Programming Assignment #3
Six Degrees of Kevin Bacon

Problem Statement

"Six Degrees of Kevin Bacon is a trivia game based on the concept of the small world phenomenon and rests on the assumption that any individual involved in the Hollywood, California film industry can be linked through his or her film roles to actor Kevin Bacon within six steps."[1] The object of this game is to link a movie actor to Kevin Bacon via shared movie roles (called collaboration links). The *Bacon number* of an actor is the degrees of separation he or she has from Kevin Bacon. For example, Tom Hanks has a Bacon number of one, as he was in `Apollo 13` with Kevin Bacon. Sally Field has a Bacon number of two, because she was in `Forrest Gump` with Tom Hanks. This game is in fact a graph problem and a simple form of social network analysis.[2]

For this programming assignment, we design a Java program that allows us to explore complex co-star relationships among actors and carry out a variety of analyses on the Internet Movie Database.[3] For example, we may be interested in finding out how influential an actor is in the movie industry (or, in general, how influential a person is within a social network). The more influential (or central) an actor is, the lower his/her total or maximum distance to all the other actors is. As another example, a person is often said to be *between* two people if the person is on a shortest path between them. The *betweenness centrality* is a measure of the control a person has on the communication between others. In order to make these analyses feasible, we need be able to compute some critical measures such as the shortest distance and the number of distinct shortest paths for any pair of people in a social network.

Requirements & Implementation

- You are to write a Java class that implements a Movies-Actors database called *MovieNet*. Its skeleton file is provided in the eTL site. It is named `MovieNetskel.java`. You can copy the file to your working directory and rename to `MovieNet.java`. You should then provide implementations of its constructor(s) and all the methods defined in the class skeleton. You can add more methods as you wish, but are not allowed to remove any method defined in the class skeleton.

- The `MovieNet` database supports nine different types of queries for analyzing movie-actor and actor-actor relationship by providing a method for each query type. The nine query types (and the requirements of their methods) are listed below. Implementation of the methods must meet the requirements.

    1. The `moviesby(actors)` method returns a list of movies that were co-starred by the actors or `null` if there is no such a movie.
    2. The `castin(movies)` method returns a list of actors who were cast in all the movies or `null` if there is no such an actor.
    3. The `pairmost(actors)` method returns a pair of actors who starred together most often or `null` if there is no such a pair.
    4. The `Bacon(actor)` method returns the Bacon number of the actor. It returns `-1` if the actor is not reachable from Kevin Bacon, the actor is not found, or Kevin Bacon himself is not found in the MovieNet database. Two actors are said to be *reachable* from each other if they can be connected via one or more collaboration links.
    5. The `distance(actor1,actor2)` method returns the shortest collaboration distance between the two actors. It returns `-1` if the two actors are not reachable from each other, or either of the actors is not found in the MovieNet database.

[1] `http://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon`

[2] If you are not familiar with this trivia game, watch these videos at `youtube`: `http://www.youtube.com/watch?v=afJRKbBEr2Q`, and `http://www.youtube.com/watch?v=vOUdy2J0lfU`.

[3] `http://www.imdb.com`

6. The `npath(actor1,actor2)` method returns the number of distinct shortest collaboration paths between the two actors. It returns `0` if there exists no path between them (*i.e.*, they are not reachable from each other) or either of the actors is not found in the MovieNet database.
7. The `apath(actor1,actor2)` method returns a shortest collaboration path between the two actors. It returns `null` if there exists no path between them (*i.e.*, they are not reachable from each other) or either of the actors is not found in the MovieNet database. Recall that more than one shortest path may exist between two actors.
8. The `eccentricity(actor)` method returns the eccentricity of the actor. The eccentricity is the maximum of all shortest collaboration distances from the actor to the other reachable actors.
9. The `closeness(actor)` method returns Dangalchev's closeness centrality of the actor. Dangalchev's closeness centrality of an actor s is defined by the following formula:

$$closeness(s) = \sum_{s \neq t} \frac{1}{2^{d(s,t)}}$$

   where $d(s,t)$ is the shortest collaboration distance between actors s and t. If t is not reachable from s, then $d(s,t) = \infty$.

- Follow the directions given in the first assignment handout regarding 'no `main()` method.'

Grading

This assignment is worth 15 percent of your final grade. In writing the code, correctness is the most important attribute, followed by efficiency. General grading guidelines are:

10 points	Program compiles without errors and is on the right track,
0-30 points	Query types 1, 2 and 3 work on simple and complex test cases,
0-15 points	Query types 4 and 5 work on simple and complex test cases,
0-15 points	Query type 6 works on simple and complex test cases,
0-15 points	Query type 7 works on simple and complex test cases,
0-15 points	Query types 8 and 9 work on simple and complex test cases.

To evaluate the efficiency of your class implementation, we will measure the time spent on the `MovieNet` database construction and query processing. If it is more than 10, 100, or 1000 times slower than my own implementation, you will lose, respectively, 10%, 25%, or 50% of the credits reserved for correctness.

The late submission and regrading policies are described in the course syllabus.

Submission

Refer to the first programming assignment for submission instructions.

Due date

The programming assignment is handed out on Thursday Oct. 30, 2014, and due by 11pm on Wednesday Nov. 26, 2014.