SE350: Spring 2018

Homework #2: Christopher Columbus

Assigned: **For Monday Class (902 & 910)** - Monday April 9th
For Wednesday Class (901) Wednesday April 11th
Due: **For Monday Class (902 & 910)-**Tuesday April 24th at midnight
For Wednesday Class (901) Thursday April 26th

Objective:

Write a JavaFX program that builds upon the Lab assignment. The first step is therefore to complete the Lab 2: "**Christopher Columbus Sails the Ocean Blue**". The grade will include evaluation of the completed Lab assignment.

- ✓ **Steps described in the lab include:**

1. Creating a 10X10 Ocean Grid
2. Adding a ship
3. Adding a keypressed handler so that you can move the ship around the map. Do not allow the ship off the side of the grid.
4. Adding a N random islands to the map. Prevent the ship from going into the island.

- ➢ **Additional Required Functionality:**

Once you have completed the Lab assignment – you will need to add additional functionality that uses the observer pattern.

Detailed Description:

Pirates

In this step you will add at least two Pirate ships to the ocean in any available cell. (Alternately you may designate two islands as Pirate islands and launch the ships from those islands--Optional). I have provided a pirate ship icon – or you may find your own.

You will then implement the observer pattern using Java's inbuilt observer/observable. The pirate ship will register itself with Christopher Columbus' ship (CCS) and will receive notifications each time the ship moves. The pirate will use the information about CCS location to chase CC.

Hint 1: Don't forget to load the new ship into an ImageView and add the ImageView to the GraphScene (node tree).

Hint 2: You are going to need to create a class for the PirateShip. Remember this class will implement the observer interface. Your ship class will now extend Observable.

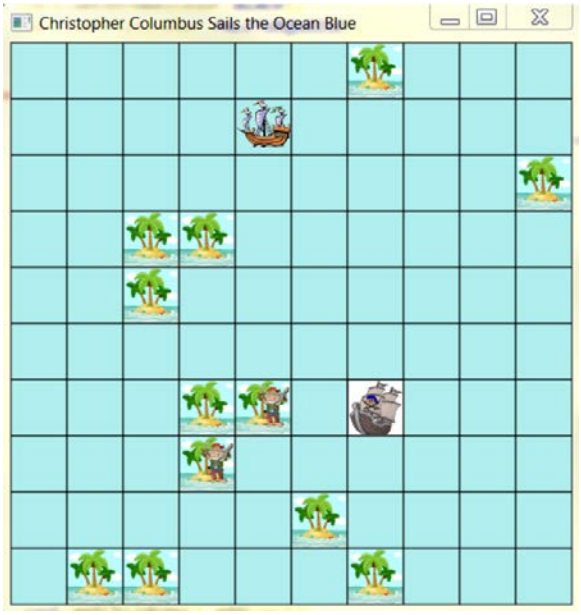
Stretch Suggestions

For some of you this will be easier than others. If you are looking for additional learning opportunities here they are:

- Add a button on the bottom of the screen. When you press the button – the game will reset. i.e. the ocean will be cleared, new islands will be created, and new ships will be placed. There are two different ways to think about this: (1) if the number of islands remains the same – then you can just randomly relocate all the ships and islands that already exist. (2) if the number of islands is going to change then you can remove everything from the scene graph and recreate the entire scene from scratch. Either option is valid – so just decide which you want to do.

- Replace blue rectangles with ocean images and/or islands with Island images.

Here is my finished example:



How to submit: All homework for this course must be submitted to D2L.

Points: The assignment is graded out of 50 points. For posting to D2L the points will be divided by 5 to produce a score out of 10. i.e. if you score 40 points you will receive 8 D2L points.

Points will be broken down as follows:

1. **10 points** – A UML class diagram of your solution. I will NOT accept computer generated after-the-fact sketches. You can sketch your UML on paper and take a photo of it if you wish, or use a tool such as Visio. Why? Because thinking about your design in advance is an essential development skill.
2. **5 points** – Journal (ONE short paragraph explaining your design decisions)
3. **35 points** – Functioning code which complies with the design requirements. Points may be deducted for reasons such as incorrect use of observer pattern, missing/incorrect key handler, zero documentation, poor coding decisions, missing functionality. You will receive a list explaining why any points were deducted.
4. **Late Penalty** – 10% penalty if turn in within 48 hours after the deadline.
   20% penalty if turn in within 7 days after the deadline. I do not accept assignments after 7 days of the deadline.

What and how to submit?

Submit only 1 zip file on D2L under assignemnt1 which include:

a. A folder called “Source” and place the source code into it.
b. Your UML class diagram sketch.
c. Your one paragraph explaining why and how observer design pattern suits this problem. And other design decision you made.