CSE 331 Software Design and Implementation

Homework 8 *Campus Paths GUI*

Due: Thursday, June 6 @ **11pm**

Handout H8

Contents:

Introduction

In the previous assignment, UW Marketing commissioned you to write a route finder tool. Marketing is pleased with your initial results, but now they've asked for a graphical user interface (GUI) that visually draws routes on a map.

You will build your GUI using Java's Swing and Abstract Windowing Toolkit (AWT) libraries. In completing this assignment, you will get practice using Swing, event-driven programming, and the MVC design pattern.

When you're done, you can share your finished work with family and friends by following the instructions for creating a JAR file.

You are expected to fix any bugs from Homework 7 that affect the correctness or performance of your application in Homework 8. Further, your Homework 8 should use the model you created in Homework 7. This may require that you modify your Homework 7 code, that is ok, but be sure that Homework 7 continues to pass all tests.

Note that this assignment, being the last one, is due on a **Thursday** to give you some extra time.

GUI Requirements

You will write a GUI and a main class to launch it named `CampusPathsMain.java`. This assignment is deliberately open-ended: the exact appearance and functionality of your GUI are up to you. The only requirements are documented below.

For the most part, we are not grading on aesthetics: it doesn't matter whether your GUI looks pretty as long as it implements the required features. Nevertheless, a design which is genuinely confusing or hard to use (at our discretion) will not receive full credit. For example, we will deduct points if we can't easily figure out how to select the two buildings, if it's hard to see the selected path, or if we can only see the whole GUI on your 27-inch monitor. In addition, your program should be generally responsive: for instance, the GUI should not take an unusually long time to find and display paths.

Your GUI is a new View and Controller for your CampusPaths application. Ideally, you should not have to make any changes to your Homework 7 model classes - they already implement all the Model functionality that you need. If you have to make any small changes (for instance, if your design in HW7 was poor and some model methods were too closely tied to your text view), then you may do so. As always, all tests from previous homeworks must continue to pass, so you may also need to change your HW7 View and Controller in that case. In file `hw8/model-changes.txt`, list any changes you made to the model. For each, write a 1-2 sentence explanation of why the change was necessary and what you could have done differently on HW7 to create a more general and reusable Model. If you made no changes, write "None" for this section.

Window size

At startup, your GUI must fit and be usable on a screen with a 1024 x 768 resolution. (You may wish to support resizing the window, but doing so is not required.) Most computers provide a way to change the screen resolution, which you can use for testing.

Required features

At a minimum, your GUI must provide the following features:

- Load the map data from `src/hw7/data/campus_buildings.dat` and `src/hw7/data/campus_paths.dat` on startup. (This should be in your model, not your view. There is no need to duplicate files; load them from the `hw7` directory.)
- Display the map of campus provided in `src/hw7/data/campus_map.jpg`.
- Allow the user to select two buildings for finding a route. You might use free-text entry, dropdown menus, selection with the mouse, or some other approach.
- Mark the selected buildings and/or path endpoints on the map.
- Draw the shortest route between the selected buildings on the map. Ideally the map should automatically be zoomed in or out when a route is drawn, so that the route is almost as large as possible while still fitting in the window, but this is not required.
- Allow the user to reset the GUI by clicking a reset button. This button should clear all markings on the map and all other controls (such as building selectors), setting the GUI back to its initial state.
- Operate robustly: No matter what the user does, your program should never allow an exception message to bubble up to the console window, and your GUI should never crash or reach a buggy/invalid state.

Swing widgets and GUI builders

Use only components from the Swing widget set for this assignment. You are also allowed to use components from AWT, on which Swing is built, but should try to stick to Swing when possible.

Some IDEs, such as NetBeans, will let you specify the appearance and behavior of your GUI and automatically generate the code for you. You may not use these tools; you must write your GUI from scratch in Java.

Launching your GUI

Because you probably didn't write automated unit tests for your GUI, `ant validate` won't be useful for this assignment. Instead, you can manually launch your GUI from the command line by running the following commands:

```
   cd [PathToYourCse331Project]/cse331/src/hw8
   ant build
   cd ../..
   java -cp bin hw8.CampusPathsMain
```

Those are the commands we will run when grading your program.

If you log into attu via ssh, but when you try to run your GUI on attu you get a message similar to

```
Exception in thread "AWT-EventQueue-0" java.awt.HeadlessException:
No X11 DISPLAY variable was set, but this program performed an operation which requires it.
```

then you need to enable X11 forwarding for your ssh session. If you are are using Linux, this can be done using the `-X` switch when starting ssh:

```
ssh -X YourUserName@attu.cs.washington.edu
```

If you normally use Windows, the most straightforward option is simply to log onto one of the Linux workstations in the basement labs. Alternatively, you can obtain special software to enable X11 forwarding on Windows. One free option is Xming used with PuTTY. After installing Xming, launch PuTTY, select "Enable X11 forwarding" under Connection >> SSH >> X11, and log in as normal. Please note that the course staff does not officially provide support for enabling X11 on Windows.

Optional Extra Features

We encourage you to get creative with both the appearance and functionality of your GUI! If you do, **make sure to get a basic GUI working and commit it to your repository** before experimenting. Swing can be finicky, and seemingly simple UI improvements often become big time sinks. If you've committed your working solution, you can always "roll back" to that revision later.

If you want an extra challenge, you are encouraged to additional features. Here are a few ideas:

- As the window is resized, make the map shrink or grow to fit the window.
- Maintain the proportions of the map so that it zooms in on a route without becoming distorted.
- Place the map in a `ScrollPane` so it can be displayed full-size. When displaying a route or buildings, jump to that spot on the map and resize if needed. (Hint: you probably need to override `getPreferredSize()` in your "canvas" class for scrolling to work.)
- Allow the user to select the endpoints of a path by clicking the mouse on the map.

- Allow the user to drag the map with the mouse to change the portion that is shown.
- Add zoom buttons, possibly with a way to recenter the image for zooming if a hand is not available to drag it (e.g. mouse double click).

List any additional features you implemented, if any, in `hw8/extra.txt`. You must commit a `hw8/extra.txt` file, even if it simply contains "None". Additional features must not mask or replace any of the required features.

Small amounts of extra credit may be awarded for interesting or useful functionality that shows considerable effort. Extra credit will not be awarded for pure "eye candy," such as changing the font and color of buttons or adding a border. Extra credit will be applied to final grades only after we have ranked the students in the class and assigned a GPA; thus, it will not lower your grade if you choose not to work on optional features, but may raise your grade slightly if you do.

Writing automated tests for GUIs is difficult and usually involves special frameworks that are beyond the scope of this course. For this reason, you are not required to write unit tests or a test driver. We will test your solution by running your main program.

We may demo some of the most impressive GUIs on the last day of lecture. If you would like your GUI to be considered, please email cse331-staff@cs.washington.edu by 11pm Thursday June 6 (the non-late-day deadline). In your email, state your CSENetID and whether you prefer to be anonymous or credited.

Reflection [1 point]

Please answer the following questions in a file named `reflection.txt` in your `answers/` directory. Answer briefly, but in enough detail to help you improve your own practice via introspection and to enable the course staff to improve CSE 331 in the future.

a. In retrospect, what could you have done better to reduce the time you spent solving this assignment?
b. What could the CSE 331 staff have done better to improve your learning experience in this assignment?
c. What do you know now that you wish you had known before beginning the assignment?

Collaboration [1 point]

Please answer the following questions in a file named `collaboration.txt` in your `answers/` directory.

The standard collaboration policy applies to this assignment.

State whether or not you collaborated with other students. If you did collaborate with other students, state their names and a brief description of how you collaborated.

Time Spent

Tell us how long you spent on this homework via this catalyst survey:

https://catalyst.uw.edu/webq/survey/mernst/198080.

Returnin

There is no returnin for this assignment.

Hints

General GUI Advice

If you've never used Swing, it is well worth your time to do some tutorials, read the example code, and generally get comfortable with GUI programming before diving into the assignment. Take it from your peers: in the past, many students who choose the latter approach have reported to us afterward that they wish they had chosen the former.

Abstraction functions, representation invariants, and checkRep() are not required for GUI classes because they generally do not represent ADTs.

User testing is a great way to verify that your interface is as easy to use as you think it is. Show your GUI to your friend/roommate/sibling. Can they can figure out how to use it without directions from you? Also remember the design principles and guidelines for user testing from the lecture on usability.

As usual, remember to practice good procedural decomposition among other best practices for style. In particular, your GUI constructor and `paintComponent` will probably be long enough that they should be broken into helper methods.

Programming With Swing

In addition to the lecture slides and demos, Oracle's Swing and 2-D Graphics tutorials are a useful resource. Also remember to use the Java API, particularly the `javax.swing` and `java.awt` packages, to see what classes and methods are available and how to use them.

To display the map and routes, you will want to write a custom component and override `paintComponent()`. You will use `Graphics` and/or `Graphics2D` to do the rendering, as we saw in lecture.

If you have trouble getting things to display at the correct size, make sure you have remembered to call `pack()` before setting your frame to be visible.

Other

As usual, your program should look for files using relative filenames starting with `src`.

What to Turn In

You should add and commit the following files to SVN:

- `hw8/CampusPathsMain.java`
- `hw8/*.java` *[your GUI classes]*

- `hw8/answers/model-changes.txt` *[list of changes to HW7. The file may simply contain "None".]*
- `hw8/answers/extra.txt` *[list of optional features you implemented. The file may simply contain "None".]*
- `hw8/answers/reflection.txt`
- `hw8/answers/collaboration.txt`

Additionally, be sure to commit any updates you make to your Homework 7 files.

Finally, remember to run `ant validate` to verify that you have submitted all files and that your code compiles and runs correctly when compiled with `javac` on attu (which sometimes behaves differently from the Eclipse compiler).

Errata

The final paragraph of the introduction has been updated to its curren state from:

However, you are *not* required to fix code quality/organization issues in your Homework 7 code. Your Homework 8 code should be well-written and well-organized, but we will not double-count for MVC-related style issues caused by design choices from Homework 7.

Q & A

None yet.