**Monopoly Game Spec**

Summary: A simple well-designed version of Monopoly with graphics displayed to help the players see what is going on within the game. Designed with many classes including: the player, property, card (for the chance and community chest cards), board, the main, and also various graphics classes dealing with displaying the board and players within it. The game plays as a regular game of monopoly, with expandability so in the future can easily be turned into the landlords game (the original version of monopoly).

Essentially, for those that don’t know how to play monopoly, players roll two dice and move that many spaces on the board (in the direction the GO tells you to). When a player lands on a spot, they have the option to buy the space or not. If they buy the space, when other players in the future land on the space, they will pay rent to the buyer. As the game progresses players will accumulate properties, and will be able to do several things with them. They have the option to trade the properties to other players in return for their properties and/or money. They have the option to mortgage the properties if they find themselves without money. And lastly, if they collect a full color set of properties, they have the option to build houses on them, which increase the price of rent. The game continues until all other players except one go bankrupt.

**Interfaces and File Formats**

* To interact with the game, the player will use a GUI
  + The GUI will include buttons, to roll the dice and move, and include a console to let them type decisions they want to make about their turn
  + All of the inputs and moves that a player does will be stored within the console to allow for debugging if necessary
* For many parts of this outside files are required
  + These files are for getting information about properties, chance cards, community chest cards, etc.
    - For properties and players this will include information about where they are and how to place the players

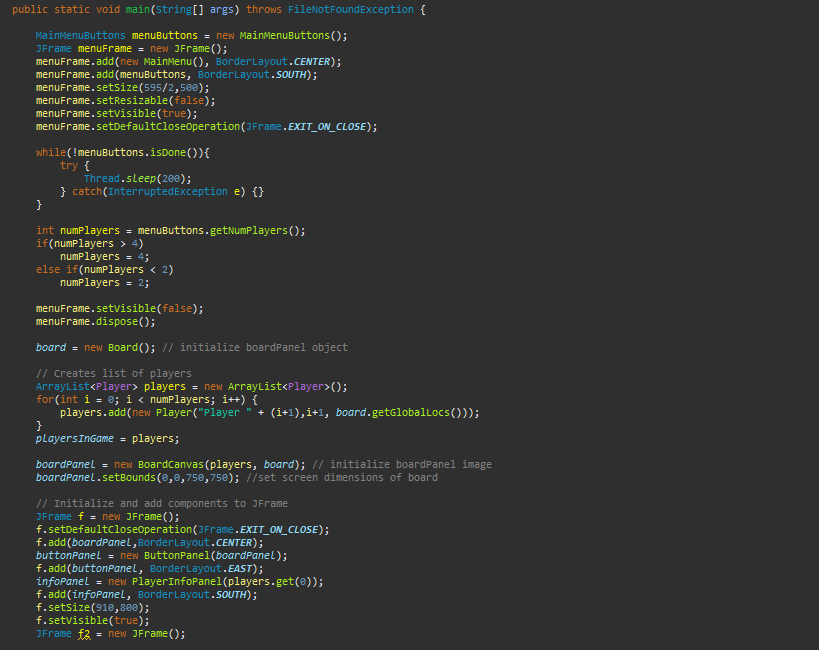
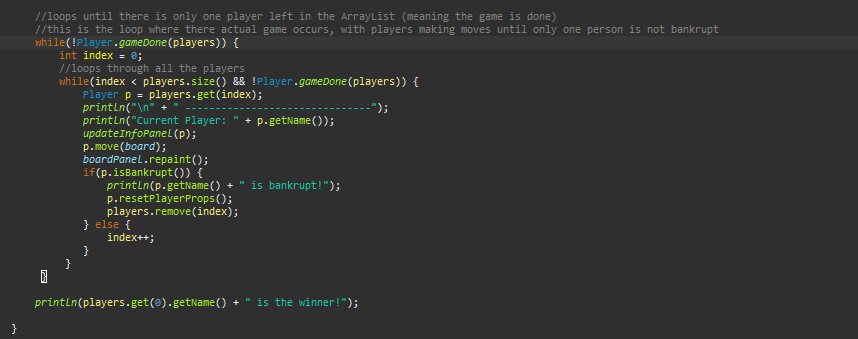
**Game Mechanics**

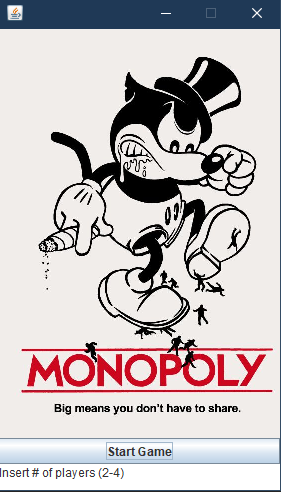
* Allow player to use button to move
  + This button will auto roll dice for the player and move their piece automatically to the correct place on the board
  + After landing on a property will tell the player what will happen next within the included console on the GUI
* Player Object Fields
  + Name of the player to be displayed when someone is making their move
    - Num of the player to keep track of which player this is
  + The current location of the player
  + The amount of money the player currently has
  + The properties that the player currently owns
    - Can be easily added and removed from, so use an ArrayList to keep it simple
  + Whether or not the player is in the jail
  + The cards the player has in their hand (get out of jail free cards)
* Turn progression (will occur after creating the number of people you are going to play with)
  + Player presses the move button
    - Auto rolls dice (generating two random numbers between 1 and 6 inclusive)
    - Moves the piece to the calculated location
  + Player then interacts with whichever property they are now on with the following possible options
    - Purchase property
    - Pay rent
    - Draw chance or community chest card (and then receive its effect)
    - Go to jail
  + Player then reaches the ending of their turn and has the following options remaining
    - Trade
      * Allows the player to select another player that they wish to trade properties/money with
    - Buy houses
      * Allows the player to build houses on properties that they have the full set for
    - Sell houses
      * Allows the player to sell houses on properties that have houses built on them
    - Mortgage
      * Allows the player to mortgage the property, receiving money in exchange for receiving no money until repayed
    - End turn

**Scalability**

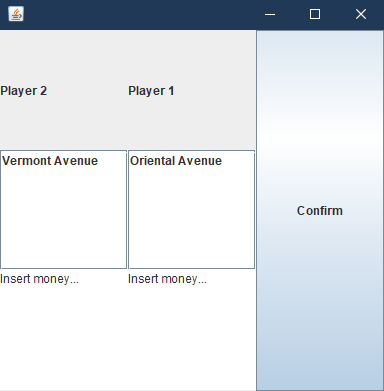
* Scaling up
  + Implementing the entire landlords game alongside the regular monopoly rules (this is the original monopoly)
  + Allowing the whole program to be operated through the use of graphics, animations, and buttons
* Scaling down
  + Getting rid of some of the end game functionality
    - Including but not limited to: trading, buy/sell houses, mortgage
  + Getting rid of many of the graphics implementations
    - Should still include players moving around a board, but the rest could be operated through the console

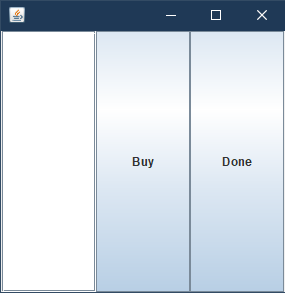
Main



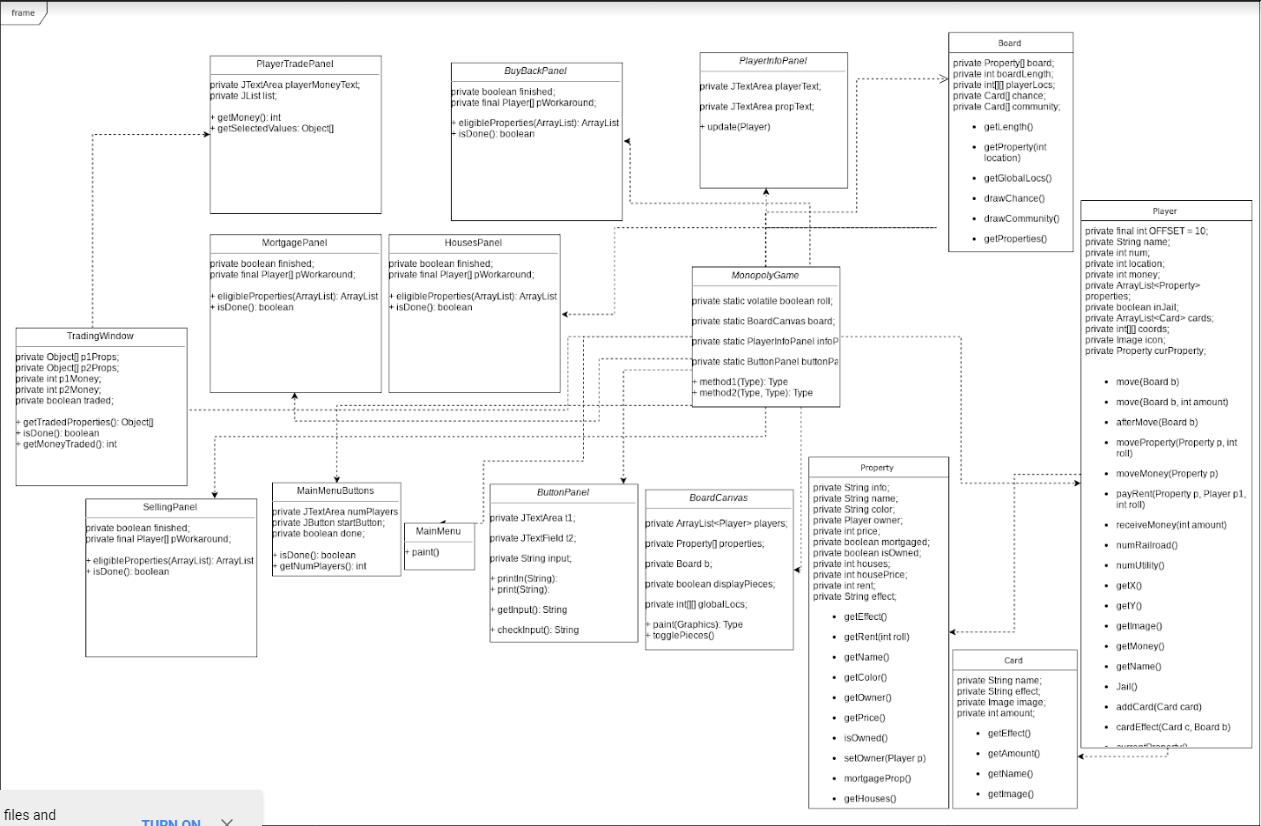
Output







UML Diagram



Graphics Interface Lesson

Things we learned:

JFrame: JFrame is the Java swing construct that creates a window.

JPanel: JPanel is the Java Swing construct that groups together multiple components into a single panel. This panel can then be displayed in a JFrame

JComponents (JButtons, JTextAreas, etc): JComponents are functional and graphical elements in Java Swing that are placed into JPanels and then displayed on JFrames. They gave us the functionality of buttons, allowed us to display text, and allowed us to scroll text.

Paint method(): The paint method is a JPanel method which allows the programmer to procedurally draw images, boxes, and other constructs onto a panel. This is how we displayed our board and the pieces and markers on it.

Overall Project Results

* Spec
  + We successfully completed the entire spec we set out to do, creating the monopoly game. We added a few extra things to it, moving in the direction of expanding the spec to what we wanted (with making it all graphics based), however were unable to completely make it there as there were more small issues that we had to iron out than we originally thought. We added a main menu to allow selection of a number of players, and created trading, houses, and mortgaging windows that are all graphically based, but the core gameplay is still based on the user inputting into the given area.
* Performance
  + Program runs well, with the only slowdown occurring when one starts the program as it is initializing everything that will be used for the rest of the program then (even with that, it is only 1 or 2 seconds).
* Quality of codebase
  + Our code is mostly robust, with many things being grouped together to allow ease of access to related things.

Reflection

* What went well
  + Overall, we are happy with the final product we created, with a large part of the game being able to be easily played through the use of buttons, etc. We are extremely pleased with how smoothly it all went, with no real big bugs that involved recoding anything.
* What went poorly
  + One thing we realize we could have improved upon, especially given more time, would be the user interface. We created the expansion option in our spec to make the entire project graphics based, but were unable to get that far and instead the user ends up having to use a console essentially (the console is put within the graphics overlay however, so it’s still better than the direct console of eclipse).
* Recommendations for improvement
  + In the future expand to have all the user inputs be through the use of buttons to allow for ease of access. Other than that, there isn’t too much that could be improved upon as everything works as intended. For extremely far out goals, one could attempt to implement networking or an easier option could be to implement various versions of monopoly.