Design Manual

To create a virtual game of Monopoly, we decided there were two main components. First we had the board. This was created using the JavaFX library, and is minimally interactive. It has a button to roll dice, and then displays the roll. It shows player information and where they are on the board. And also shows the board in its entirety so that the user can see where everything is.

The other main component in this project was the underlying code that makes the game work. This includes our main classes that are used to create objects of players, properties, and dice, among other things. These pieces are tied together with a main class that starts the game, and runs the game with specific rules. Many of these rules are already in place in the classes, however. For example, in the Railroad class the payRent method takes into account how many railroads the property owner owns, in order to calculate the rent owned by the player who landed on it.

User Stories

<u>Completed:</u>

- As a starting player, I want the bank to give me $1,500 to start
- As a visual person, I want a board that looks much like the real monopoly board so that I feel like I'm really playing the game
- As a visual person, I want two dice to roll
- As a player, I want two random numbers to be generated so that I can determine the number of spaces to move

- As a player with properties, I want to be able to see what I own
- As a player, I want each property to have a set cost to buy, know how many houses are on the property, and rent based on the number of houses and hotels
- As a player who is in jail, I want to leave jail when I roll doubles or pay $50
- As a popular person, I want to be able to play with 3 friends for a 4 person game
- As an advanced player, I want to have the monopoly tokens to represent players
- As a player, I want to move the number of spaces I rolled
- As a real estate owner, I want to be able to buy houses *(see rule change)
- As a player who landed on an owned property, I want to be able to pay rent
- As a player who landed on an unowned property, I want to be able to buy the property or not (with no penalty), if I have enough money
- As a player who doesn't have enough money to pay rent, but has houses, I want to sell enough houses to pay rent
- As a player who doesn't have enough money to pay rent and has no houses, I want to sell enough property to pay rent
- As an advanced player, I want to have to pay $50 to get out of jail after 3 times of not rolling doubles
- As a real estate owner, I want there to be a maximum of 4 houses or one hotel on each property
- As a visual person, I want a space marker to be able to see where on the board I am

<u>Incomplete:</u>

- As a group starting the game, I want a random player to be picked as the start person and an order of players to follow
- As a player who passes GO, I want to earn $200.
- As a player who lands on community chest space (3), I want to pick up a community chest card and perform its action.
- As a player who lands on a chance space (3), I want to pick up a chance card and perform its action
- As a player who lands on a tax space, I want to be able to pay tax to the banker
- As a player who doesn't have enough money to pay rent and has no properties, I want to be kicked out of the game
- As an advanced player, I want to be able to trade with other players
- As a more advanced player, I would like a richer experience with free parking (get money that has been collected from community pot)
- As a more advanced player, I want to be able to play against the computer for practice
- As an advanced player, I want $400 for landing on GO
- As an advanced player, I want to be able to trade properties with another player
- As an advanced player, I want a richer experience with the more complex chance and community cards (i.e. moving pass go without monetary reward)
- As an advanced player, I want to be able to mortgage properties
- As an advance player, I want to go to jail after rolling 3 doubles in a row

In order to follow good object oriented design, we tried to use user stories to break our problem up into manageable parts, and create many classes based on these stories.

For example, we had many user stories that talked about the properties monopoly. User stories wanted to be able to buy an unowned property, put houses and hotels on a property, and pay rent when you land on an owned *Property*. Using these stories we developed CRC cards, in this case we developed the following:

Class: Property

Responsibilities: knows its name, knows its color, knows its purchase price, knows if it's owned, knows who owns it, knows its rent cost, knows which properties have houses/hotels

Collaborators: Player

This CRC card helped us decide what classes we wanted to implement, and what they help us accomplish. For the property CRC card/class we decided that we wanted it to have several properties. These then turned into the attributes and methods of the *Property* class. We also had *Player* as a collaborator for *Property*. This meant that we had the *Player* class called/used in the *Property* class. This came in the form of the property knowing who its owner was. We actually had the player know which properties they owned as well.

In the end we wanted the *Property* class to be able to individually represent each property on a monopoly board, with the other spaces being represented by the *Jail*, *Railroad*, and *Utilities* classes. All of these classes that represent spaces on the board would be connected to the *Player* and *DieModel* classes, but not each other. The *DieModel* class is used to roll the dice, and this data is needed to move the player and decide some things in the specific classes.

The *Player* class was another important CRC card, and related to almost every user story based on the fact that they all assumed that they would be playing the game and have certain things belong to them. The CRC card is as follows:

Class: Player

Responsibilities: knows how much money they have, knows what properties they have, knows what token represents it.

Collaborators: Properties, Die, Utility, Railroad, PlayerToken, Jail

The *Player* class is at the root of most of our project's functionality. It is what keeps track of who a player is, where they are, how much money they have, and what they own. It is the part that ties together all of our other classes. Because it needs to have information from almost every other class that we created.

Once we started to work on the project, we realized that creating the graphical part of the board would be more of a challenge than we initially anticipated. So we allocated more resources and made an entire package dedicated to the graphical board. So this went from just being its own class to being a collection of classes. This seemed worth it however, because so many user stories talked about the importance of seeing what is going on in the game by having a board. We specifically referenced the following user story:

"As a visual person, I want a board that looks much like the real monopoly board so that I feel like I'm really playing the game"

This guided our decision to put more time into creating a good looking board, because it seemed important to the user to have something that makes them feel as if they are really playing Monopoly.