12 Finding Your Way.

Introduction

The goal of this assignment is to introduce you to the design of larger programs using the Model-View-Controller design pattern. The model for your program is designed to find a path between two nodes of a graph. The view can be display of the path in the Canvas or a description of the path in a text form in the console, or even an animation of the path search process. We give you fairly detailed specifications for the model part of the program. The requirements for the view are minimal. They only describe the basic functionality — you are free to enhance your presentation as you wish — and will be given credit for the work.

One third of the credit for this assignment will be given for a design document that describes the data, the organization of the program, the key program components, and the design of tests. Imagine you want someone to keep improving your program — provide a road map that explains what your program does and how does it do it.

One third of the credit for this assignment is for the model part.

The last third of the credit is for the view — grading both the design of the user views and the design of the program that drives it. A small bonus may be earned for exceptionally well designed display or interactions.

12.1 Project Review

The assignment is due on Tuesday, April 18th at 8:00 pm. During the day on the 18th (and possibly spilling over to Wednesday) I will meet with every student for 10 minutes — to review your portfolio (either electronic or a paper copy) and to view the project as it is ready at that time.

The Model

Graph

Your program needs to represent a graph with nodes. Each node has a name — using strings of no more than two characters makes the display easier, but other choices are fine. Each edge represents a bi-directional connection. If you can get from A to B at the cost 10, then you also can get from B to A at the same cost. Each edge has a numerical (integer) value

that represents the cost of traversing along that edge. You can think of it as representing the distance, or the cost of flight between cities, etc.

Algorithms

Your model should implement three graph traversal algorithms:

- Depth-First Seaarch
- Breadth-First Search
- Shortest Path Search

To implement the shortest path you need to represent a priority queue. You may design the heap algorithm discusses in the class, but it is perfectly fine (and desirable) to use one of the Java Collections Framework classes to do the job.

Using Libraries

Furthermore, throughout the project you are encouraged to leverage as much as possible from the existing Java libraries (both the Java Collections Framework, and the JPT libraries). The designer should focus on the design of interfaces between tasks, between components, wrapper and adopter class that allow you to use an existing library class in a customized setting.

The View

The requirements

The view at the minimum should have the following functionality:

- User should be able to see a representation of the graph.

  This can be a graphical display, a text that lists the nodes and the edges (with their weights), a graphcal display of the text that lists the nodes and the edges.

- User should be able to select which of the three algorithms is to be used for the subsequent task.
- User should be able to specify the origin and the destination of the desired path.

- The user should be able to see the resulting path.

  Again, this may be a graphical display, or just a text listing the nodes along the path.

The frills

Of course, the view can be much more elaborate. Here is a list of possible enhancements:

- Highlight the path is a different color in the graphics display.
- Display the steps in the search by highlighting in a different color the *s*een nodes, the *f*ringe nodes, the *o*rigin, the *t*arget, and the *u*nseen nodes. Animate the process using either the timer, or a user advance triggered by a key press.
- Animate the reconstruction of the path by traversing from the found target back to the previous node, all the way up to the origin.
- Select the origin and the target in a graphics display using a mouse.
- Display in a GUI the path length and possibly the nodes along the path.
- Choose the algorithm through a GUI.
- Make the graphics look like a game — e.g. traversing a street map or a maze.

12.2 The Advice

The design part of each project typically takes the greatest amount of time. the more time you spend thinking things through, the easier it is to actually write the code.

Make sure you think the whole framework through before you start programming. Spend some time researching the Java libraries to see what tasks can be done using the existing tools. Write sample adapters to see how the existing class can be used in your setting.

For example, in our soritng assignment the Traversal interface allowed us to supply the data to the algorithm in a number of different ways — and

allowed us to the produce the result in a universally readable manner as well.

The desing the key component by specifying their interfaces — the method headers, the interfaces that various classes must implement or use to get information from others.

For now, you have not learned about various tools and techniques to support such design process — other than class diagrams. Any description that you find helpful in clarifying the roles of the different classes and interfaces in your program is acceptable.

The design document you produce (which could be primarily in the form of javadocs) should describe all data definitions and the key methods, as well as give a general overview of the project organization.

Enjoy