Bilkent University

Department of Computer Engineering

CS 319 Term Project: Monopoly

Section 01

Group 1A

FINAL REPORT

Atakan Dönmez
Elif Kurtay
Musa Ege Ünalan
Mustafa Göktan Güdükbay
Yusuf Ardahan Doğru

Instructor: Eray Tüzün

Teaching Assistant(s): Barış Ardıç, Emre Sülün, Elgun Jabrayilzade

Table of Contents

1. Introduction

In this project, we completed the implementation of the Monopoly game. In the beginning, we introduced new rules such as each piece having special skills, a thief card in the cards, a player choosing to open a chance or community chest one turn after drawing, a player being able to move when they are on a transport station, and a wheel of fortune instead of a free parking space. We implemented the mentioned new rules, except for the transportation rule. The transportation rule was to allow players to move between 4 transportation spaces when they visited one. However, later on we thought that this rule would make the game unfair among the players and decrease the enjoyment and satisfaction. Besides, many other new and interesting features are also introduced. For example, players are allowed to open a chance card or community card after more than one turn which is referred to as “postpone” in the game. We made this modification to give more flexibility to the user. Another example is the “Thief” which is a temporary fake player that will join the game from a card. After joining, the thief will have a random target player and it will try to chase that player. If it does catch the player, the thief will steal money from the targeted player and disappear. All of our new rules we introduced at the beginning of our project were all implemented.

Additionally, we proposed to add new features such as designing custom maps and choosing them, the game having single-player and multiplayer options where computer players are added into the game, save and load options, restart the game option, and putting a turn limit or giving a right to forfeit to decrease game time. We implemented all these features as well.

The work distribution among the group members are as follows:

Atakan Dönmez:

- Implementation of the auction operation
- Cards and card events
- User-Interface design
- Implementation of the GameScreenController
- Functional Requirements
- Class Diagram
- Contribution to the reports

Elif Kurtay:

- Implementation of digital player using decorator and strategy pattern
- Implementation of the main game logic and Game class
- Implementation of the thief
- Advancements to controller classes
- Restart option
- User-Interface design
- Use-case Diagrams, Activity Diagrams, State Diagrams, System Decomposition
- Contribution to the reports

Musa Ege Ünalan:

- Implementation of trade and build operations
- Implementation of Board and Space classes
- Custom map design
- Implementation of the GameScreenController
- User-Interface design
- Class Diagram and Object Design
- Contribution to the reports

Mustafa Göktan Güdükbay:

- Load screen and file manager
- Wheel of fortune screen
- Implementation of audio and music
- Player and Property classes
- User-Interface design
- Activity Diagram, Class Diagram
- Contribution to the reports

Yusuf Ardahan Doğru:

- Exception handling
- Token class
- Sequence Diagram
- Contribution to the reports

We believe that every group member made a significant contribution during all stages of the project, including the analysis, design, implementation and documentation of the project.

2. Lessons Learned

In this project, we learned how to develop and organise a software project using a top-down approach. We used various UML diagrams to express our ideas pictorially during the analysis and design stages, including *class diagrams*, *sequence diagrams*, *use-case diagrams*, *activity diagrams*, and *subsystem decomposition* diagrams. We used design patterns such as *Strategy*, *Decorator*, *Singleton*, and *Observer* patterns. These design patterns helped us when we were writing the code. We learned the importance of UML diagrams and design patterns in software development.

We used GitHub and Git Version Control System during the development of our project, which stores the source codes of our implementation and the documentation for our project. We discovered team work on a programming project through Github. We practised coding cleanly and commenting our codes to speed up the process of understanding each other's code. We also learned to perform testing and debugging on our commits before pushing them to the master branch to reduce future bugs.

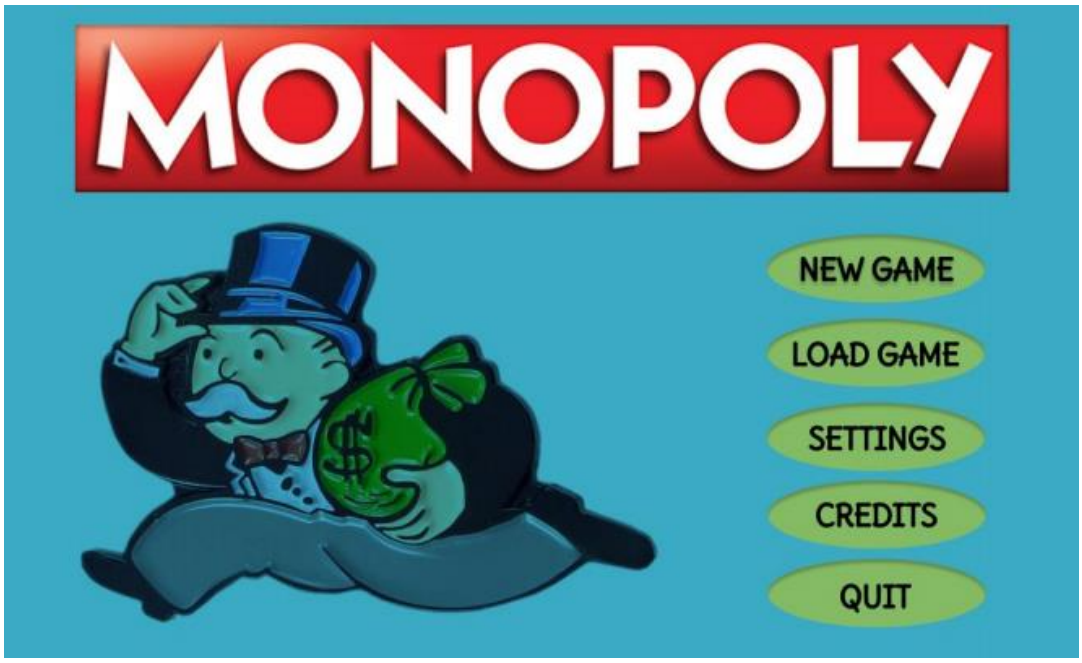
During implementation, we learned and used some useful Plugins and libraries. We used the JavaFX library in our implementation, and we realized the power of this library in creating UI components. In addition, we used the Lombok library to reduce repeated and generic code by using annotations.

We used Slack for project-related communication among the group members, the professor, and the teaching assistants. We also had many meetings using the Zoom application. We had pre-scheduled meetings every week and some more emergency call meetings when needed. We learned the importance of synchronised communication and how to organise the time of our meeting. By following our plan for communication from the beginning to end of the project, we never had unexpected time related problems, loss or delay of information, a disconnection of a team member, or wrong understandings. Furthermore, we shared a drive folder where we kept all of our deliverables and project related files.

3. User's Guide (Manual)

This section will explain how to play the game step by step to the players.

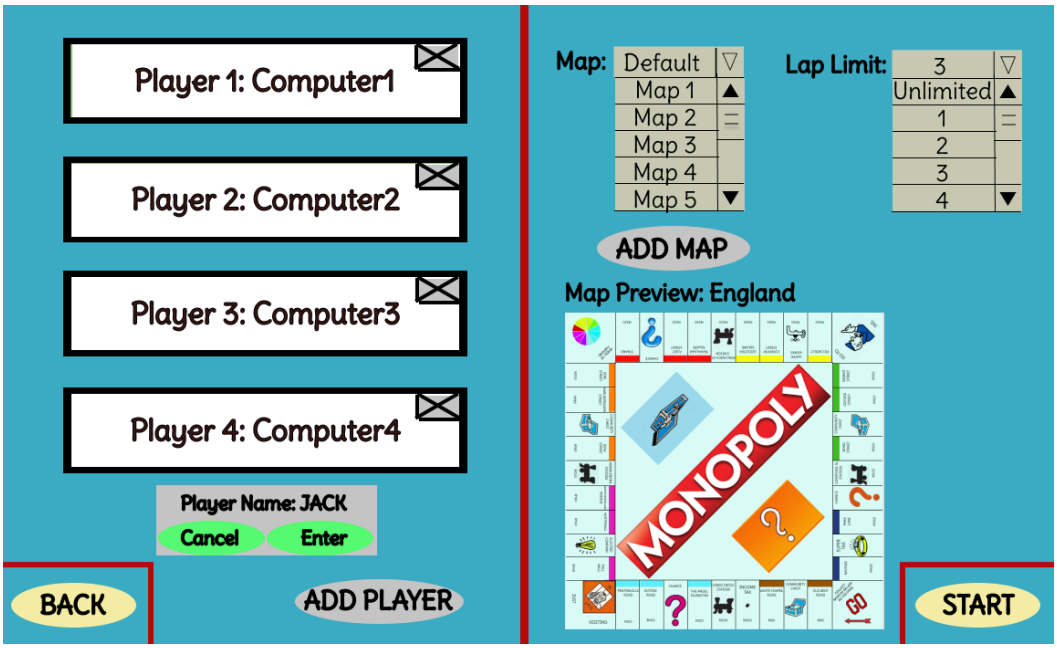
3.1 Main Menu



In the main menu, players can start or load a new game, change settings, access credits, or quit the game. Pressing the **New Game** button initializes a new game. The **Load Game** button loads a previous game. **Settings** allows the player to change the audio, and the player is given access to additional information upon clicking the **Credits** button. The **Quit** button closes the game.

3.1.1 Creating a New Game

Upon clicking the New Game button on the Main Menu, players reach the following screen:



In this screen, players can be added to the game by clicking the **Add Player** button. It is up to the players to decide the count of players and bots. The players are also given access to some properties of the game they create such as different maps and lap limits, which is shown in the upper right of the screen. Users may want to delete some players in case they made some mistakes, which can be done by clicking the **X** button on the top right of the player demonstration. The players, having finished the customization of the new game, may click on the **Start** button at the bottom right of the screen to start the game.

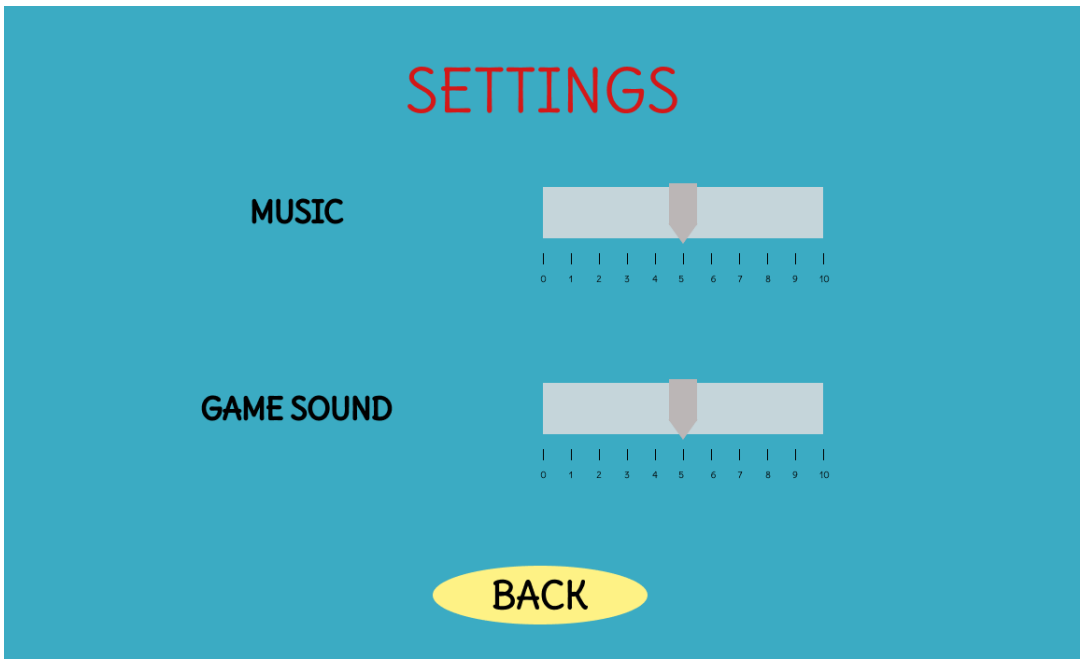
3.1.2 Loading a Game

Upon clicking the Load Game button in the Main Menu, users reach this screen:

In this screen, various games that were played and saved before are shown. The dates, names and the times are shown to make it easier for the player to choose which game they want to resume. Choosing any one of the games loads the game that was saved.

4.1.3 Settings

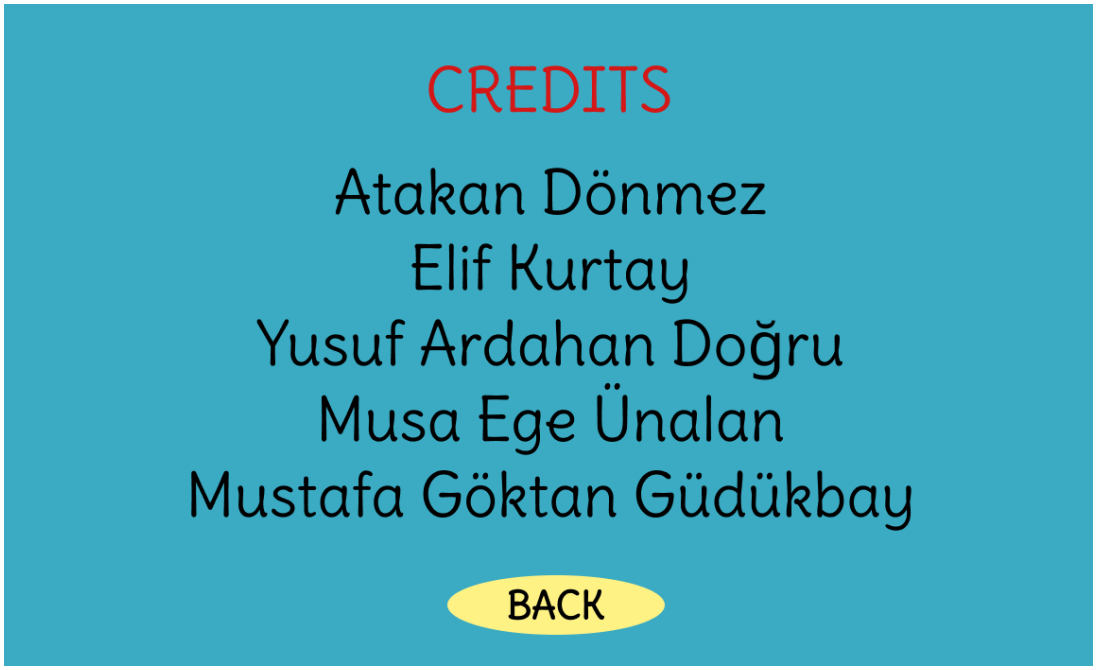
Upon clicking the Settings button in the Main Menu, users reach the following screen:



In this screen, the users can change the music and game sound by clicking and dragging the indicator icon. After finishing the audio adjustments, users can press the Back button to go back to the Main Menu.

3.1.4 Credits

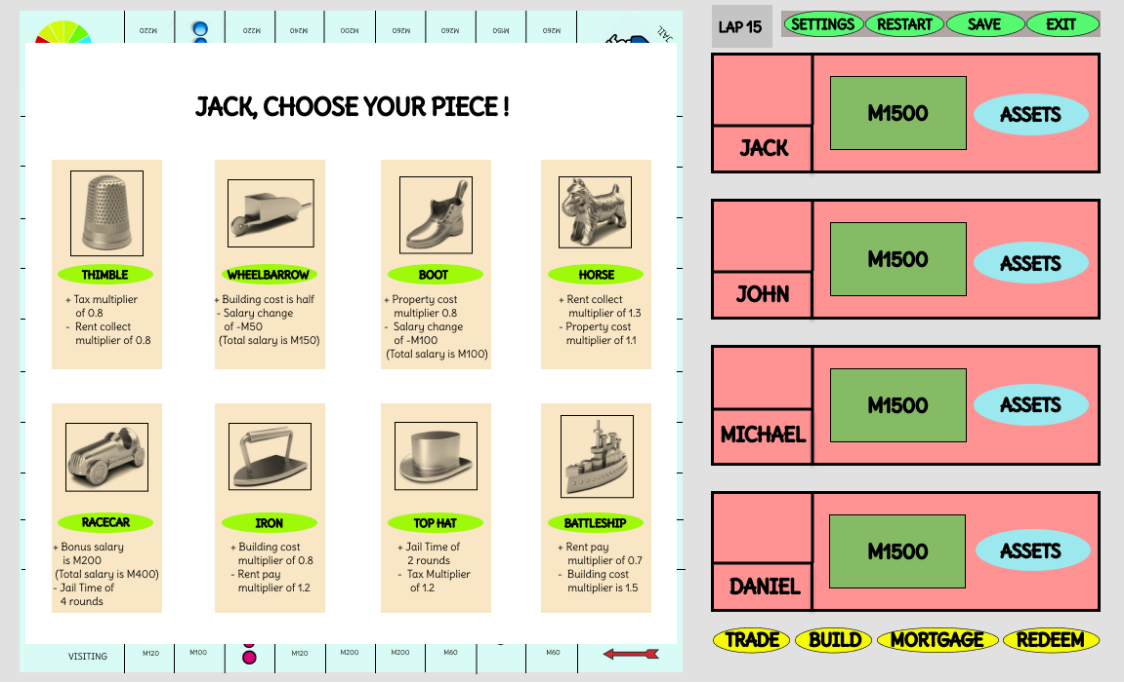
Upon clicking the Credits button in the Main Menu, users reach this screen:



After seeing the developers' information, the users can press the Back button to go back to the Main Menu.

3.2 Initialization

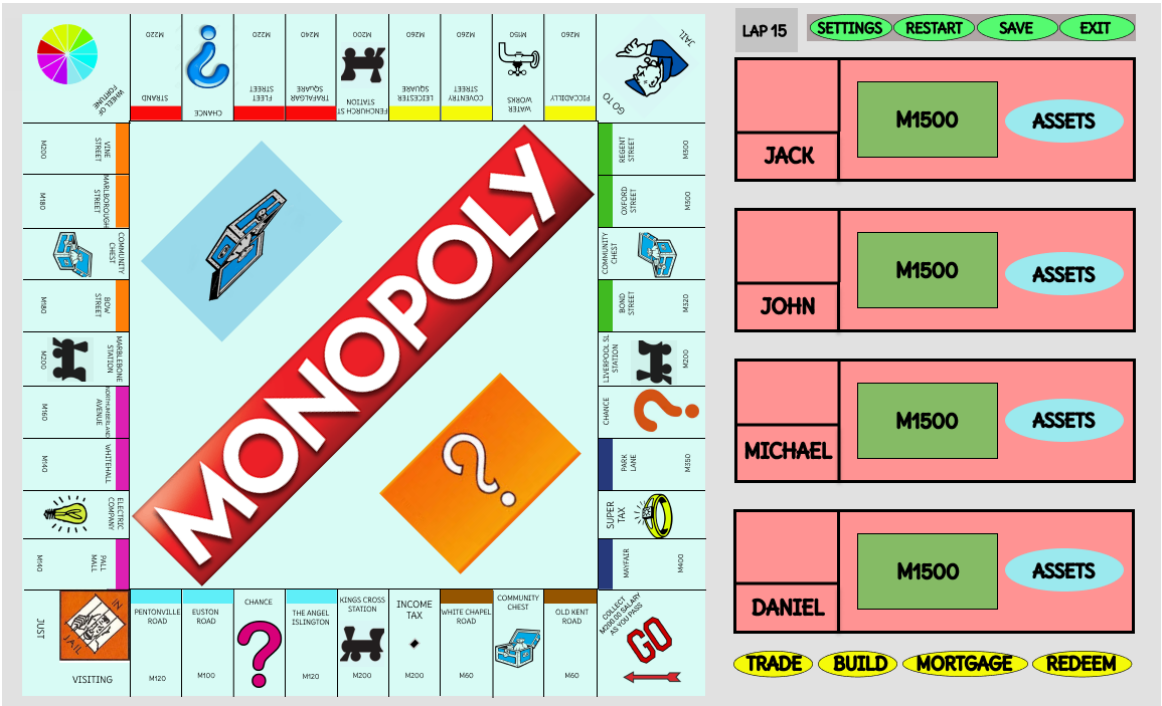
After a new game starts, players choose their avatars, which happens in the following screen:



The players can see the abilities of each avatar and choose accordingly. After the avatars are chosen, the game starts automatically.

3.3 In-game Actions

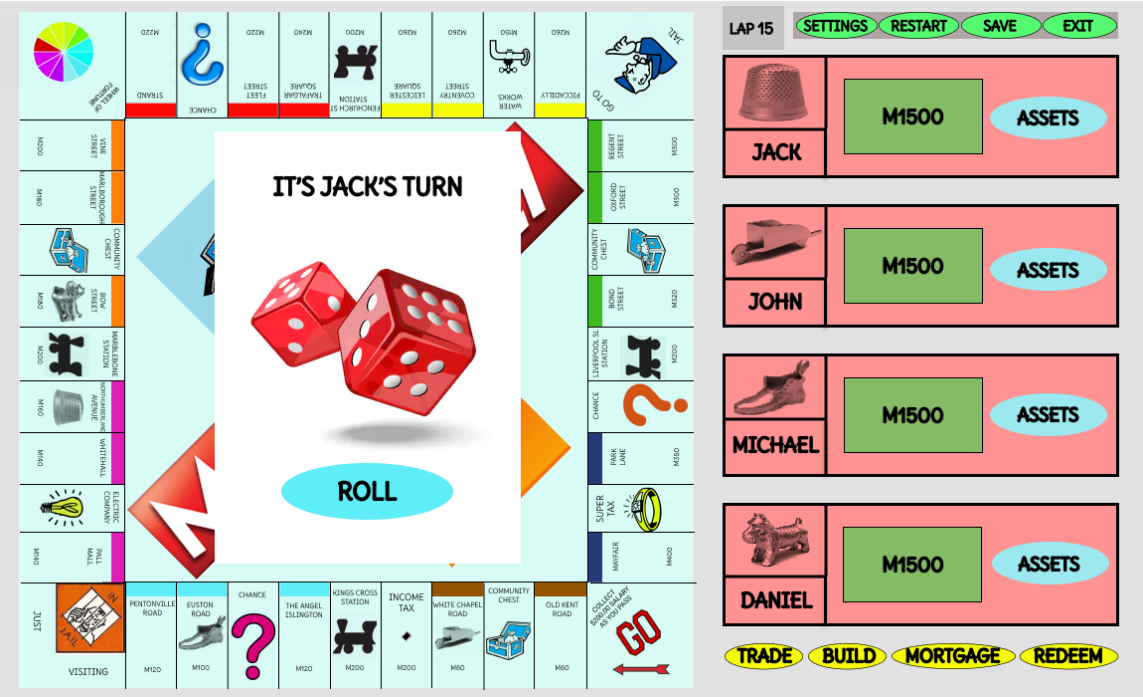
After opening a new or a saved game, the players will encounter the following screen:



The players can choose to restart, save and exit the game by using the buttons at the upper left. They can also access setting in-game with the Settings button. The Assets button that is shown in the players' information can allow players to see which assets other players own. Players can also choose to trade, build, mortgage and redeem with the help of the buttons in the bottom right. The map shown in the screen is a classic Monopoly map which is chosen in creating a new game.

3.3.1 Turn Phase 1

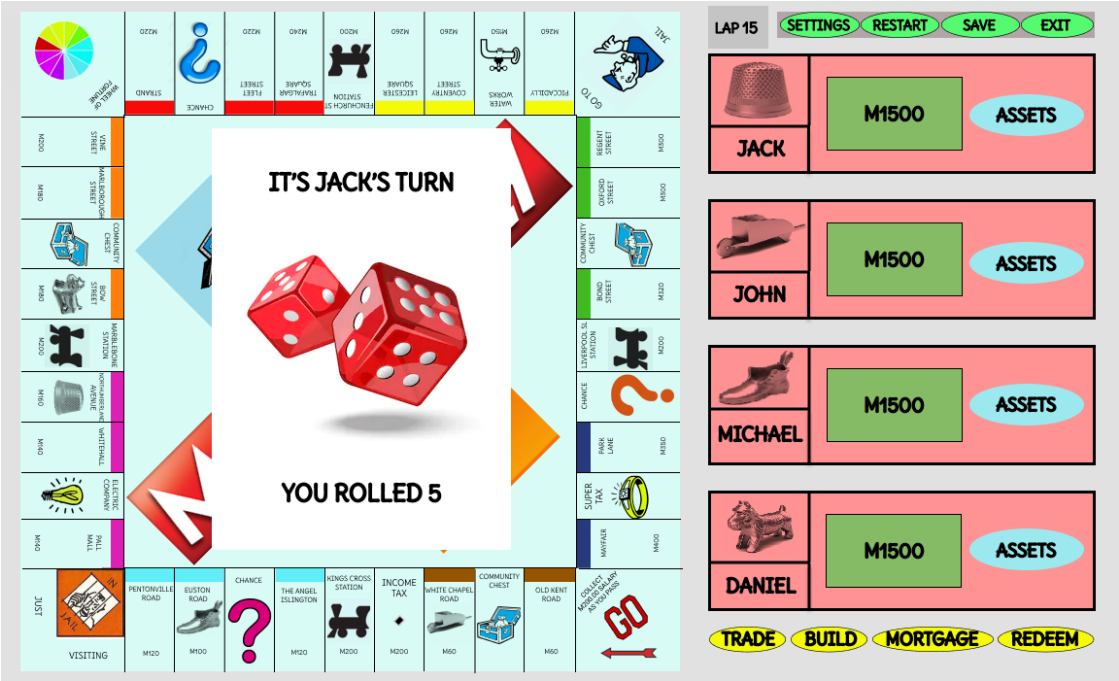
After the game starts, when the sequence and the avatars of the players are determined, at the start of each player's turn, the player will come across the following screen:



Here, the players can choose to roll the dice, or participate in other activities such as trading, building, mortgaging and redeeming. The players can also restart, save or exit the game. Settings can also be accessed. The player can choose to roll the dice before or after the other options. Clicking on roll will reveal the dice that the player has rolled, and changes his/her position accordingly.

3.3.2 Turn Phase 2

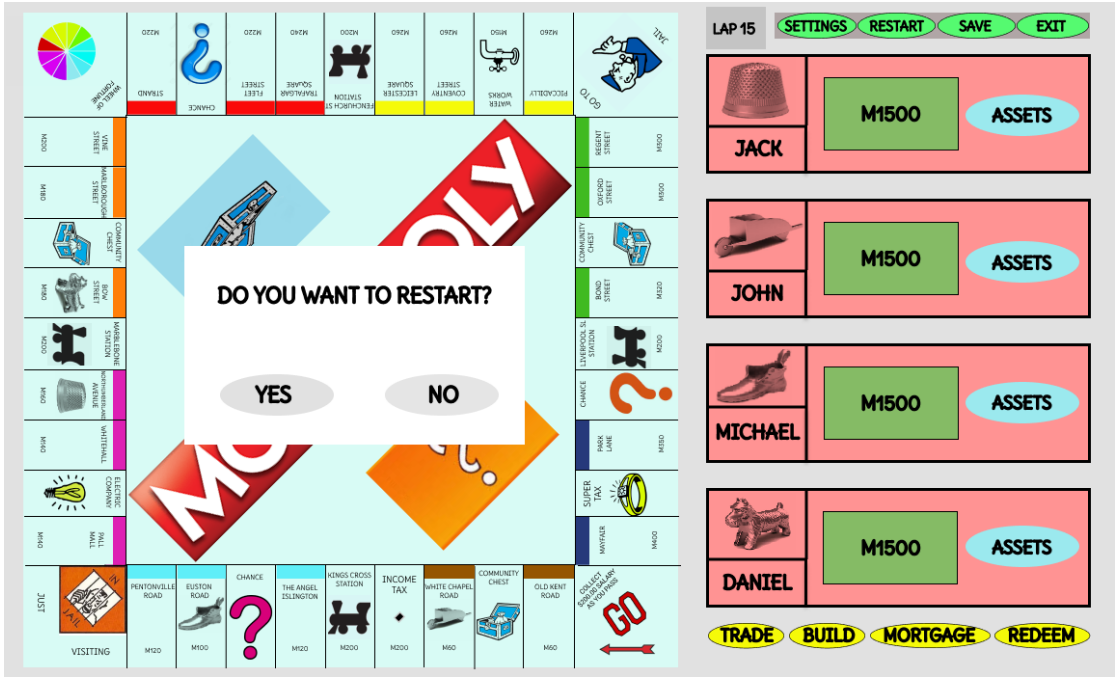
After clicking the **Roll** button, the player comes across the following screen:



The player's roll is shown here, and the player's position on the map is updated accordingly. The player may choose to initiate other actions after rolling the dice.

3.3.3 Restart

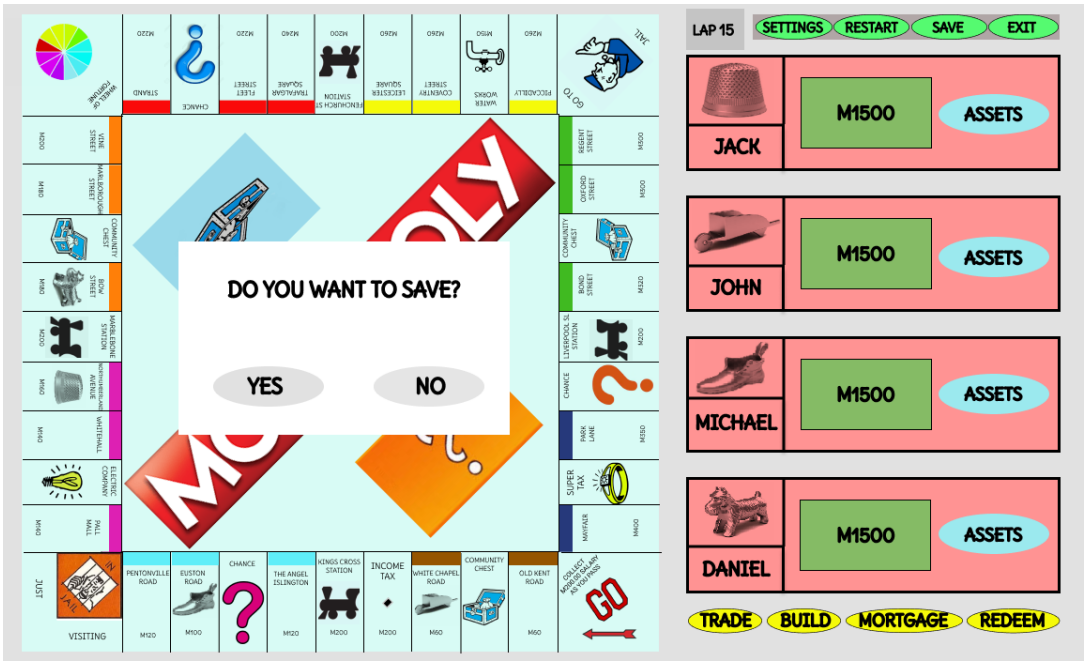
Pressing the **Restart** button at the top right corner leads the player to the following screen:



Players may choose to restart, or cancel their decision and continue playing the game. In case the **No** option is chosen, the game will resume normally. Choosing **Yes** restarts the game, where every player has the initial conditions that of a new game.

3.3.4 Save

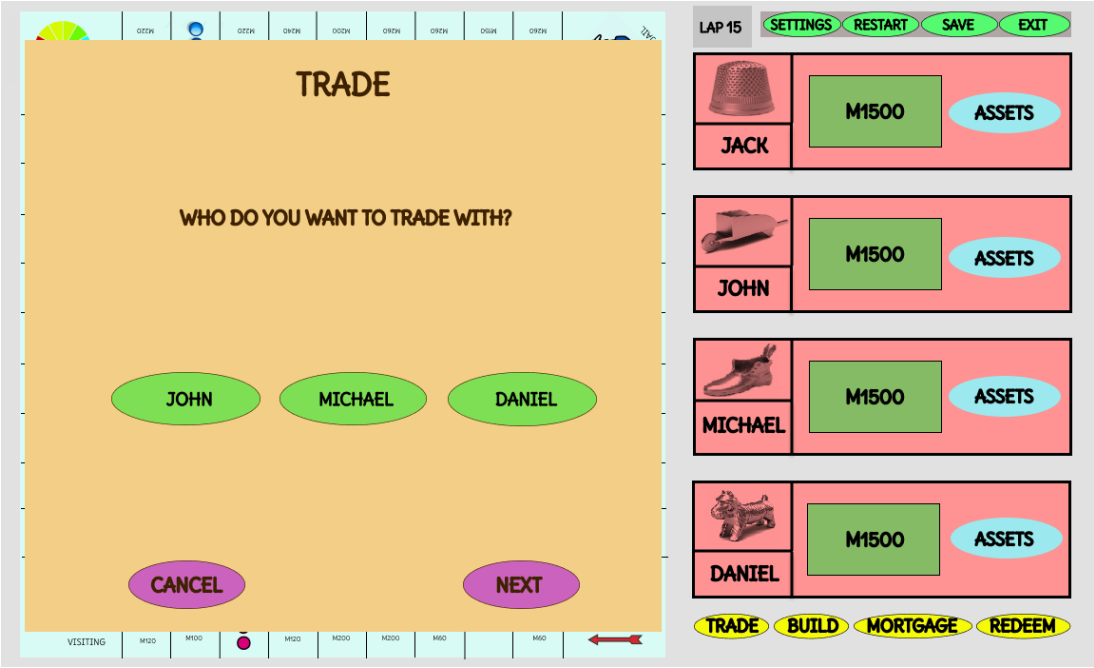
Pressing the **Save** button at the top right corner leads the player to the following screen:



Players can choose to save the game manually, which enables the game to be accessed in the **Load Game** option. The assets, names and the positions of the players are stored in case the game closes and the players can access the version of the game that was saved manually from the **Load Game** option. Choosing not to save the game will allow the game to resume normally.

3.3.5 Trade (Phase 1)

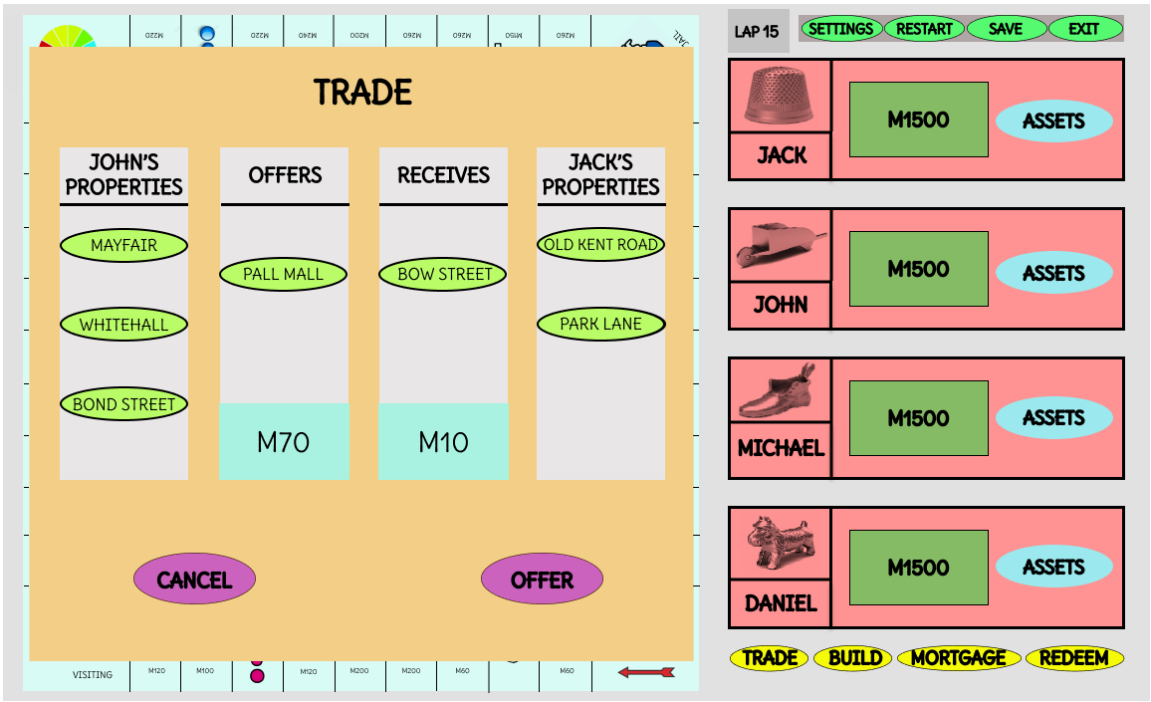
Pressing the Trade button at the bottom right corner leads the player to the following screen:



Here, players can choose the player that they want to trade with, and press the **Next** button to confirm and initialize the trade.

3.3.6 Trade (Phase 2)

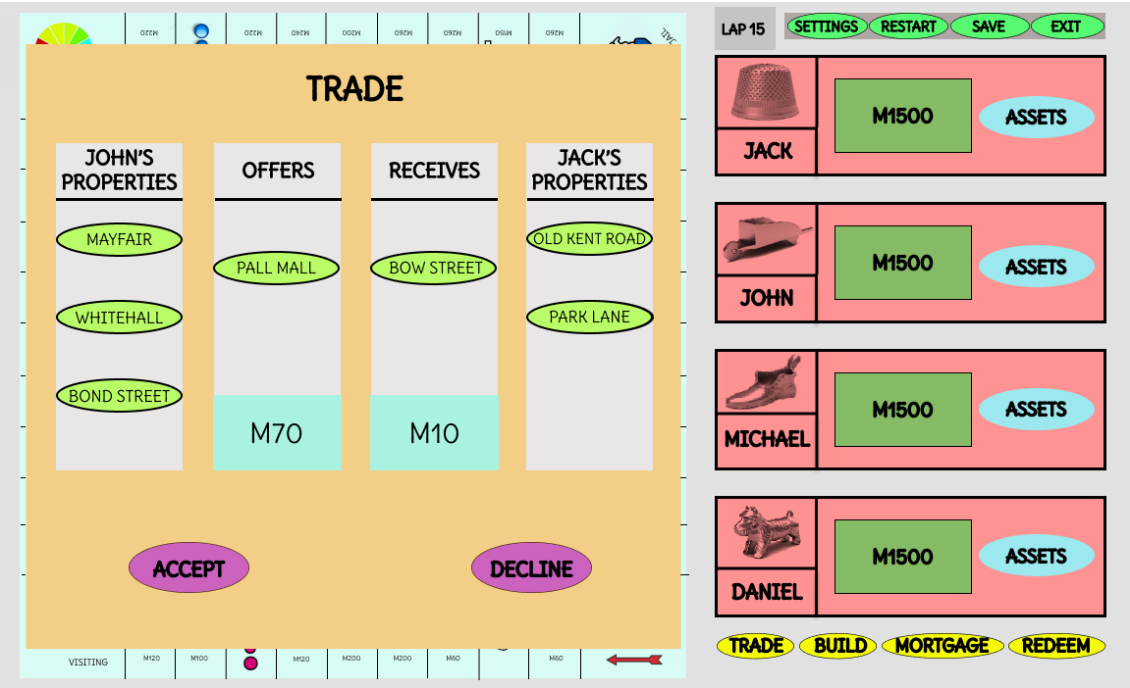
Clicking the **Next** button after choosing the player to trade with brings the players to the following screen:



The players then are able to put their assets on the Offering or Receiving end to show the places they are willing to trade with. Clicking **Offer** will finalize the offering stage and offer the current assets to the other player.

3.3.7 Trade (Phase 3)

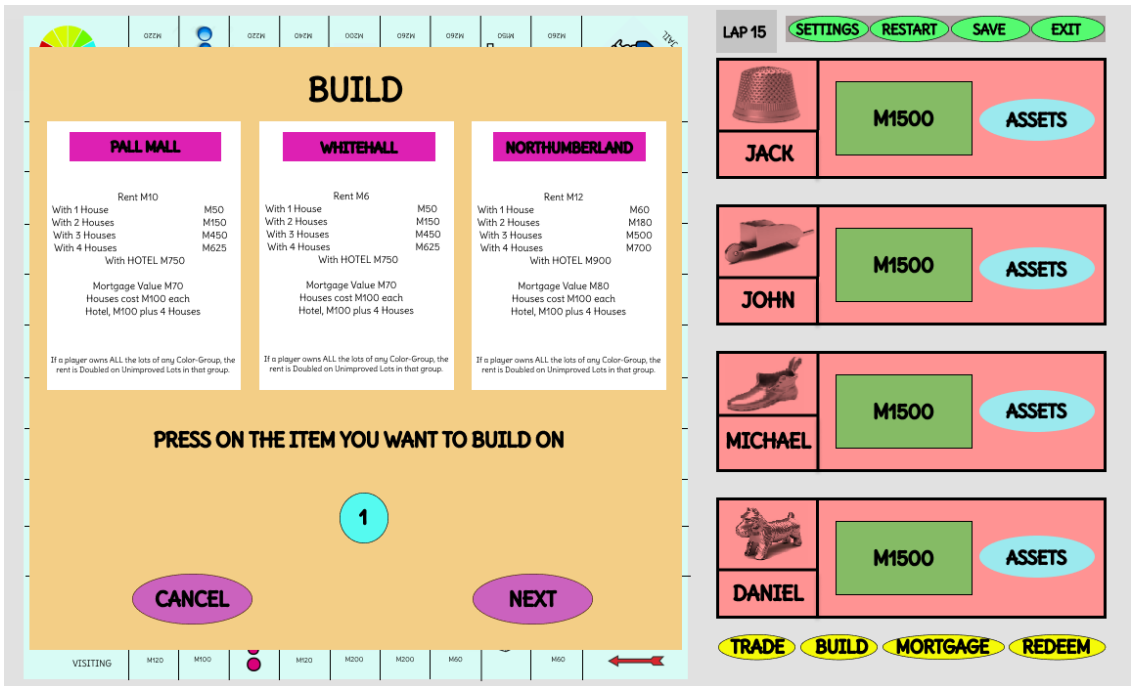
After the Offer button is pressed, meaning the offer is finalized, the other player who is at the trade sees the following screen:



The offer receiving player then can choose to accept the offer, or decline it by the buttons below the Trade page, finishing the trade.

3.3.8 Build (Phase 1)

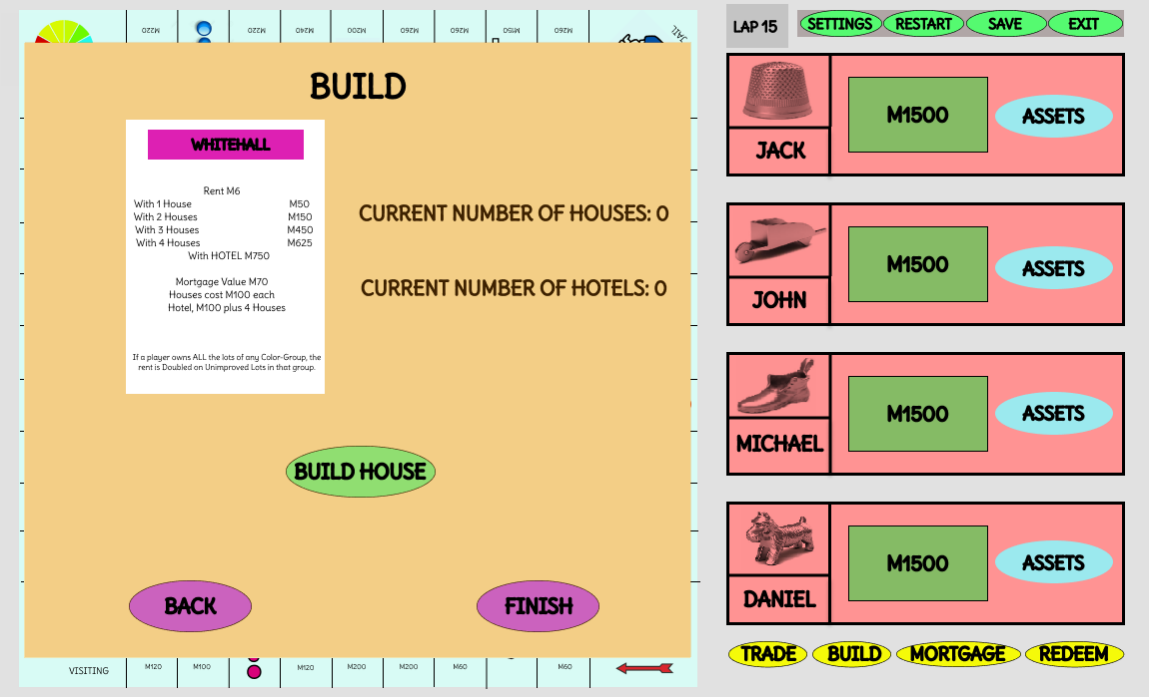
Pressing the **Build** button at the bottom right corner leads the player to the following screen:



The Build feature is for the players to build some kind of lodgement on their lands, which are the assets. Firstly, the player has to choose the asset she/he wants to build on. After choosing the asset, the player can click the **Next** button to initiate the building. Similarly, to cancel the Build function, the **Cancel** button should be pressed.

3.3.9 Build (Phase 2)

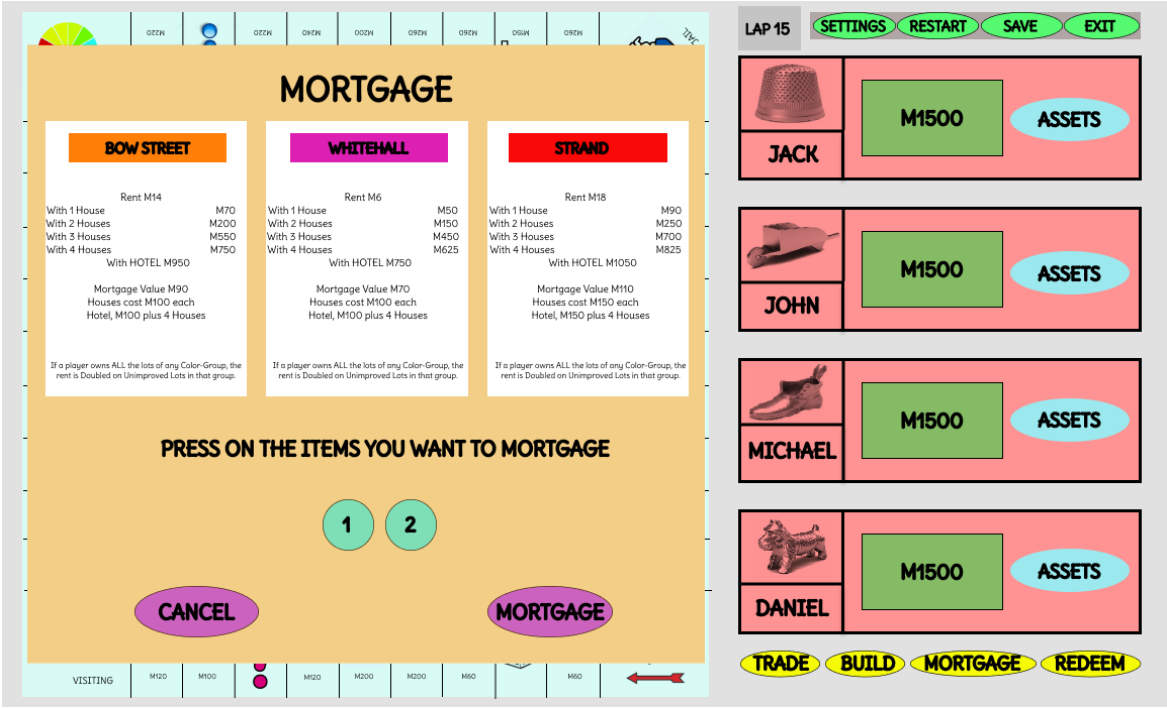
The player reaches the second phase of the build feature by pressing the Next button in the first phase.



In this stage, the player can build houses on the chosen property

3.3.10 Mortgage

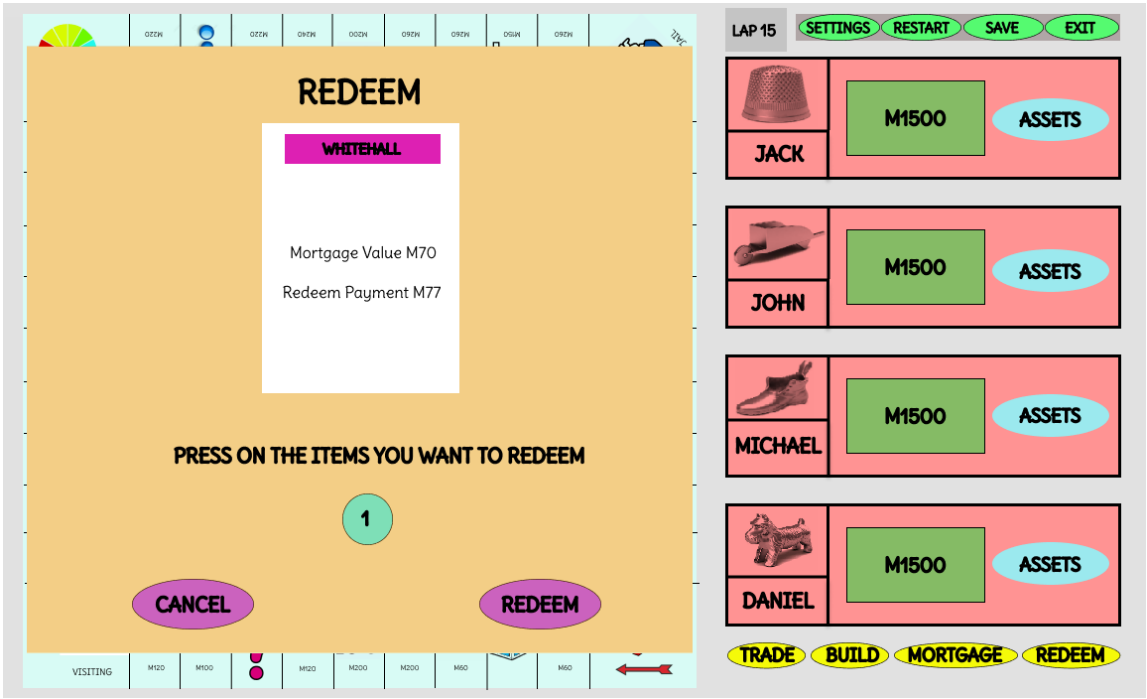
Pressing the **Mortgage** button at the bottom right leads the player to the following screen:



The player is able to mortgage her/his items in this screen. She/he first chooses the items, then clicks the **Mortgage** button to finish.

3.3.11 Redeem

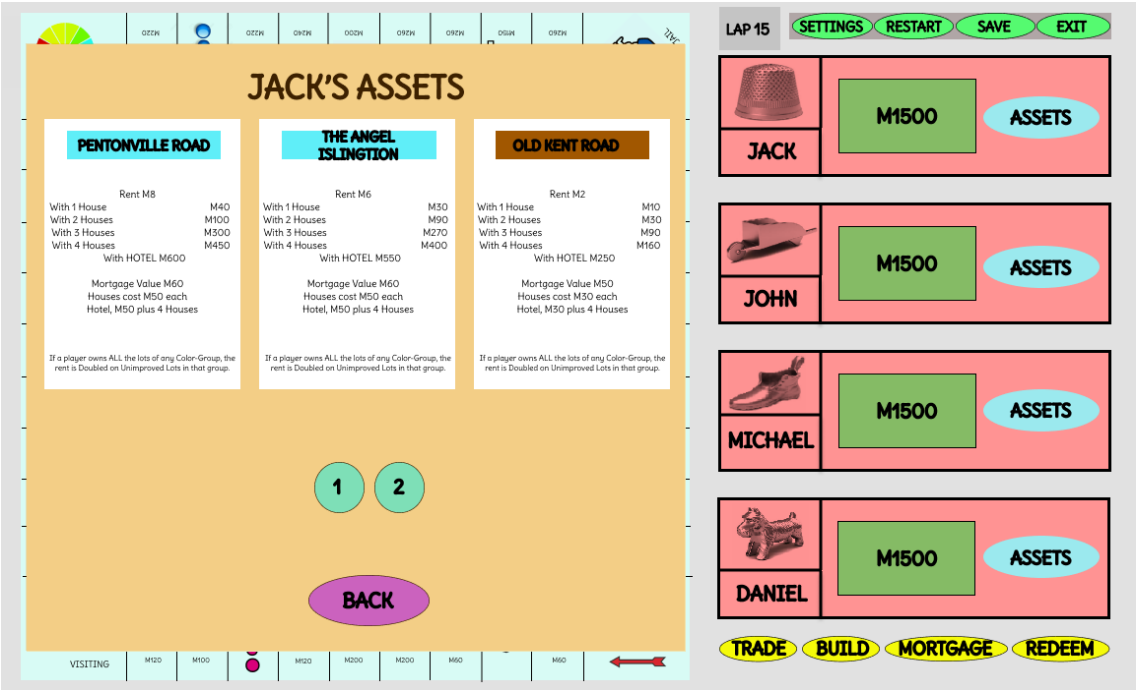
Pressing the **Redeem** button at the bottom right corner leads the player to the following screen:



The player can recover possession of an item by choosing it and pressing the **Redeem** button.

3.3.12 Assets

Pressing the **Assets** button at the players' information section leads the player to the following screen:

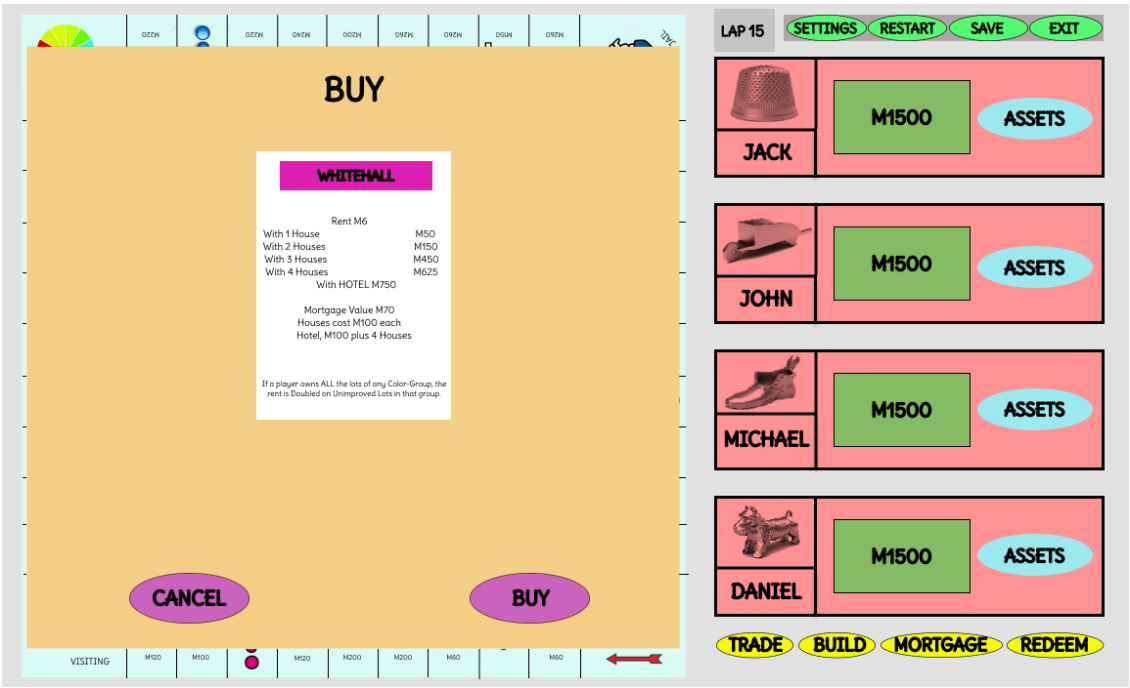


The screen shows the assets of the chosen player.

3.4 Additional In-Game Actions

3.4.1 Buying on the Land

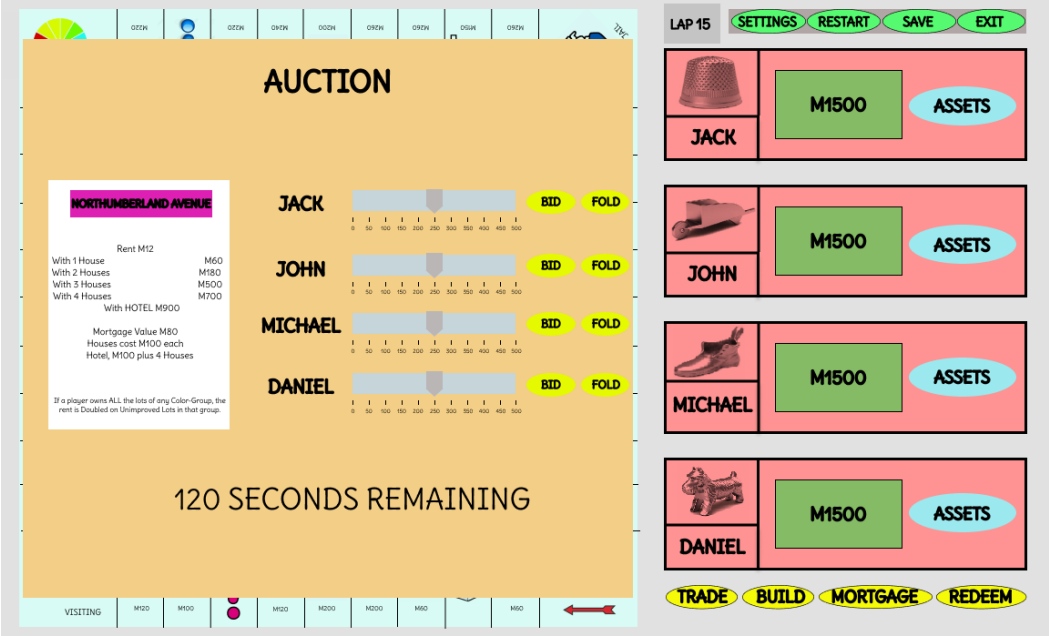
After rolling the dice, the player may or may not want to buy the land that she/he has landed on. In case that the player wants the land, buying is initiated and the player is shown the following screen:



The player can buy the item by clicking the **Buy** button.

3.4.2 Auctioning

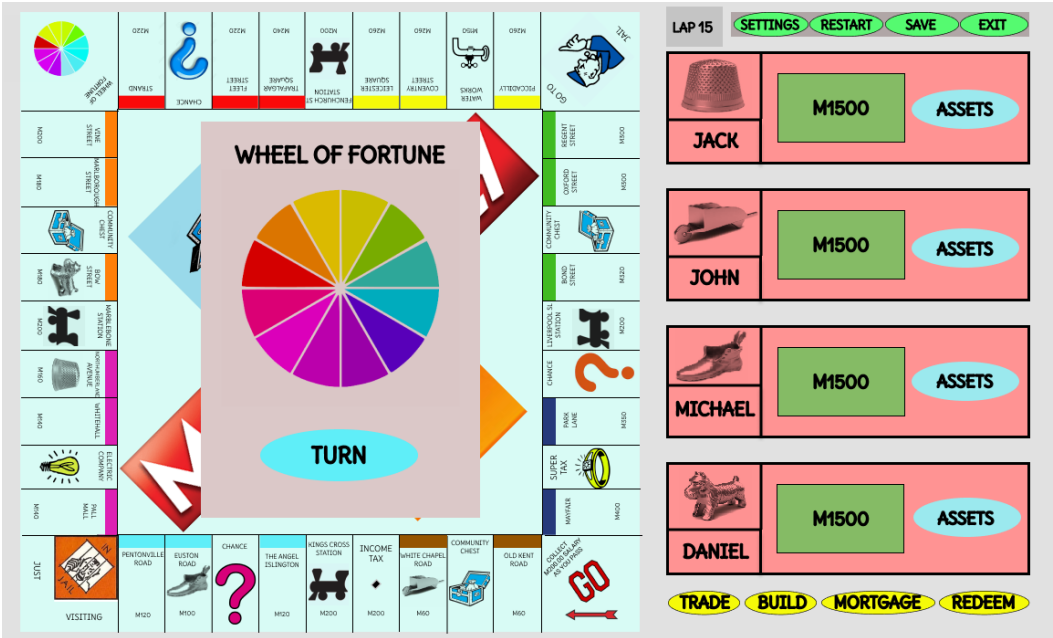
Assets can be auctioned in the game. This is the screen shown while auctioning a piece of land.



All 4 players participate in the auction, and may choose to fold at any time. The player that does not fold until the end gains the possession of the auctioned asset. Players are able to give up the auction by pressing the **Fold** button. Likewise, the **Bid** button allows the participants to bid and raise the prize of the auctioned item.

3.4.3 Wheel of Fortune

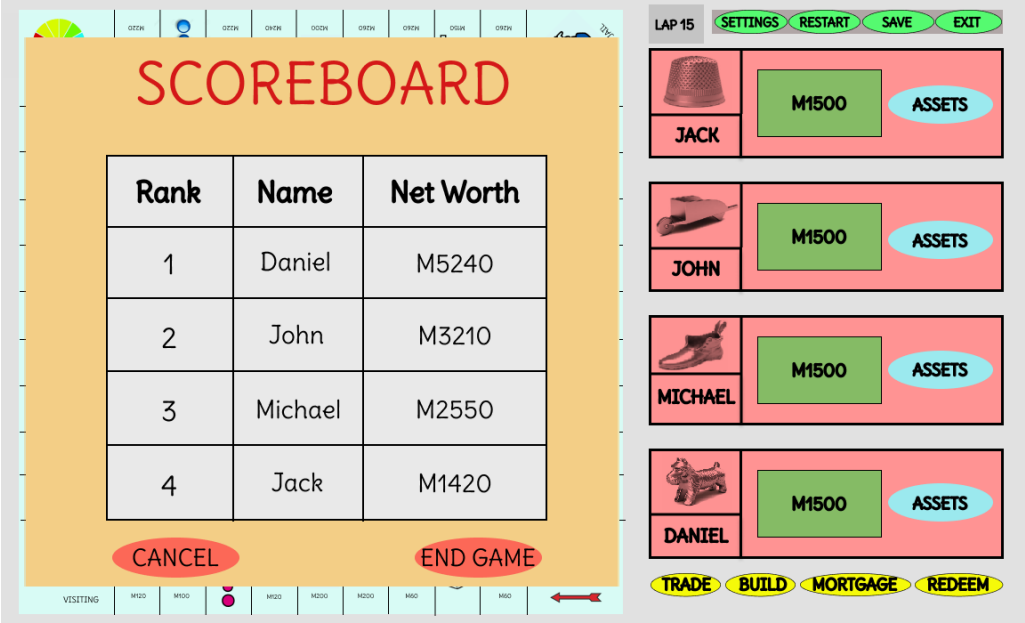
The Wheel of Fortune is a special case where players land on a specific position on the board. The specific places activate the Wheel of Fortune and players see the following screen:



Clicking the **Turn** button initiates the spinning of the wheel of fortune, which provides a random effect to the player activating the wheel.

3.4.4 End of the Game and the Scoreboard

When the game finishes, the players can see the scoreboard and end the game.



4. Build Instructions

4.1 System Requirements

Our game was implemented in Java so the user needs to have a Java Virtual Machine (JWM) on their computer to play the game.

4.2 Installation

In order to play the game, the users must download the jar file from the link:

https://github.com/elifKurtay/CS319_1A_Monopoly/

After downloading and opening the jar file, the user can play the game.

Build Instructions

In order to download, compile and run the game, the following instructions are needed to be followed. Make sure you satisfy the preconditions before starting to follow the instructions. Note that the

instructions are given for building on IntelliJ IDEA Community 2020.3 Edition, which is the latest version currently.

Preconditions:

- Have IntelliJ IDEA editor 2020.3 version installed on your computer. (Eclipse will also work, but note that these instructions are only written for IntelliJ IDEA. Addition of the Lombok plugin in Eclipse is different.)
- Have Git installed on your computer to clone the remote Git repository into your IDE.
- Have Oracle's java jdk 8 installed on your computer. We use the JavaFX library in our project, any jdk that is jdk 11 or higher will not include this library and you might need to add it yourself. OpenJDK also does not include the JavaFX library. Therefore, installing an OpenJDK distribution that has the JavaFX library might be required. A link for the advised sdk is given:

oracle jdk-8u271: https://www.oracle.com/tr/java/technologies/javase/javase-jdk8-downloads.html

Instructions:

1. Open IntelliJ editor. Select to create a new project from Version Control.
2. Choose Git for the Version Control system.
3. Go to the Github repository of this project and get the link for cloning the project. Clone link is also put here:

   https://github.com/elifKurtay/CS319_1A_Monopoly.git

4. Choose the directory to create the project and click on the "Clone" button. The IntelliJ IDEA and Git will then download the files from the remote repository and create a new project with the files.
5. Go to File -> Project Structure, under Project Settings click on the "Project" section. Choose the SDK for the project. The java sdk needed is the jdk 1.8 for this project. You can show the location of this folder for the editor to find it if required. If you do not have the 1.8 java version, please refer to preconditions of build instructions.
6. In File -> Project Structure -> Project Settings, any language level above 8 will not create a problem.
7. In File -> Project Structure -> Project Settings, please select a directory for project compile output as well.
8. Go to File-> Settings -> Plugins, in Marketplace search for "Lombok". When found, install. To make sure of installation, you can check File -> Project Structure -> Project Settings -> Modules -> Dependencies and see the lombok dependency.
9. This step is required for IntelliJ IDEA 2020.3 and later. If you have an earlier version than 2020.3, this line might not be required. After completing the instruction and building the project, if there are errors found with getter or setter methods during compilation, you might need to refer to this step as well. Go to File-> Settings, under "Build, Execution, Deployment" category click on the "Compiler" section. Add the following in the "Shared build process VM options:" field:

   -Djps.track.ap.dependencies=false

This is required for IntelliJ IDEA 2020.3 and later because a recent bug is introduced in this very new version that prevents Lombok from running with the compiler.

10. In File-> Settings -> Compiler -> Annotation Processors, click on enable annotation processing.
11. Click Build.
12. Run src/Main.java.

Custom Map and Custom Chance/Community Chest Cards:

A custom map can be created by adding these required fields inside a JSON object and saved as a JSON file. A map example can be found on our git repository.

Required Fields:

"mapName" : String
Name of the map

"propertyGroupCount": integer

How many property groups are there including transport and utility properties, so 8 (land properties) + 2 (utility + transport) for the default London map.

"numberOfPropertiesInGroups": Integer Array

Number of properties in each of the property groups, [2,3,3,3,3,3,3,2,4,2] for the default London map, last two being utility and transport

"propertyGroupColors" : String Array

Hex code of each of the land property colors, transport and utility properties do not have color codes. ["955500","5feef8","dd20b3","ff7f08","fa0909","f5fa09","40ba21","27397b"] For the deafult London map.

"spaces": JSON object array
Holds all of the spaces on the map
Spaces consists of the 4 required special spaces in the specified places, and the rest of the spaces consists of common spaces.

"cards": JSON object array, holds two JSON object arrays "chanceCards" and "communityChestCards"
Holds all of the chance and community chest cards

Special Spaces:

GO Space:

```
{
        "type": "GoSpace",
        "name": "Go"
}
```

GO space is a special space and must be at the index 0 of the spaces array, and it will be the first space (first corner) on the board.

Jail Space:

```
{
        "type": "JailSpace",
        "name": "Jail"
}
```

Jail space is another special space and must be at the index 10 of the spaces array, and it will be the 10th space (second corner) on the board.

Wheel of Fortune Space:

```
{
        "type": "WheelOfFortuneSpace",
        "name": "Wheel Of Fortune"
}
```

Wheel of Fortune space is another special space and must be at the index 20 of the spaces array, and it will be the 20th space (third corner) on the board.

Go To Jail Space:

```
{
        "type": "GoToJailSpace",
```

```
    "name": "Go To Jail"
}
```

Go To Jail Space is another special space and must be at the index 30 of the spaces array, and it will be the 30th space (fourth corner) on the board.

Common Spaces:

Land Property Space:

```
{
    "type": "PropertySpace",
    "propertyType": "LAND",
    "name": "Old Kent Road",
    "value": "60",
    "propertyGroup": 0,
    "titleDeedCard": {
       "rents": [2, 10, 30, 90, 160, 250],
       "houseCost": 50,
       "hotelCost": 50,
       "mortgageValue": 30
    }
}
```

A land property space is a common space, it has the type "PropertySpace", propertyType "LAND", name of the property, the buying value of the property, the propertyGroup number it belongs to and the title deed card object it is associated with.

The title deed card object holds the rents, house cost, hotel cost and the mortgage value of the property.

Rents is an array of 6 numbers, the rent without any houses, the rent without any houses but all properties from the same group, the rent with 1 house, the rent with 2 houses, the rent with 3 houses, the rent with 4 houses and the rent with a hotel respectively.

Transport Property Space:

```
{
    "type": "PropertySpace",
    "propertyType": "TRANSPORT",
    "name": "King's Cross Station",
    "value": "200",
    "propertyGroup": 8,
    "titleDeedCard": {
       "rents": [25, 50, 100, 200],
       "mortgageValue": 100
    }
}
```

A transport property space is a common space, it has the type "PropertySpace", propertyType "TRANSPORT", name of the property, the buying value of the property, the propertyGroup number it belongs to and the title deed card object it is associated with.

The title deed card object holds the rents and the mortgage value of the property.

Rents is an array of 4 numbers, the rent when you own 1 transport property, the rent when you own 2 transport properties, the rent when you own 3 transport properties and the rent when you own 4 transport properties respectively.

Transport property group numbers should always come after Land property group numbers, and before the Utility property group numbers.

There should always be at maximum 4 transport properties on the map

Utility Property Space

```
{
        "type": "PropertySpace",
        "name": "Electric Company",
        "propertyType": "UTILITY",
        "propertyGroup": 9,
        "value": "150",
        "titleDeedCard": {
           "multipliers": [4, 10],
           "mortgageValue": 75
        }
}
```

A utility property space is a common space, it has the type "PropertySpace", propertyType "UTILITY", name of the property, the buying value of the property, the propertyGroup number it belongs to and the title deed card object it is associated with.
The title deed card object holds the rents and the mortgage value of the property.
Rents is an array of 2 numbers, the rent when you own 1 utility property, the rent when you own 2 utility properties respectively.
Transport property group numbers should always come after Transport property group numbers.
There should always be at maximum 2 utility properties on the map.

Card Space:

```
{
        "type": "CardSpace",
        "name": "Chance",
        "cardType": "CHANCE"
}
```

A card space is a common space, it has the type "CardSpace", cardType and name of the space.
cardType can either be "CHANCE" or "COMMUNITY_CHEST"
There is no limit on how many Card spaces can exist on the map as long as the required spaces are on the map.

Tax Space:

```
{
        "type": "TaxSpace",
        "name": "Luxury Tax",
        "taxType": "LUXURY"
}
```

A tax space is a common space, it has the type "TaxSpace", taxType and name of the space.
taxType can either be "LUXURY" or "INCOME"
There is no limit on how many Tax spaces can exist on the map as long as the required spaces are on the map.

Cards:

Cards have a cardEvent and cardText, cardEvent has a type that specifies what kind of card event it is going to do

Thief Card:

```
{
        "cardEvent": {
           "type": "THIEF"
        },
        "cardText": "Deploy Thief Targeting: "
}
```

Deploys the thief on the board.

Advance Card:

```
{
            "cardEvent": {
               "type": "ADVANCE",
               "targetSpace": "Go",
               "canCollectSalary": true
            },
            "cardText": "Advance to \"Go\" (Collect M200)"
},
```

Advances the player to the specified targetSpace. The player collects salary if they pass go and canCollectSalary is true.

Advance Spaces Card:

```
{
            "cardEvent": {
               "type": "ADVANCE_SPACES",
               "moveAmount": -3,
```

```
            "canCollectSalary": true
        },
        "cardText": "Go Back Three Spaces."
}
```

Similar to Advance cards but can make a player move a specified amount of spaces. The player collects salary if they pass go and canCollectSalary is true.

Receive Get Out of Jail Free Card:

```
{
        "cardEvent": {
            "type": "RECEIVE_GET_OUT_OF_JAIL_CARD"
        },
        "cardText": "Get out of Jail Free. This card may be kept until needed, or traded/sold."
}
```

The player receives one GOOJ card.

Collect Card:

```
{
        "cardEvent": {
            "type": "COLLECT",
            "from": "BANK",
            "amount": 50
        },
        "cardText": "Collect M50 from the bank."
}
```

The player collects 50 from either the bank or other players still in the game.
"from": can be either "BANK" or "PLAYERS"
"amount": specifies the amount the player is going to collect from the bank or other players

Pay Card:

```
{
        "cardEvent": {
            "type": "PAY",
            "to": "PLAYERS",
            "amount": 50
        },
        "cardText": "You have been elected Chairman of the Board. Pay each player M50"
}
```

The player pays 50 to either the bank or other players still in the game.
"to": can be eiter "BANK" or "PLAYERS"
"amount": specifies the amount the player is going to pay to the bank or other players

Pay Per Building Event:


```
{
        "cardEvent": {
           "type": "PAY_PER_BUILDING",
           "to": "BANK",
           "amount": [25, 100]
        },
        "cardText": "Make general repairs on all your property: For each house pay M25, For each hotel pay M100."
}
```


The player pays the specified amounts per house and hotel to the bank.
"amount": The first number is the cost per house, the second number is the cost per hotel.

Go To Jail Card:


```
{
        "cardEvent": {
           "type": "GO_TO_JAIL",
           "targetSpace": "Jail",
           "canCollectSalary": false
        },
        "cardText": "Go to Jail. Go directly to Jail. Do not pass GO, do not collect M200."
}
```


Sends the player to the jail. The player collects salary if they pass go and canCollectSalary is true.