Assignment 2
COMP 86

Introduction

For the next few assignments, we will build up an imaginary spaceship traffic monitoring or control system -- or any other similar type of interactive simulation or game you prefer -- step by step. You can choose what to simulate and design the rules of the simulation, its graphic design, commands, and operations any way you like, as long as they follow the general requirements described in the assignments.

The simulation consists of a number of "vehicles" moving around on a "map" or background area. The vehicles could be spaceships, satellites, asteroids, airplanes, cars, trucks, ships, Tufts shuttle buses, or anything you prefer. You can choose a project that is relevant/interesting to you, because you will be working on the same project for the next several assignments. You can choose any application that satisfies the technical requirements in each assignment (buttons, animation, picking, etc).

Your program display might look like a ship radar display with ships moving around the water, an air traffic control console for conventional airplanes, a highway or railroad network with cars and trucks travelling along the roads or tracks, or a hypothetical spaceship control system. It will have a display like a map or radar screen, showing the vehicles moving around and various pushbuttons and other GUI controls to interact with the simulation and control the vehicles. Eventually, we will provide widgets for the user to interact with the map as a whole as well as to create, modify, and delete individual vehicles on it.

Assignment

The first step, obviously, is to decide on what your application will be -- spaceships, planes, ships, trains, asteroids, or whatever you like. Think about what your program will simulate, how it will work, and how the user interface will look and work. Submit a short (1 paragraph or so) description of your design.

Then, write a Java/Swing application program that creates a window with some of the main elements you will need for this project. It should contain a blank canvas or drawing area and one or more control panel areas with some buttons, scrollbars, or other widgets for the functions you might want in your final program. It should provide a reasonable layout for the window and its elements. The drawing area is the section of your window where the main action will eventually occur. For this assignment, create a blank canvas or drawing area widget, which will consume a significant part of your main window. (This object can can subclass JPanel.) And draw something very simple inside of it.

Buttons and other Widgets

Each button should have an appropriate label. Give your widgets plausible-sounding names for features your simulation might have. Your program should print a different message for each of the buttons. Similarly, other widgets should print messages when they are used.

Main Program

Your main() program, which is a public static method in one of your classes, should simply instantiate an object of that class. Its constructor will then set up your window, instantiate and initialize the objects you need, such as the buttons and canvas, setVisible() the window, and then let the window system take over and wait for callbacks.

Once this program is running, it will simply display your objects, rearranging as necessary if the window is resized or exposed.

You should probably choose a BorderLayout for the main window, which is the default for JFrame anyway. You can experiment with various ways of arranging the subcomponents to find one you think works best. Typically you would put the canvas in the center, and the buttons and other items in some of the border areas.

(Note that if you use a FlowLayout instead, then to set the size of your canvas you may need to use: `setPreferredSize (new Dimension (width, height))` rather than plain: `setSize (width, height)`.

Include a text file with your submission, containing instructions for how to compile and run your program, the short description of your design as mentioned above, and of any other special features of your program you want to call to the attention of the grader.

Program Design and Practices

Your program design should exploit the features of object-oriented programming (encapsulation of code and data, support for abstract data types, polymorphism/overloading, inheritance).

You should follow these general Java programming practices:

- Make all instance variables of your classes **protected** or **private**. If you need access outside of the class, provide it with *set* and/or *get* methods; don't access protected the variables from outside the class (even though Java -- unfortunately -- allows us to).
- Avoid most global variables or widely-accessible public variables; pass the data you need explicitly.
- Put each class in a separate .java file.

Design Documentation

In addition to your program, submit documentation about the design of your system in these forms:

- An outline showing the inheritance hierarchy
- An outline showing the aggregation hierarchy (which objects contain or "own" which other objects)

Submit this documentation electronically in text form, along with brief instructions for how to compile and run your program. Include it as part of the `readme` file that you submit with your assignment.

These diagrams may be trivial for this assignment, but will become more interesting with later assignments.

Grading

As in the previous assignment, a portion of your overall score is reserved for work that goes beyond the basic minimum assignment specification.