EECS 293

Software Craftsmanship

2014 Fall Semester

Programming Assignment 4

Due at your recitation session on September 22-26, 2014

Reading

Read Chapter 5 in the textbook.

Grading Guidelines

Starting with this assignment, an automatic C (or less) is triggered by

- Any routine with complexity greater than 4 or by
- Any substantially repeated piece of code.

Public Key

Your public key will be used for homework submission later on in the semester. Follow the instructions on blackboard (Course Documents: Cryptographic Keys) to create, submit, and verify your public key.

Programming

First, make any changes that were discussed in your recitation session, and refactor the code to avoid the penalty above.

The class `Friend` keeps information on (another) user and its distance, as the number of Links needed to reach him. It supports the methods:

- `Friend()` creates an invalid friend. It throws no exception.
- `boolean set(User user, int distance)` sets a friend at the given distance and marks the friend as valid. If this `Friend` had already been associated with a user and distance (that is, if the `Friend` was already valid), it does nothing and returns false, otherwise it returns true.
- `User getUser()` returns the `User` that is recorded in this `Friend`. If this `Friend` is invalid, throw a `UninitializedObjectException.`
- `int getDistance()` returns the distance to this friend. If this `Friend` is invalid, throw a `UninitializedObjectException.`

- `String toString()` returns a human readable code for this `Friend`. If the `Friend` is invalid (and only if the `Friend` is invalid), the method should return the `String` "Invalid Friend".

These methods throw a `NullPointerException` if any argument is null.

The class `SocialNetworkStatus` has one more status code called `INVALID_DISTANCE` and the methods:

- `Set<Friend> neighborhood(String id, Date date, SocialNetworkStatus status)` returns all of the `Users` throughout the social network that are directly or indirectly associated with the user that has the given id, along with the distance from the initial user. The starting point should be reported as a friend at distance zero.
  - If the user id is absent, set the status to `INVALID_USERS`.
  - In case of success, set the status to `SUCCESS`.
- `Set<Friend> neighborhood(String id, Date date, int distance_max, SocialNetworkStatus status)` returns all of the `Users` throughout the social network that are directly or indirectly associated with the user that has the given id, along with the distance from the initial user. The starting point should be reported as a friend at distance zero. Only friends at distance of `distance_max` or less should be reported.
  - If the user id is absent, set the status to `INVALID_USERS`.
  - If the distance is negative, set the status to `INVALID_DISTANCE`.
  - In case of success, set the status to `SUCCESS`.

This method throws a `NullPointerException` if any argument is null.

As usual, these classes may contain as many auxiliary private methods as you see fit, and additional helper classes may be defined. Your classes must be documented in JavaDoc. Test cases must be created in JUnit to test all its public methods.

Discussion Guidelines

The class discussion will focus on:

- High-level code organization
- The design and documentation of your project
- Functions that exceed McCabe's complexity of 4 (if any)
- Non-structured programming constructs and break statements (if any)

On-Line Submission

Submission is due at the beginning of your recitation session. Turn in with your code:

- A README file explaining how to compile and run the code.
- A Make or Ant build file.

- A comment at the top over file containing your name, email address, a one-sentence description of the file, and if necessary a longer comment describing the design of the file.

The code should be handed in a zip, tar.bz2, or tar.gz archive. Archives in 7z cannot be accepted.