Anna Smith

Design Rationale

To design my 'ocean explorer' I used four classes. The OceanExplorer class handled all GUI operations. OceanExplorer extends the Application class. Thus it had a start() method, which it overrode from the Application class, for operations which were handled at the start of the application. It first creates an AnchorPane which will be used to bind objects to the GUI window by their (x,y) coordinates. It calls in a Stage object as an argument to start() and creates a Scene which is added to this Stage and displayed on the GUI. It creates three objects: an OceanMap, Ship, and list of Pirates, which are all added to the pane. These three objects are each defined in their own class. The OceanExplorer class also has a main() method which uses an inherited launch() method from Application to start the application. A separate class, the OceanMap class contains two methods: drawMap() and placeIslands(). It will create the grid used for the ocean explorer GUI and display islands on the grid. On the grid will be displayed Ship and Pirate objects. The Ship class keeps track of the current position of the Christopher Columbus ship by initializing a currentLocation variable in its constructor and updating this variable in methods called from startSailing() in OceanExplorer, which tracks user key inputs and moves the ship accordingly. The Ship class extends Observable and is an observable for the Pirate class, which extends Observer. When the ship moves it updates the Pirate class with inherited methods setChanged() and notifyObservers(). The Pirate class keeps track of both its location and the Christopher Columbus ship location. It overrides the method update() from Observer and calls its own method moveShip() to move the pirate ship in relation to the other ship. As the program uses multiple pirate ships, the OceanExplorer class instantiates a list of Pirate objects that are each individually set as observers of the Ship object and added to the root pane. The Ship image and image location are set in OceanExplorer but the Pirate image variables are updated within the Pirate class as the pirate images will need to move whenever the ship location is updated. This program design worked well as the OceanExplorer class controlled all GUI operations and kept track of updates to class instantiations of OceanMap, Ship, and Pirate and the class methods and data objects of these three classes were controlled within their class. Also the observer, observable pattern worked well for Ship and Pirate as it provided methods and organization for how to update the pirate location for each move of the ship.