramer);
graph.addEdge(rachel, ross);
graph.addEdge(ross, rachel);
graph.addEdge(ross, ben);
graph.addEdge(ben, ross);
System.out.println(graph.getDistance(rachel, ross)); //should print 1
System.out.println(graph.getDistance(rachel, ben)); //should print 2
System.out.println(graph.getDistance(rachel, rachel)); //should print 0
System.out.println(graph.getDistance(rachel, kramer)); //should print -1
```

Evaluation

Overall this homework is worth 50 points. To earn full credit you must follow these requirements:

- Your solution should work with the client code above. The `getDistance` method should take two people (as `Persons`) as arguments and return the shortest distance (an `int`) between the people, or `-1` if the two people are not connected.
- Your graph implementation should be reasonably scalable. We will test your graph with several thousand vertices and edges.
- Your solution must build on `travis-ci.com` using our Gradle and Checkstyle build configuration.
- 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 `static` fields or methods except for the `main` method(s) and constants.¹ Note that we will not penalize the appropriate use of `static` in constants.

¹See [here](#) for an example on appropriate usage for constants.

- Follow the **Java code conventions**, especially for **naming** and **commenting**. Hint: use **Ctrl + Shift + F** 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 **breadth-first search**.
- You may use the standard Java libraries, including classes from `java.util`, but no third-party libraries.
- In addition to being compatible with our client code, you may create any number of files, classes, or methods to complete this task. You may generalize your `FriendshipGraph` implementation to work with classes other than `Person`, but this feature is not necessary for this assignment.
- You may assume that each person has a unique name.
- You may handle incorrect inputs however you want (printing to standard out/error, silently failing, crashing, throwing a special exception, etc.)
- Some of the Java conventions will be enforced mechanically by Checkstyle during a build with Gradle (e.g., on Travis-CI). Passing the Checkstyle check is necessary, but not sufficient.
- You should write additional samples to test your graph, similar to our main method. You may modify the Gradle configuration to run your code automatically on Travis-CI, but that is not required.
- To print something to standard out, use `System.out.println`. For example:

```
System.out.println("DON'T PANIC");
```

We will grade your work approximately as follows:

- Submitting a compiling solution: 20 points
- Working `getDistance` implementation: 10 points
- Fulfilling technical requirements (the bullets directly under “Evaluation”): 10 points
- Documentation and code style: 10 points