Monopoly Group 3 Design Decisions Made Milestone4:

Authors:

- Sarah Chow 101143033
- Kyra Lothrop 101145872
- Bardia Parmoun 101143006
- Owen VanDusen 101152022

MVC Pattern:

- MVC Advantages: flexibility, re-use, multiple presentations of the same data and allowing the user to select what data to present during runtime.
- For all these reasons for user interaction with a monopoly game, the MVC pattern is used
    - **The Model:** BoardModel.java
        - This class handles the logic for the monopoly game
        - BoardView/Listener are BoardModel's only point of contact. BoardModel sends updates that are printed in BoardController if user input is required or printed in the view if no user input is required
    - **The View:** BoardView.java
        - Interface class which contains the method signatures that are implemented in the BoardListener class
    - **The Controller:** BoardController.java
        - This class prompts users for information and passes the information to the Board Model class

Abstract class - BoardCell, Interface class - Buyable :

- **Board Cell**: Abstract BoardCell.java contains the name, owner and cell type of the cell with getters and setters
    - This contains the information relating to individual cells on a monopoly board
    - All cells that are a part of the board such as Go, Property, Railroads, Utilities, Jail, GoToJail extends BoardCell to have any board location classes with specific Monopoly attributes.
- **Buyable**: The interface class Buyable holds method signatures that are implemented by buyable locations on the board.

BoardEvent, Board Frame, Monopoly Event:

- **Board Event**:
    - BoardEvent encapsulates the information that the model passes to its views
    - It is the main point of contact between the BoardModel and the BoardListener
- **Monopoly Event**:
    - MonopolyEvent.java is the main point of contact between the BoardModel and the BoardListener
    - MonopolyEvent is an abstract representation of all the different event types that can be passed to the listener from the model
- **Player Event**:
    - PlayerEvent.java is the main point of contact between the BoardModel and the BoardListener
    - PlayerEvent handles all the events relating to players

BoardFrame:

- **BoardFrame**:
    - BoardFrame extends JFrame and contains the swing elements that the user can interact with to create a GUI-based board

Array List Usage:

- The views, cells and players in BoardModel are stored in arraylists to be able to remove and add to the arraylists
- The addition capabilities of an arraylist are needed to add many properties/go/tax locations to the board for the cells arraylist

Enum Usage**:**

- Enums are used for Status, CellType, Command,
- A main reason enums are used is to reduce the chance of a typo in the code and in turn the need to update the typo in multiple locations in the classes
- In using the enums for the statuses they can have a stringCommand to keep track of the player string commands. Using this, there is only one location in the code that contains the string of the possible player commands

AI players:

- AIPlayer.java extends Player
- AIPlayer contains nextMove method which decides what the AI players will do depending on the command options
- AI decisions:
    - The AI player will buy properties if given the option
    - If the AI player lands on a location where they owe fees, the fees are paid. If the fees cannot be paid, the AI will try to sell properties and if the AI still cannot pay the fees, it will forfeit the game
    - If the AI rolled doubles it will roll again
    - If the AI has not surpassed the build limit it will build a house
    - If the AI may not do any of the activities listed above, it will pass the turn to the next player

Save and Load:

- Applied Java's built in serialization to save and load the status of a Monopoly game. This is done by BoardModel implementing Serializable and the BoardModel methods of serializationLoad and serializationSave to load and save games respectively.
- In order to create a customized board, the information for the unique board is saved in an XML file with all the information pertaining to the game such as the names, prices, image paths, etc. Once the user selects which board they would like to use, the setup information for the game is found in this file.
- Since the user can create new games, load and save games, a method in the BoardModel accounts for the changing of the board by using an infinite while loop and the boolean variables loadGame and newGame. If loadGame is true, and newGame is true, a new game will be made. This will prompt users for the players and game information then start the game by calling play(). If only the loadGame is true then just play() will be called.