Assignment 6

CMSC204

Myers

New concepts tested by this program

Implement Graph Interface

Use Graph to maintain a network of Actors

Shortest Path Algorithm implementation

Social networks are everywhere! They connect people with other people. There is an internet meme called “The Six Degrees of Kevin Bacon”, which describes how movie actors are connected, directly or indirectly, with Kevin Bacon. Surprisingly, very short chains exist between any pair of actors, even obscure ones. For more information, see [Six Degrees of Kevin Bacon](https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon) and [The Oracle of Kevin Bacon](http://www.cs.virginia.edu/oracle/).

A Graph is a set of Vertices and a Set of Edges. You will creating an ActorGraph that will be implementing a Graph interface with a set of Vertices of Actors and a set of Edges that identify the connection between two actors with a movie name. The header for your ActorGraph will be as follows: public class ActorGraph implements Graph<Actor, MovieEdge>.

Within the Graph interface is a method dijkstraShortestPath. You will be coding the Dijkstra’s Shortest Path algorithm. You will then be able to find the connections between two actors through the movies that they have acted in.

For example, to connect Robin Williams to Charles Chaplin we might find:

**Robin Williams via “Dead Poets Society” to Norman Lloyd**

**Norman Lloyd via “Limelight” to Charles Chaplin**.

The ShortestPath algorithm typically uses a weighted graph which means that the edges have a weight, and this is used to determine the shortest path. For this implementation, each weight will be 1, so that the shortest path is based on the number of movies in the path.

**Data Element – Actor (Vertices)**

You will be creating a class Actor which will hold all the information for an actor and implements Comparable. For simplicity sake, we will store the name in a single String in the following format: Last name, First name, so they can be sorted easily by last name. For the JUnit test, you will need a constructor of the Actor class that takes in a string, which is the name of the actor in the format: Last name, First name. You will also need a getName method to return the name of the actor in the format: Last name, First name. Follow the Javadoc for Actor.

**Data Element – MovieEdge (Edges)**

You will be creating a class MovieEdge which will hold all the information for an edge and implements Comparable. It contains a source and a destination, both are Actor objects, a weight (use 1 for the movie shortest path), and a String which holds the movie name that these two actors are connected by. It must have methods of getSource() and getDestination() which return the Actor objects and getMovieName() which returns a String with the movie name for the JUnit Tests. Follow the Javadoc html for MovieEdge.

**Data Structure – ActorGraph**

You will be creating a class ActorGraph which implements GraphInterface. The header for this class will be: public class ActorGraph implements GraphInterface<Actor, MovieEdge>. Implement the methods. This is an undirected graph, which mean that an edge from Actor1 to Actor2 also means there is an edge created from Actor2 to Actor1.

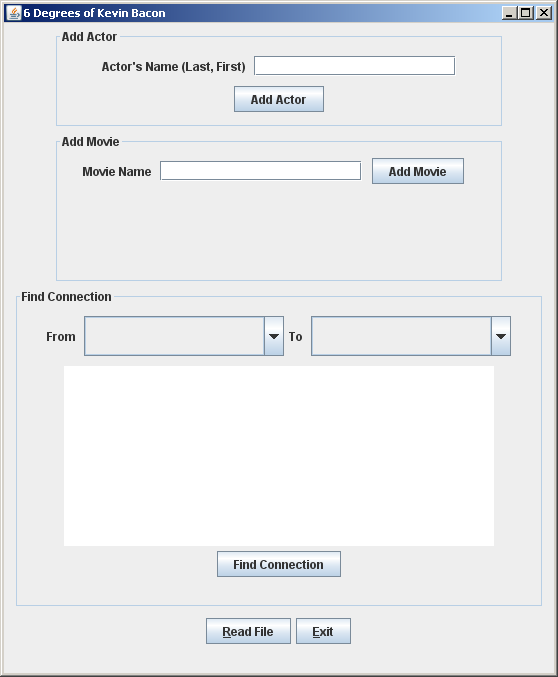
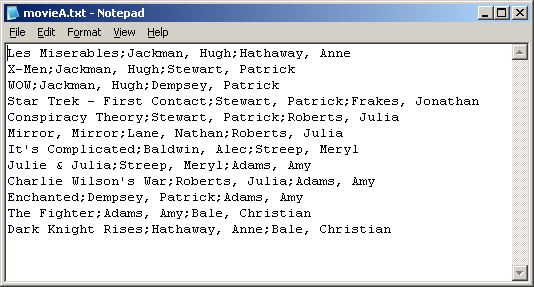
**Data Manager - ActorGraphManager**

Implement the ActorGraphManagerInterface that holds an object of the ActorGraph data structure and has methods for maintaining the data structure. The ActorGraphManager takes input from the GUI and provides output to the GUI through methods such as addActor and addMovie.

**GUI Driver**

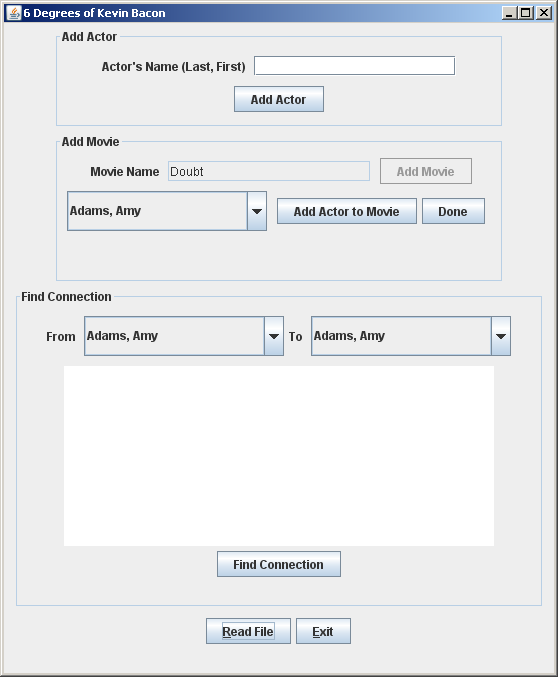
Create a GUI driver that will allow the user to read from a file to add Actors and Movies, or add Actors and Movies by typing in names. It will then display the connections between Actors. Use a FileChooser to allow the user to select a file to read from.

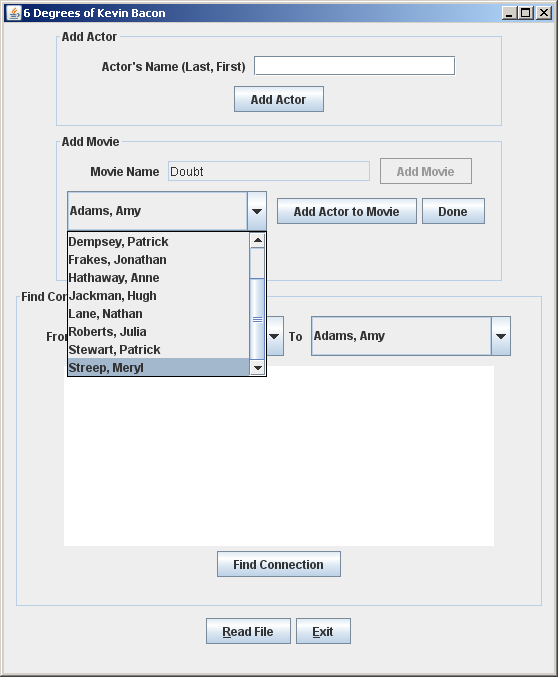
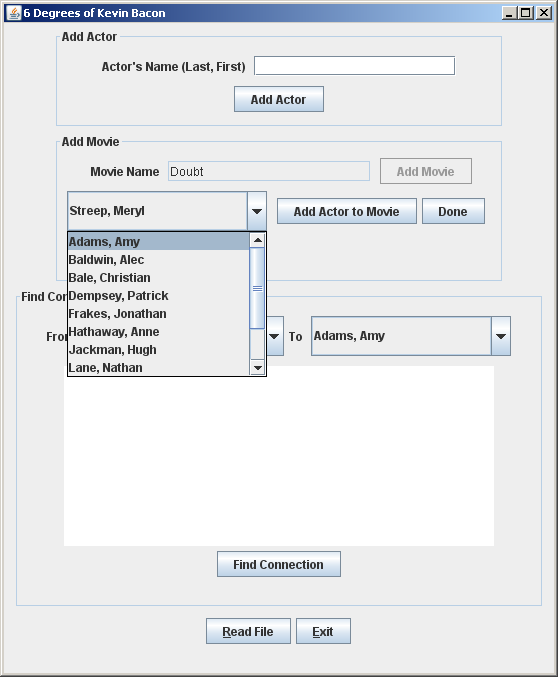
Beginning screen. First read a file.

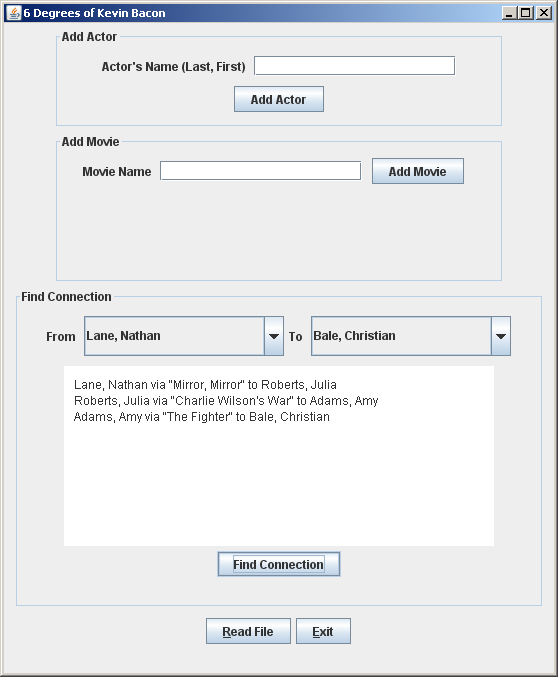
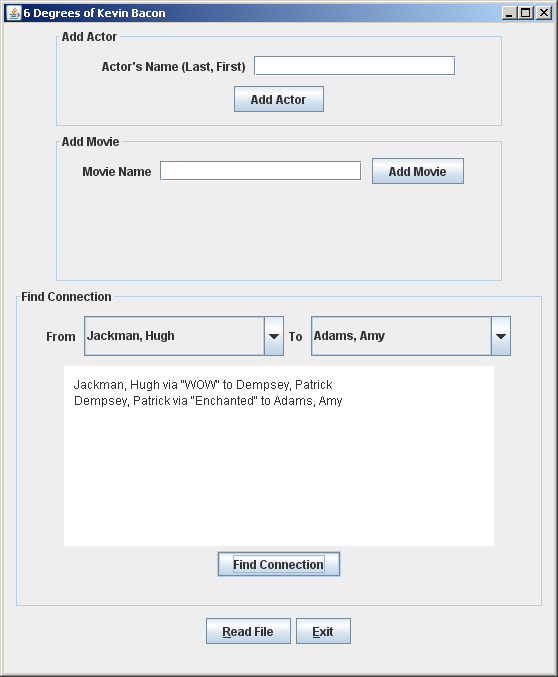
Delimiter of “;”

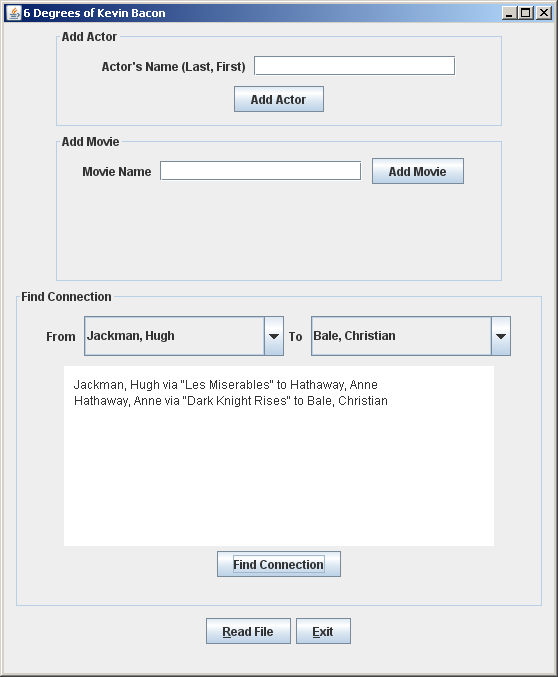
Add a movie by entering the movie name and the actors:



Find a connection between actors. Select two actors and the path is displayed in the text area.



Deliverables:

Java files - The src folder with your data structure, data elements, data manager and Junit Test (.java) files

Javadoc files - The doc folder with your javadoc for student generated files

UML Class Diagram (an image, not the proprietary format, must be a .jpg or .pdf)

Deliverable format: The above deliverables will be packaged as follows. Two compressed files in the following formats:

LastNameFirstName\_AssignmentX\_**Complete**.zip [a compressed file containing the following]

                        UML.jpg

                        doc [a directory] *include the entire student*-*generated javadoc folder*

                                    file1.html (example)

                                    file2.html (example)

                        src [a directory] *contains your .java and Junit Test (.java) files*

                                    File1.java (example)

                                    File2.java (example)

                                    File\_Test.java (example)

LastNameFirstName\_AssignmentX\_**Moss**.zip (a compressed file containing only the following: *contains your .java files – NO FOLDERS!!)*

                        File1.java (example)

                        File2.java (example)

**Program Grade Sheet**

**Assignment #6**

**CMSC204**

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Blackboard Date/Time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **DOCUMENTATION** (20 pts)

# CheckList for Assignment 6 is included and completed 1 pt \_\_\_\_\_

# Javadoc generated for all student created classes: 4 pts \_\_\_\_\_

JUnit Test Class 6 pts \_\_\_\_\_

Implement the STUDENT methods of DataManagerTest

Create a JUnit test for your FriendGraph - FriendGraphTest

UML Diagram 4 pts \_\_\_\_\_

Lessons Learned 5 pt \_\_\_\_\_

In 3+ paragraphs, highlight your lessons learned and learning experience from working on this project.

How did you do? What have you learned? What did you struggle with? How will you approach your

next project differently?

## **PROGRAMMING** (80 pts)

Programming Style and internal class documentation (within source code) 6 pts \_\_\_\_\_

Class description using Javadoc

Author’s Name, Class, Class Time, @author

Methods commented using Javadoc, @param, @return

Accuracy

Public tests 10 pts \_\_\_\_\_

Your tests (student tests) 8 pts \_\_\_\_\_

Private tests 10 pts \_\_\_\_\_

Program Details

1. Actor 8 pts \_\_\_\_\_
   1. holds the name (LastName, FirstName)
   2. implements the standard methods - constructors, getters/setters, toString, equals, etc.
   3. implements the Comparable interface
2. MovieEdge 8 pt \_\_\_\_\_
   1. implements Comparable
   2. stores references to both Actor objects, weight (1) and movie name
   3. implements the standard methods - constructors, getters/setters, toString, equals, etc
3. ActorGraph 10 pt \_\_\_\_\_
   1. implements GraphInterface
   2. use an adjacent matrix or adjacency list to store graph
4. ActorGraphManager 10 pt \_\_\_\_\_
   1. Implements ActorGraphManagerInterface
   2. Contains an ActorGraph
5. GUI details 10 pt \_\_\_\_\_
   1. Use FileChooser to select file
   2. User can add a movie or/and an actor.
   3. User can choose from list of available actors (alpha order, no duplicates)
   4. Actor list is updated when adding an actor

Total 100 pts \_\_\_\_\_