Bilkent University

Department of Computer Engineering

CS 319

Object-Oriented Software Engineering

Term Project

Section 2

Group 2F

Monopoly

Analysis Report

Project Group Members

Adil Meriç	21802660
Burak Öçalan	21703769
Doğa Tansel	21802917
Osman Batur İnce	21802609
Ömer Faruk Akgül	21703163

Supervisor: Eray Tüzün

Contents

1. INTRODUCTION

You and three or more adversaries are trying to expand your own business empires. You acquire and sell property, exchange money for districts that you desire. You invest in infrastructure and capitalize your districts, making it harder for your opponents to build when they are busy paying you rent. Slowly and with strategy, you drive your opponents to bankruptcy and take the lead in your area, as you expand your power over multiple districts including your opponents'. Finally, when you drive all your opponents to bankruptcy, you will have monopolized and conquered your area.

Monopoly is a very popular, turn-based strategy game based on capital investment. The purpose of each player is to drive their opponents to bankruptcy by investing heavily on the districts they own, which makes their opponents lose money and therefore buying power by paying rent. The game is simple yet very well-suited for object oriented design. The game also is very easy to customize, which allowed us to add our own features. The game follows the classic rules, but there have been a few modifications for adapting the game for computers. We have also added our own game modes for variety and choice:

- Blitz
- Reverse

We will write the game in Java. As of right now we have not decided on which platform we will use to write the code. However, we will, as much as we can, try to use every bit of knowledge that we learn from our CS - 319 lectures and our instructors.

2. OVERVIEW

2.1. Game Components

To play the game, a few things are needed:

- 3-8 players
- Tokens to represent each player
- A pair of dice
- Property, utility and railroad type cards giving information about those spaces

- Community chest and chance cards
- Houses and hotels
- Money
- Map(s) of 32 slots including
    - 20 districts/properties
    - 3 railroads/canals/stations
    - 2 chance stops
    - 2 community chests
    - 1 tax administration
    - a jail entry
    - a jail
    - a free parking space
    - and the GO space

In addition, there are a few extra features that can also be used:

- 3 different game modes
    - Classic
    - Blitz
    - Reverse
- Game saving/loading
- Mode and map editor
- User interface settings
- Game manual and rules
- Credits

2.2. Player

Monopoly is a game to be played with multiple people. Since there are many components on the map that can be easily bought, sold, and traded by players, it is easy for players to bankrupt other players. The game cannot be played with only two players because usually one of the two players gets bankrupt pretty quickly and the game does not last very long. Since we intend for monopoly not to be very short or easy, we decided that at least 3 players must be

present for the game to be played. The players will all first decide their turns by dice and then take their turns during the game, buying and selling properties to bankrupt the other players and become the winner.

2.3. Main Menu

The game can be played and altered using the main menu. Games can be initialized, customized, saved and loaded from the play game button. Different game modes can be selected through options. Sound effect levels, music levels and visuals can also be changed through options. We also added a button for a mode editor, allowing the players to edit their maps, properties and prices using the three base game modes.

2.4. Maps

Our game includes three default maps for classic mode and one for the other two modes. The default maps are Turkey Cities, World Cities and Bilkent.

The Turkey Cities Map comprises 20 properties, 3 railroads, 2 chance slots, 2 community chests, 1 tax administration, a jail entry, a jail, a free parking lot and a start (GO) slot. There are 32 slots in total. (The same types of areas will be referred to as their original names in the game such as property, community chest, etc.).

The World Cities Map comprises 20 cities (property type), 3 canals (railroad type), 2 chance slots, 2 community chests, 1 tax administration, a jail entry, a jail, a free parking lot and a start (GO) slot. There are 32 slots in total.

The Bilkent Map comprises 20 buildings (property type), 3 cafeterias and restaurants (railroad type), 2 chance slots, 2 community chests, 1 tax administration, a discipline entry (jail entry type), a discipline center (jail type), a free parking lot and a start (GO) slot. There are 32 slots in total.

2.5. Instructions

Players can look at the instructions and rules before or during the game through the menu. These tell the players what guidelines they have to follow to play the game correctly and reach a conclusion.

2.6. Changes in Gameplay

In our game we decided to make a couple of changes to the rules or the gameplay, as we didn't see some of the original rules or locations on the maps fit for an implementation of the game for computers. Here are the following changes:

- We have 3 railroads instead of 4 on the board.
- The number community chests and chance stops have both decreased from 3 to 2.
- Tax administrations have been lowered to 1.
- There are now only 5 properties on each edge of the board.
- In the original game, when a player goes bankrupt, the bank immediately auctions all properties to the highest bidder, so it remains owned. In our game, we removed auctions (as playing from one computer makes it useless to have properly functioning auctions). Therefore, when a player goes bankrupt to the bank, all of his properties become unowned. Before going bankrupt, the player mortgages every property to pay his debts.
- We have also removed the utilities (such as electrical utility) as we thought they are unnecessary and make the implementation of the game more difficult.
- In the community chest and chance cards, the cards are shuffled every time they are to be used, which is not done in the original game where a card is taken from the top of the stack.
- Additionally, we added ways to modify and change maps and locations as well a few game modes which are not part of the original game.

2.7. Improvements of the Analysis

Since our first version of our analysis report, we have received feedback on what we can improve in our analysis stage of our project. We have made the following changes and improvements:

- We have shortened our list functional requirements in order to explain what happens in the game rather than how those things happen. Their explanations have been made in the appropriate places of the report.
- Small changes as well as a “changes in gameplay” part have been added to the overview section.
- The use case diagram has been improved as there were some problematic parts at iteration 1. There was a flow between use cases in the previous phase. Also, some use cases were missing. These are solved and the diagram is updated.
- The sequence diagrams have also been improved and specified to be more accurate and comprehensive. Interactions of building a house/hotel is displayed using sequence diagram. The method/parameter names are updated as there has been changes in class diagram. All the recommendations in the feedback are applied.
- We have also changed our order determination state diagram and made it into an activity diagram which was suggested in our feedback. We have also made 4 new state diagrams which are comprehensive as well.
- Finally, we have made the necessary additions and changes to our class diagram. We had some missing functionalities at phase 1. To add these functionalities classes and their relations are updated.

3. FUNCTIONAL REQUIREMENTS

3.1. Game Setup

After opening the game, a user must be able to do the following:

- Read the user manual with the instructions and rules (how to play)
- Adjust sound effects and music levels
- Display credits
- Exit the game
- Start the game with the following initialization options.

3.2. Game Initialization

The game must have some default settings. If a user wants to start a Monopoly game, they must be able to start the game with the default settings or change these settings before the game starts. To change the game settings, the user must be able to do the following:

- Select a mode among the game's three modes (classic, blitz, reverse)
- Select a number between 3 and 8 (3 and 8 are included) which represents the number of players in the game
- Change players' starting money (note that each player must start with the same amount of money)
- Change players' total time (only in Blitz mode) (note that each player must start with the same amount of total time)
- Select a map among various default maps (only in Classical mode) (note that there is only one default map for modes other than Classical mode)
- Modify the selected map by changing the locations' names and prices (details are specified in Mode Editor)
- Save the initialization settings for further use

- Start the game with these initialization settings

As a last step before the game starts, each player must be able to the following:

- Change their nicknames
- Select a token among the set of unselected tokens.

Also before the game starts, the application must be able to do the following:

- Let players roll a dice and
- Determine the turn order of players according to the results of rolled dice.
- Give the turn to the first player in the determined turn order while beginning.
- Shuffle the card packs, which will be described later

3.3. Game İnterface

After the game starts, in any given time before the game ends, each user must be able to to do the following:

- See the game board, the tokens and the dice
- Get informed about each player's current status (nickname, money, remaining time - only in Blitz mode)
- Get informed about each location's current status, if any (rent, price, owner, etc.)
- Pause the game
- Leave the game
- Save the current game

3.4. Player Turn

When a player takes the turn, this player is called as the "current player". The current player must be able to do the following:

- Roll the dice

- Move their token among the locations on the board according to the result of the rolled dice
- Take some amount of money from the bank if they pass the “GO” location during their movement
- Land on a location on the board as a result of the movement described above
- Take action according to the location that he lands. These actions will be described in detail later sections.
- Roll the dice and perform the other previous steps again if the previous roll is a double
- End his turn
- Lose if their money drops below zero and they can’t sell anything

3.5. Property

Properties are one of the location types in the game. Like any other location, the current player must be able to land on a property. Current player must also be able to do the following:

- Determine whether a property is owned or not
- Determine who owns a property if it is owned
- Purchase an unowned property that he lands by paying the price of the property to the bank
- Not to purchase an unowned property that he lands
- Pay the rent of an owned property that he lands to the owner of the property
- Charge rent from the other players who land on his properties
- Have a monopoly by owning all properties of the same color
- Charge double rent for his monopolized properties
- Build houses and hotels, which will be referred to as buildings in general, on the monopolized properties which increases the property’s rent. Note that at most 4 houses or 1 hotel can be built to a property
- Sell a building on a property that he owns to the bank for half of the building price
- Mortgage a property that he owns, which means that all buildings on the property is sold to the Bank for half their original prices and no rent will be acquired from the property

- Unmortgage a mortgaged property that he owns by paying the price with %10 interest
- Buy any property owned by another player by paying the price determined with the negotiations between current player and owner
- Sell a property that he owns to another player and take the corresponding price from the player

There are multiple maps with different properties such as cities of Turkey and buildings of Bilkent (Appendix A).

3.6. Railroads

Railroads are one of the location types in the game. Most of the functional requirements for railroads are similar to those of properties. However, they will be repeated for convenience. Like any other location, the current player must be able to land on a railroad. Current player must also be able to do the following:

- Determine whether a railroad is owned or not
- Determine who owns a railroad if it is owned
- Purchase an unowned railroad that he lands by paying the price of the property to the bank
- Not to purchase an unowned railroad that he lands
- Pay the rent of an owned railroad that he lands to the owner of the railroad
- Charge rent from the other players who lands on his railroads
- Charge increased rent according to the number of railroads he owns
- Sell a railroad that he owns to another player and take the corresponding price from the player

There are multiple maps with different types of railroads (Appendix B).

3.7. Jail

Jail is one of the location types in the game. The functional requirements about jail are given below. Current player must be able to do the following:

- Go to jail by either landing on the “Go to Jail” location, drawing the “Go to Jail” card or rolling three consecutive doubles
- Move his token to the “Jail” location on the board as a result of the previous action
- Collect rent from another player in the jail
- Not to buy or sell buildings or properties in the jail
- Get out of jail by either using a “Get out of jail free” card, throwing a double which allows the player to move the amount rolled but not roll again, or by waiting three turns and paying a $50 fine.
- Roll the dice and move in the same turn that he paid $50 fine and get out of jail

3.8. Community Chest

Community Chest is one of the location types in the game. Like any other location, the current player must be able to land on a Community Chest. After landing on a Community Chest, current player must also be able to do the following:

- Draw a Community Chest Card from the card pack
- Act according to what is written on the drawn card
- lace the drawn card on the bottom of the deck

The functional requirements below describe the actions that the current player can perform according to the card he draws. After picking up a card, current player must be able to do the following:

- Pay some amount of money
- Receive some amount of money from the bank
- Obtain freedom right from jail
- Advance to starting point
- Advance to jail

There are many different options in the community chest cards (Appendix C).

3.9. Chance

Chance is one of the location types in the game. Most of the functional requirements for Chance are similar to those of Community Chest. However, they will be repeated for convenience. Like any other location, the current player must be able to land on a Chance. After landing on a Chance, Current player must also be able to do the following:

- Draw a Chance Card from the card pack
- Act according to what is written on the drawn card
- Place the drawn card on the bottom of the deck

The functional requirements below describe the actions that the current player can perform according to the card he draws. After picking up a card, current player must be able to do the following:

- Pay some amount of money
- Receive some amount of money from the bank
- Advance to Go
- Advance to nearest Railroad
- Advance to Jail

There are many different options in the community chest cards (Appendix D).

3.10. Tax

Community Chest is one of the location types in the game. Like any other location, the current player must be able to land on a Community Chest. After landing on a Community Chest location, the current player must also be able to pay the given tax.

3.11. Game End

The game must be able to do the following:

- Ensure that the players who lost don’t take turn again
- End itself if there are less than or equal to one player standing

After the game ends, players must be able to do the following:

- See the leaderboard
- Go back to the main menu

3.12. Modes

We have three different modes: Classic, Blitz, Reverse. All functional requirements given above apply to all three modes. Blitz and Reverse has also some extra functional requirements, which are given below.

3.12.1. Blitz

In this mode, the game must be able to do the following:

- İnitialize each player with some limited amount of time at the start of the game
- Display the remaining times of each player at any given time
- Decrease the remaining time of the current player
- Remove the player from the game if his time runs out

3.12.2. Reverse

In this mode, the board must have Reverse locations in place of the Community Chest locations. Like any other location, the current player must be able to land on a Reverse. After the player lands on a Reverse, the game must be able to do the following:

- Reverse the direction flow of the game, which means that all of the tokens will move in the negative direction.

4. NONFUNCTIONAL REQUIREMENTS

4.1. User Interface and Human Factors

- ➔ Any human who is bigger than 7 years old and has played at least one other computer game that has a graphical user interface should be able to play this game.
- ➔ Users who have played the monopoly box game before and have used a computer several times should be able to understand the game in less than 20 minutes, without reading the user manual.
- ➔ Users who haven't played the monopoly box game before should be able to understand the game in less than 3 hours, with reading the user manual and practicing.
- ➔ Users should not be able to crash the game by only using the user interface provided by the game.
- ➔ If the game is working on a user's computer, the user must be able to play the game with only a working mouse and a screen which is capable of fully monitoring the game.

4.2. Documentation

- ➔ A "How To Play" document should be available for the users to help them understand how to play the game
- ➔ Documentation of the source code of the game should be available for the future maintainers of the game to help them understand the structure of the code

4.3. Hardware Considerations

- ➔ Any computer with a recommended hardware which is given below, should be able to run the game.
- ➔ Recommended Hardware:
- ➔ CPU: SSE2 compatible instruction set with clock frequency at least 1.5 GHz
- ➔ RAM: 512 MB
- ➔ Video Card: 256 MB VRAM with Shader Model 3.0 or better
- ➔ Free Disk Space: 150 MB

4.4. Performance Characteristics

- ➔ If the game is played with a computer that has recommended hardware, the game should start in less than 1 second.
- ➔ With the same computer above, the game should respond to any of the user actions in less than 0.25 seconds.

4.5. Error Handling and Extreme Conditions

- ➔ Game should ignore all error prone actions from the user and send a warning message.
- ➔ The crash rate of the game must be lower than 0.1%.
- ➔ In at least %50 percent of crashes, the game should save the current progres. After saving, the game should close itself with an error message.
- ➔ Even if the game can't save the current progress of the game in a crash condition, users should be able to continue from the previous turn after restarting the game.

4.6. Quality Issues

- ➔ The game should keep and display all information correctly.
- ➔ For each user action that is outside the rules of the game, the game should not proceed the action and send a warning message to the user.
- ➔ After a failure, the game should be available to restart in less than 1 second.
- ➔ The game should be not available for at most 5 minutes in a 24 hour period.
- ➔ The game should be portable to Windows, Macintosh and Linux machines.

4.7. Resources and Management Issues

- ➔ The game should be backed up once in a month.
- ➔ The game itself should be responsible for back up.
- ➔ The game should be installed by users.
- ➔ After the submission of the game, the client should maintain the game.

5. SYSTEM MODELS

5.1. Use Case Models

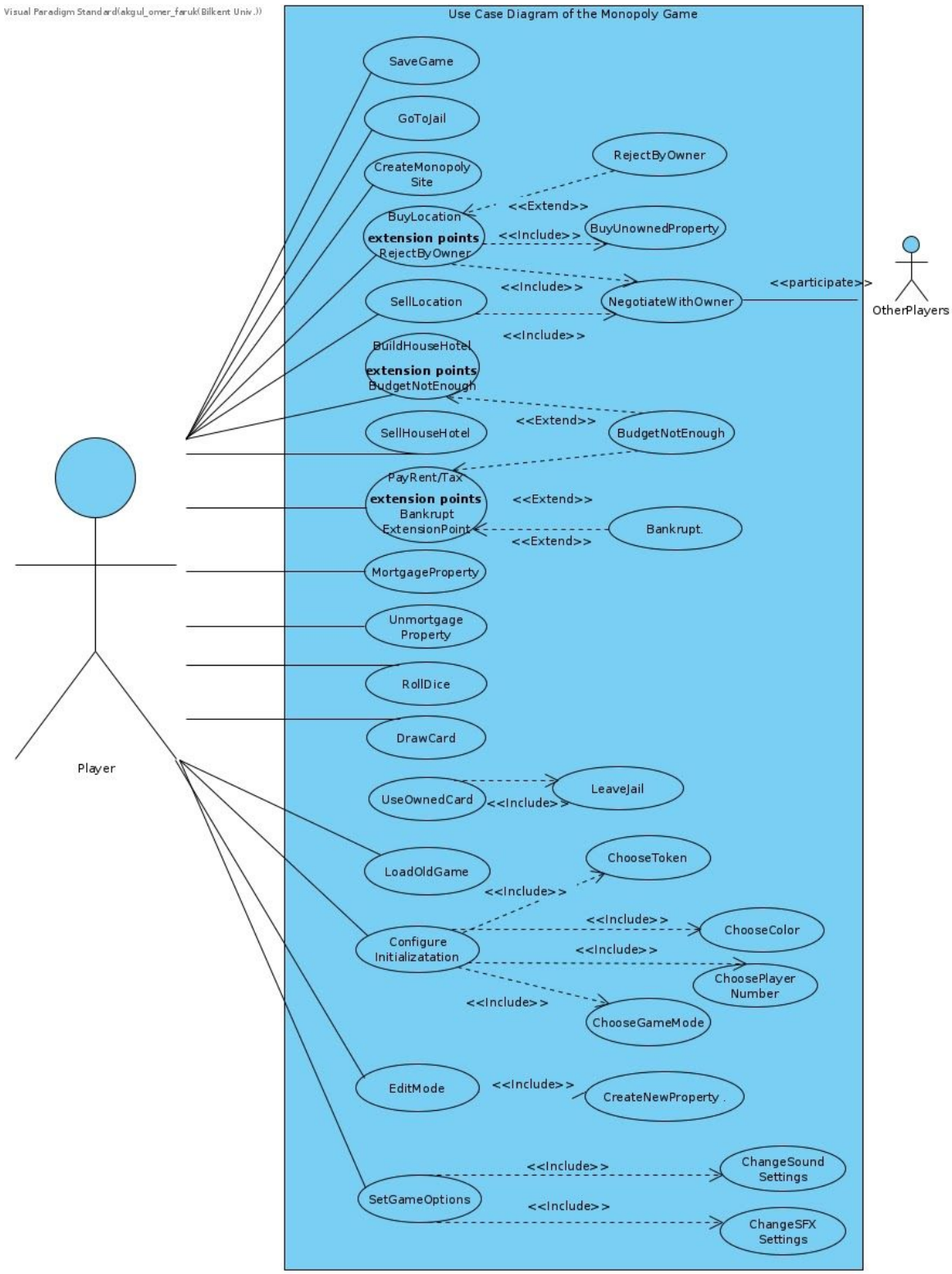

Figure 5.1: Use Case Diagram of Monopoly

5.1.1. Playing the Game

Name	PlayGame
Participating Actors	Initiated by Player Communicates with OtherPlayers
Flow of Events	1. A player presses the start button. 2. The system prompts a screen demanding users to roll dice. 3. Players roll the dice one by one and send the results to the system. 4. The system determines the order and initializes the game. Then, it displays the game screen based on the given criteria such as game mode, number of players and specified player colors. 5. Players roll the dice when their turn comes. 6. The system updates the location of the player based on the dice value. Then, the content of the location is displayed to the player. 7. Players perform operations such as buying, selling, paying, drawing and mortgaging based on their strategy and game restrictions. 8. The system checks the player, location and other necessary conditions. If the requested action obeys the game rules, the system performs required operations. 9. Player ends their turn. Other players repeat similar actions based on their game strategy.
Entry Condition	Player starts the game after choosing the game mode, number of players, and specifying colors and tokens.
Exit Conditions	1. All the players except one bankrupts so one player wins the game. 2. Player clicks on the exit button during the game. If a player saves before exiting, the existing game can be played later.
Quality Requirements	System should be robust to perform required functions without crashing.

5.1.2. Buying a Location

Name	BuyLocation
Participating Actors	İnitiated by Player Communicates with OtherPlayers
Flow of Events	1. Player rolls dice and comes to a location. 2. The system displays the content of location such as cost, rent, building prices and mortgage. 3. Player clicks on the buy button. 4. The system checks the balance of the player and withdraws the cost from his/her account, if the balance is enough. Also, the status of the property and site are updated.
Alternative Flow of Events	1. Player rolls dice and comes to a location. 2. The system displays the content of location such as cost, rent, building prices and mortgage. 3. Player clicks on the buy button but the location already has an owner, which is among OtherPlayers. 4. The system prompts a screen to ask the offer. 5. Player makes an offer and sends it. 6. The system prompts another screen to ask the owner whether he/she approves the offer. The cost of operation is withdrawn from the account of the player, if the owner accepts. Also, the status of the property and monopoly site are updated.
Entry Condition	Player comes to a location and clicks on buy.
Exit Conditions	1. Player buys the location and system performs the required actions. 2. The owner, a member of OtherPlayers, rejects the offer.

5.1.3. Selling a Location

Name	SellLocation
Participating Actors	İnitiated by Player Communicates with OtherPlayers
Flow of Events	1. Player rolls dice and comes to a location. 2. The system displays the content of location such as cost, rent, building prices and mortgage. 3. Player clicks on the sell button to sell a building on property that he owns to the bank for half of the building price. 4. The system updates the property and monopoly site status and deposits the money to the player's account.
Alternative Flow of Events	1. Player rolls dice and comes to a location. 2. The system displays the content of location such as cost, rent, building prices and mortgage. 3. Player clicks on the sell button but there is a difference this time. At least one of the other players has made an offer. 4. The system prompts a screen to ask the offer. 5. OtherPlayers makes their offers and sends them to the player who is currently playing. 6. The system prompts another screen to ask the owner whether he/she approves the offer. The cost of operation is transferred from the account of the tender winning player to the Player, if Player accepts. Also, the status of the property and monopoly site are updated.
Entry Condition	Player comes to a location and clicks on the button "sell".
Exit Conditions	Player sells the location and system performs the required actions.

5.1.4. Paying Rent or Tax

Name	PayRentTax	
Participating Actors	İnitiated by Player Communicates with OtherPlayers	
Flow of Events	1. Player rolls dice and comes to a location. 2. The system displays a message telling Player that the location has an owner and requests the rent. 3. Player clicks on the “pay rent” button. 4. The system transfers the money from Player to the owner, a member of OtherPlayers.	
Alternative Flow of Events	**1**	1. Player rolls dice and comes to a location. 2. The system displays the content of tax location. 3. Player clicks on the “pay tax” button. 4. The system transfers the money from the Player to the bank.
Alternative Flow of Events	**2**	1. Player rolls dice and comes to a location. 2. The system displays that the current budget is insufficient. Then, the system asks the user to perform actions to collect the expected money.
Entry Condition	Player comes to a location demanding rent or tax.	
Exit Conditions	1. Player pays the tax or rent. 2. Player performs other actions as the balance of Player falls below zero. Then Player pays the money. 3. Player goes bankrupt.	

5.1.5. Building a House or Hotel on a Location

Name	BuildHouseHotel
Participating Actors	Player
Flow of Events	1. Player rolls dice and comes to his/her own location. 2. The system displays the content of location such as cost, rent, building prices and mortgage. 3. Player clicks on the “build” button. 4. The system displays the build screen. 5. Player pays the price of the house if there is an available house. If all the houses are already bought, hotel prices can be paid. 6. The system updates the property status and rent.
Alternative Flow of Events	1. Player rolls dice and comes to his/her own location. 2. The system displays the content of location such as cost, rent, building prices and mortgage. 3. Player clicks on the “build” button. 4. The system displays the build screen. 5. Player makes a request to build a house/hotel. 6. The system warns player as the budget falls below zero when the demanded operation is performed.
Entry Condition	Player rolls dice and comes to his/her own location.
Exit Conditions	1. Player builds a house/hotel and system performs the required actions. 2. Players cannot build a house/hotel and the player’s turn ends.

5.1.6. Creating a Monopoly

Name	Create Monopoly Site
Participating Actors	Player
Flow of Events	1. Player owns all properties of the same color. 2. The system creates a monopoly belonging to Player, which means all the rents of the monopoly sites with the same color are doubled.
Entry Condition	Player buys all the properties with the same color.
Exit Conditions	The system automatically exits.

5.1.7. Mortgaging Property

Name	Mortgage Property
Participating Actors	Player
Flow of Events	1. Player clicks on “mortgage” button as his/her budget is about to zero. 2. The system sells all buildings on the property to the bank for half of their original prices. The system deposits the price to the Player's account.
Entry Condition	Player clicks on the “mortgage” button.
Exit Conditions	1. Player turns end after mortgaging.

5.1.8. Drawing a Card

Name	DrawCard
Participating Actors	Player
Flow of Events	1. Player comes to either the chance or community chest location. 2. The system displays a screen for guiding Player to draw a card. 3. Player clicks on the “draw card” button. 4. The system shuffles and draws one of the cards. The system displays the card and applies the written action.
Entry Condition	Player comes to either the chance or community chest location.
Exit Conditions	1. The system automatically exits after completing written operations.

5.1.9. Using an Owned Card

Name	UseOwnedCard
Participating Actors	Player
Flow of Events	1. Player clicks on the “use owned card” button. 2. The system performs the action written on the card and discards the card from Player’s cards.
Entry Condition	Player is punished with “GoToJail” and he/she wants to evade punishment. Player clicks on the “use owned card” button.
Exit Conditions	The system automatically exits after using the card.

5.1.10. Configuring İnitialization

Name	ConfigureInitialization
Participating Actors	Player
Flow of Events	1. Player requests the creation of a Monopoly game. 2. The system prompts a screen to ask the number of players, their names, token types and colors. 3. Players fill required positions. Then, the player chooses one of the modes: Blitz, Classic, and Reverse. 4. Player finishes the initialization process.
Entry Condition	Player opens the Monopoly game and enters the configuration on the screen.
Exit Conditions	1. Player completes the process after filling all the required areas. 2. Player returns back to the main menu without completing.

5.1.11. Loading a Saved Game

Name	LoadOldGame
Participating Actors	Player
Flow of Events	1. Player requests to continue a previously saved game. 2. The system finds the storage file of the previous game and initializes the game. 3. Player continues playing.
Entry Condition	Player clicks “load game” option and chooses one of the saved games.
Exit Conditions	Player returns back to the main menu without opening a game.
Quality Requirements	System should be able read the file and load the game in a second.

5.1.12. Editing Mode

Name	EditMode
Participating Actors	Player
Flow of Events	1. Player makes a request to change the map of the Monopoly game. 2. The system prompts a screen, which displays an editable map. 3. Player clicks on the blocks of the map one by one. 4. On each click on a block, the system displays a screen, where the player is asked to enter name, cost, rent, hotel price, house price and mortgage. 5. Player fills all the blank areas and saves the block content. After completing all the blocks, the player saves the new mode. 6. The system adds the new mode to the mode list.
Entry Condition	Player selects the edit mode option in the main menu.
Exit Conditions	1. Player completes and saves the process after filling all the required areas. 2. Player returns back to the main menu without completing.

5.1.13. Setting Game Options

Name	SetGameOptions
Participating Actors	Player
Flow of Events	1. Player enters the options menu. Then, he/she changes the values of sound and sfx. 2. The system applies the changes without requiring any saving operation.
Entry Condition	Player selects the options in the main menu.
Exit Conditions	1. Player returns to the main menu after applying changes.

5.2. Dynamic Models

5.2.1. Turn Activity Diagram

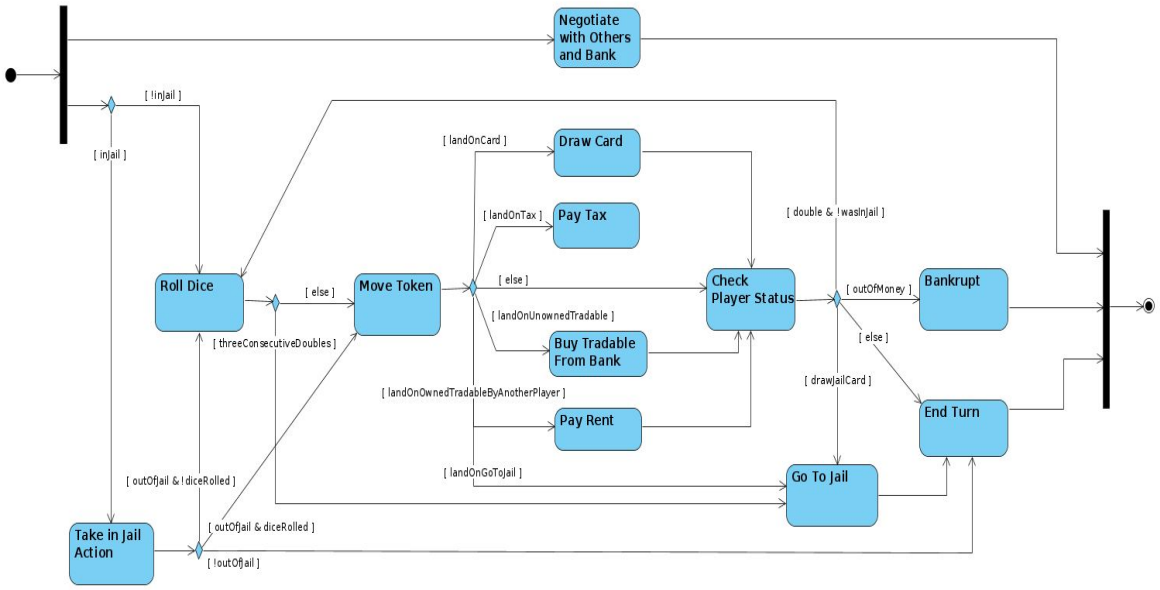

Figure 5.2.1: Activity Diagram for Player Turn

This is the activity diagram about a single turn of a player. The diagram starts when the player's turn starts, the diagram ends when the player's turn ends. To start with, we have "Negotiate with Others and Bank" activity, which includes trading properties and railroads - which we call as "tradables" - with other players, mortgaging tradables, building-selling houses to properties, etc... In short, this activity contains the use cases which can be done at any stage in a player's turn. That's why it is parallel to the rest of the diagram.

At the start of his turn, if the player is in jail, he might take some "In Jail Actions" which will be discussed in a later diagram. As a result of his actions, if he doesn't get out of jail, he simply needs to end his turn. If he gets out of jail, he will continue his turn as a normal turn. If he rolls dice in jail and gets out, he doesn't need to roll the dice again, he can simply move his token according to the result of rolled dice. If he gets out of jail without rolling dice, he needs to roll dice to continue.

If the player is not in jail, he simply rolls the dice and moves his token with the result on the dice. Then he lands on a location - which is the name of a square in the Monopoly board. According to where he lands, the player might do different activities. If he lands on a card location, Chance or Community Chest, he must draw a card and act accordingly. Landing on Tax makes the player pay tax. As it is mentioned earlier, tradables are simply properties and railroads. If a player lands on an unowned tradable, he might buy a tradable from the bank. If he lands on a tradable which is owned by another player, he must pay rent. If the player lands on the Go To Jail location, he must go to jail directly. If he lands on any other location, such as Go or Free Parking, he doesn't take any specific action. Note that the money coming from passing Go is inside the Move Token activity.

After the player carries out his location specific activity, there is a general player status check. If the player is out of money and any kind of negotiation cannot save him, he goes bankrupt. If the player has drawn a jail card during the Draw Card activity, he directly goes to jail. If a player has rolled a double and was not in the jail at the start of his turn, he rolls his dice and carries out the previous steps again. Note that he might get out of jail by rolling double, in this case the player doesn't roll the dice again. Also note that if a player rolls three consecutive doubles in a single turn, he directly goes to jail.

If the player goes to jail because of any actions mentioned above, he must simply end his turn. Note that turn has two different ending scenarios; player bankrupts or player ends his turn willingly.

5.2.2. Jail State Diagram

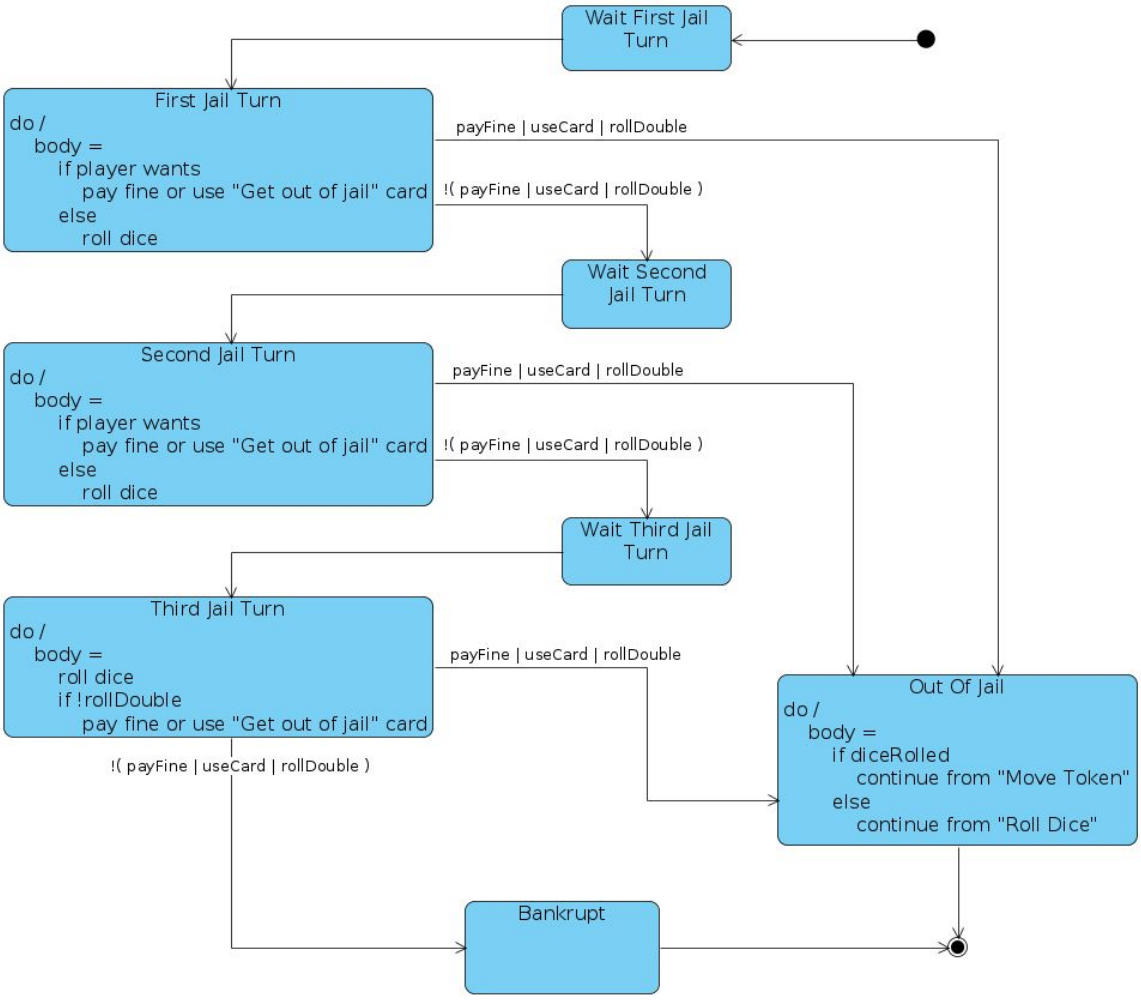

Figure 5.2.2: State Diagram for Being Jailed

The state diagram given above describes the states of a player in jail. This diagram is also connected to the Turn Activity Diagram. In fact, this state diagram mostly explains the "Take In Jail Action" activity in that diagram. So, in this state diagram, some activities from the activity diagram are mentioned.

First of all, when a player goes into jail, he needs to wait until his next turn, which is mentioned in the diagram as the "First Jail Turn". In the first jail turn, the player can pay a fine or use the "Get out of Jail Free" card to get out of jail. If he doesn't want to do any of them, he will roll the dice. If the result of the dice is double, the player again gets out of the jail, else, he stays

in the jail and waits for his next turn. Note that the player can't pay a fine or use a card after rolling the dice. Second Jail Turn is similar to the first jail turn.

Third Jail Turn is somewhat different. In this turn, the player must roll the dice directly. If the dice is double, the player gets out of jail. Else, he must pay a fine or use the "Get out of Jail Free" card. If he can't even do that, he goes bankrupt. Note that in the Turn Activity Diagram, we didn't mention that a player can go bankrupt in jail, in order to keep it as simple as possible. Instead, we are giving this detail here.

As mentioned in the activity diagram, after getting out of jail, if the player has rolled the dice in the current turn, he continues his turn from the "Move Token" activity with the result of his rolled dice; else he must roll dice and continue. Note that if the player gets out of jail in the first or second turn, "diceRolled" is equivalent to "rollDouble". However in the third turn, even if the player doesn't roll a double, he can pay a fine or use a card to get out of jail. Then he moves his token according to the dice he rolled in the jail.

5.2.3. Property State Diagram

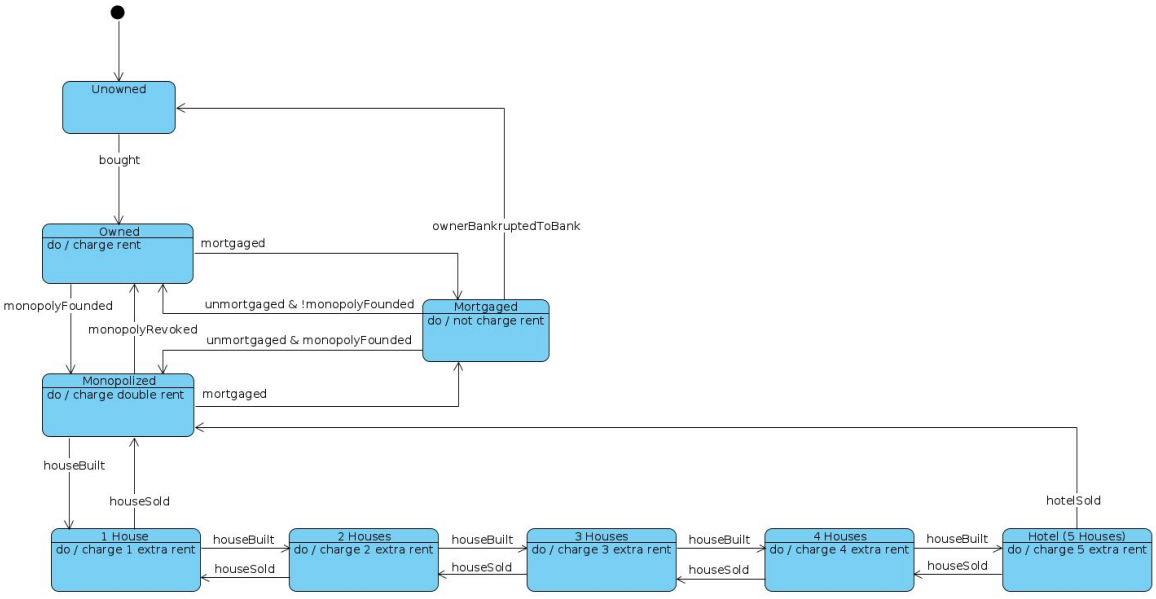

Figure 5.2.3: State Diagram for the Status of a Property

The state diagram given above demonstrates the states that a property can be in during the gameplay. At the beginning of the game, each property is "Unowned". When a player buys it from the bank, it becomes "Owned" and its owner charges rent from other players stepping on the property. If the owner acquires all properties having the same color, this means that the owner has founded a monopoly and all properties with this color are upgraded to "Monopolized", and the owner charges double rent from these properties. If the owner mortgages a property, this property becomes "Mortgaged" and the owner cannot charge rent from the property.

The transitions between "Owned" and "Monopolized" can occur in many ways. For example, when the owner of a monopoly sells a property to another player, all of the properties in the monopoly will be downgraded to "Owned" state. Note that the sold property is now owned by another player. If the owner of a monopoly mortgages one of the properties, the other properties in this monopoly will again be downgraded to "Owned" state. When a mortgaged property is unmortgaged, it can be either owned or monopolized according to its current state. As you can see, there are many details about the negotiations between players and the bank. However, this is not our concern in this diagram. Therefore, we simply denoted all of those possibilities with "monopolyFounded" and "monopolyRevoked" boolean values.

Since there are no auctions, when a player goes bankrupt to the bank, all of his properties become unowned. Note that before going bankrupt, the player mortgages every property to pay his debts, so his properties must be in the state "Mortgaged" before being "Unowned".

About the houses and hotels, they can be built only when the property is monopolized. Original game rules ensure that the property can have buildings on it only if it is monopolized. So we can consider extra houses as upgrades to the "Monopolized" state. According to the number of houses on a property, the owner charges extra rent. Houses can be bought and sold one by one. Note that a hotel can be sold at once.

5.2.4. Railroad State Diagram

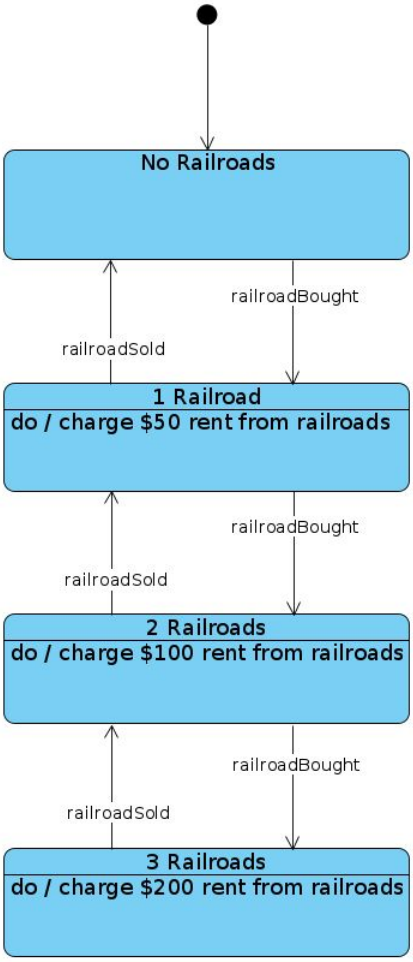

Figure 5.2.4: Railroad State Diagram

The state diagram given above is demonstrating the effect of railroad count of a player to the railroad rents. The states keep the number of railroads a player has. At the beginning of the game, all players start with zero railroads. Players can buy or sell their railroads one by one. As you can see from the diagram, the more railroads a player has, the higher amount of charged rents.

5.2.5. Game Direction in Reverse Mode State Diagram

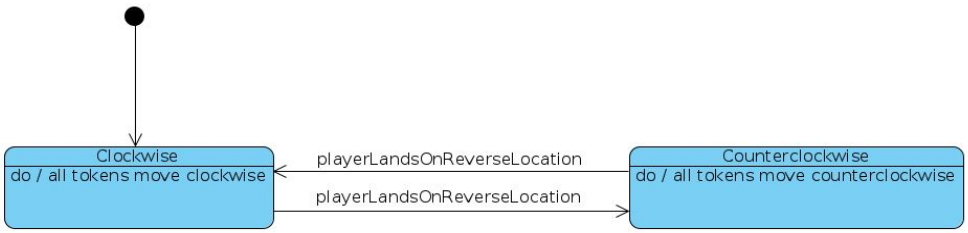

Figure 5.2.5: Reverse Mode State Diagram

The state diagram given above demonstrates the game direction in a reverse game. At the beginning of the game, the game's direction is clockwise, which means that all tokens move in the clockwise direction on the board. Whenever a player lands on a Reverse location, the game direction changes; meaning, all tokens start moving in the opposite direction.

5.2.6. Purchasing Item

The diagram below displays the high level interactions between classes, when a player comes to the property which is not purchased yet. The diagram below displays the high level interactions so it does not use the function names and parameters. The aim here is to provide the reader with the general information flow between classes. However, the following diagrams will be using function names, as well. This way, we think the reader is going to feel more comfortable while looking at the following diagrams.

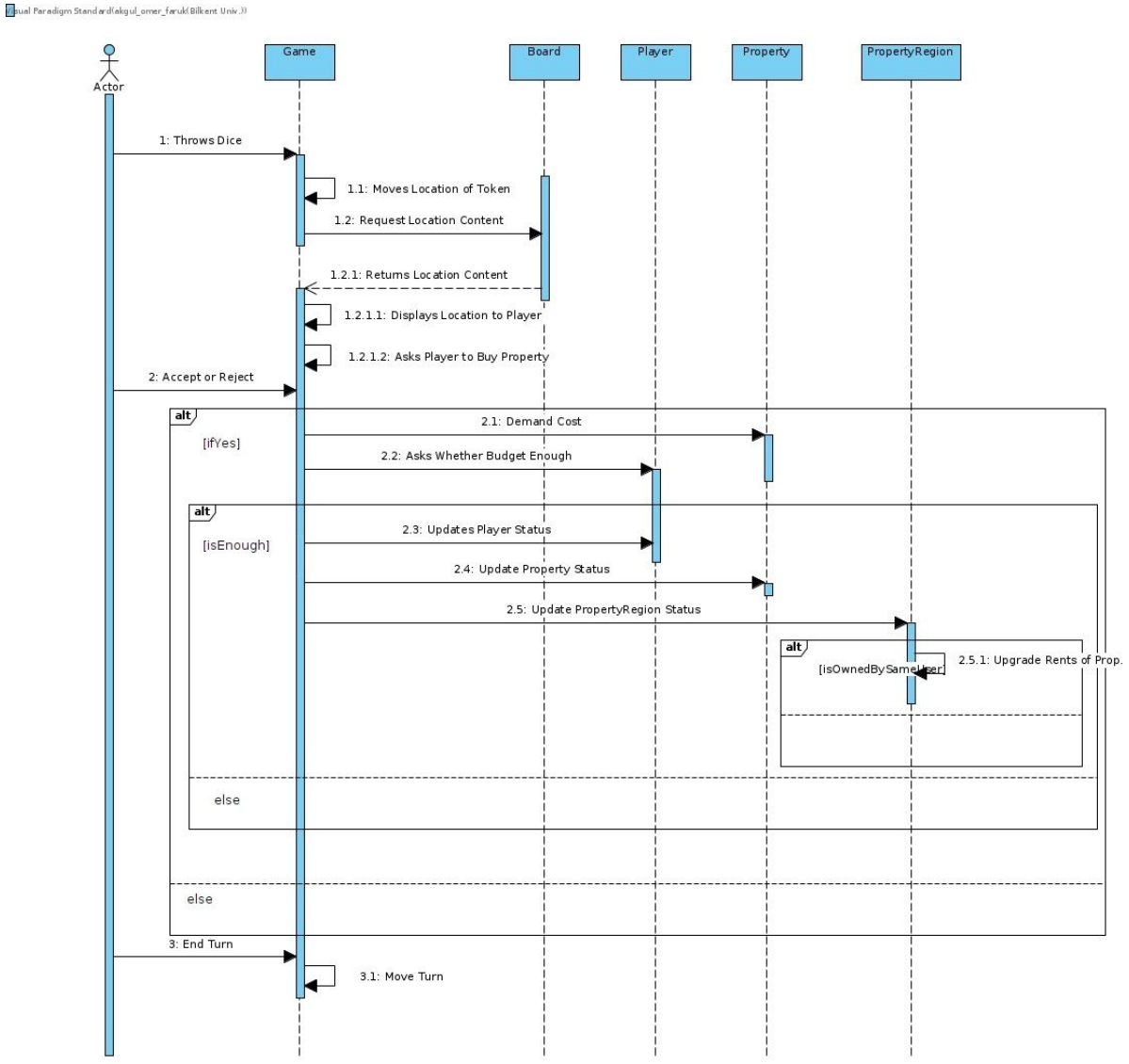

Figure 5.2.6: Sequence Diagram of Purchasing Item

The Game class acts as an engine of the Monopoly game so most of the interactions are triggered by the Game class. All the actions in the diagram above starts when the player rolls a dice. The Game class moves the token of the player forward based on the sum of dice. Then, a query is sent to the Board class to learn the content of the place. The content of the location including name, cost and rent information is displayed. If the player decides to purchase, Game withdraws the price of money from the player's account and updates the assets of the player. Also, the status of property is updated such that the owner information can be learned during the upcoming steps. However, if the player's account is less than the price, a warning message is displayed on the screen. Finally, the actor ends his/her turn, which triggers Game to move the turn.

5.2.7. Paying Rent and Tender Processes

The diagram below is designed to show how objects interact with each other, when the player comes to the place owned by another player. In addition, the bankruptcy process for the player whose money is going to be withdrawn is explained. In the diagram below, starting from the paying rent step might be more focused. However, as the functional interaction while the dice is being rolled is not displayed in the diagram above, we wanted to add these steps to the diagram below, as well.

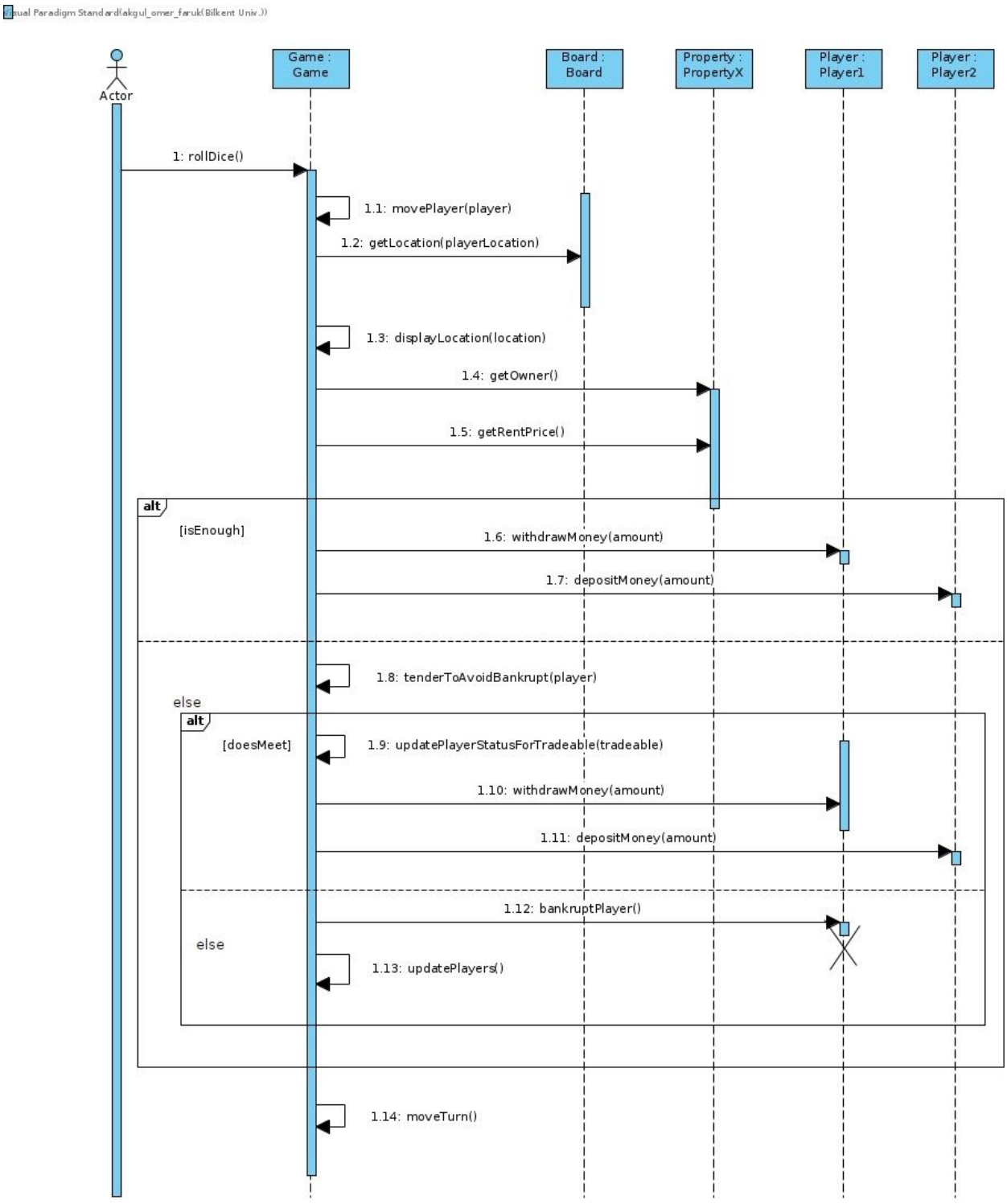

Figure 5.2.7: Sequence Diagram to Display Rent and Tender Processes

As in the previous scenario, the interaction starts when the actor rolls the dice. However, this time, the location is owned by another user. Thus, the current player has to pay a rent to the owner. After learning the owner and rent amount, Game withdraws the rent from the current player and deposits to the account of the owner. Nevertheless, the current balance of the player might be less than the rent. In such a scenario, the tender process should be initialized by the player to avoid bankruptcy. If the player earns enough money to pay the rent, the game will move on, but if the player still cannot pay, then the player is bankrupt.

5.2.8. Drawing Chance / Community Chest Cards

The following diagram aims to visualize the interaction of objects, when the player comes to the chance block. It could be integrated with the diagram above but, instead, drawing a separate diagram is preferred to increase the understandability of diagrams. The diagram is approximately the same for community chest block except the payment address.

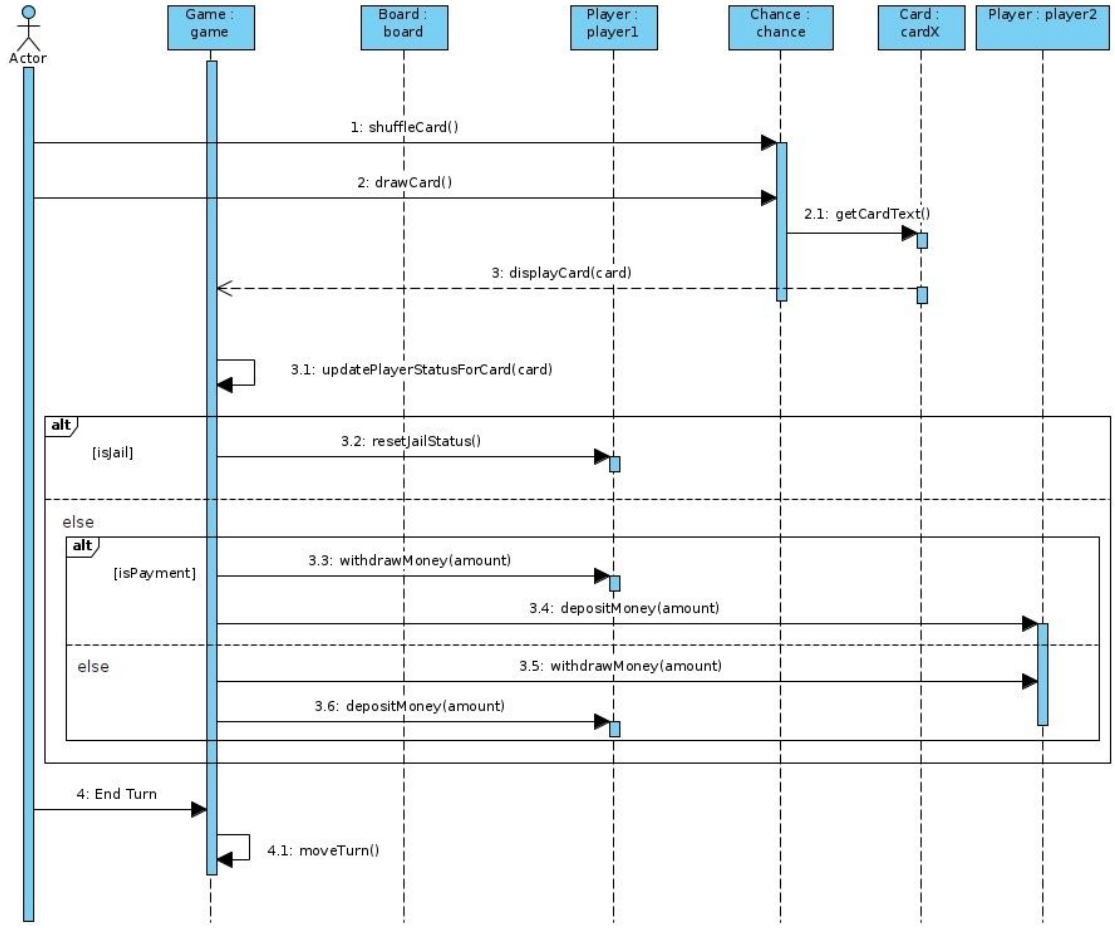

Figure 5.2.8: Sequence Diagram for Card Operations

When the player comes to the Chance location, the following process is performed. The process is valid for the Community Chest, as well. The player shuffles the cards and draws one of them. Chosen card object is displayed on the screen. Then, the game object updated the status of the player according to the card drawn. For instance, if the card says “Got to Jail for 3 Rounds”, the jail status of the player is updated accordingly. If the card says “Take money from all the other players”, the game object withdraws money from other users and deposits it to the current user.

5.2.9. Building House

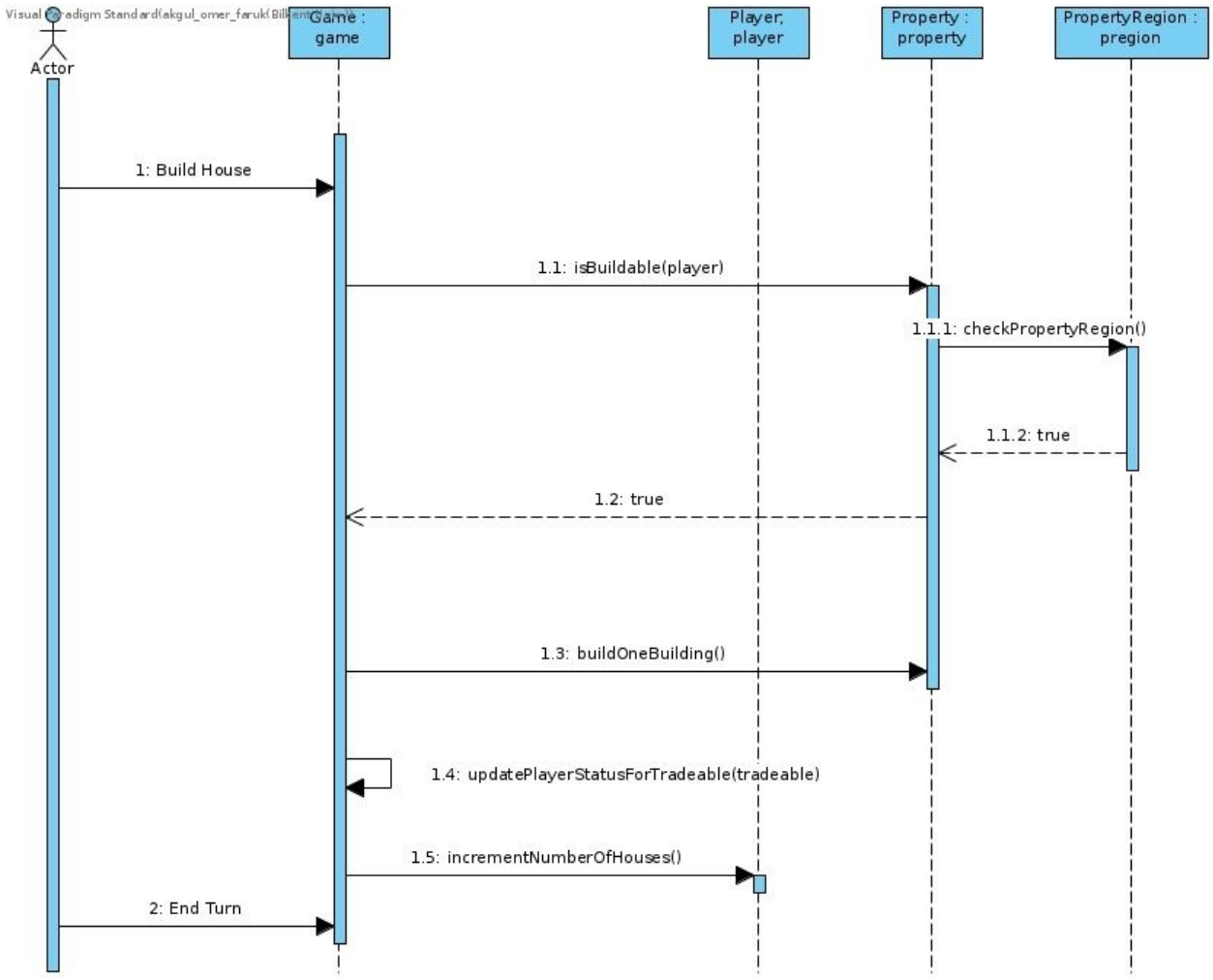

Figure 5.2.9: Sequence Diagram for House Building Operation

When the player wants to build a house on his/her own property, the game manager checks whether the property is appropriate to be builded. If the answer is yes, the house is builded and required updates are performed by the game manager.

5.2.10. Reverse Game Mode

When one of the players comes to the reverse block of the game, game direction changes as shown in the diagram below.

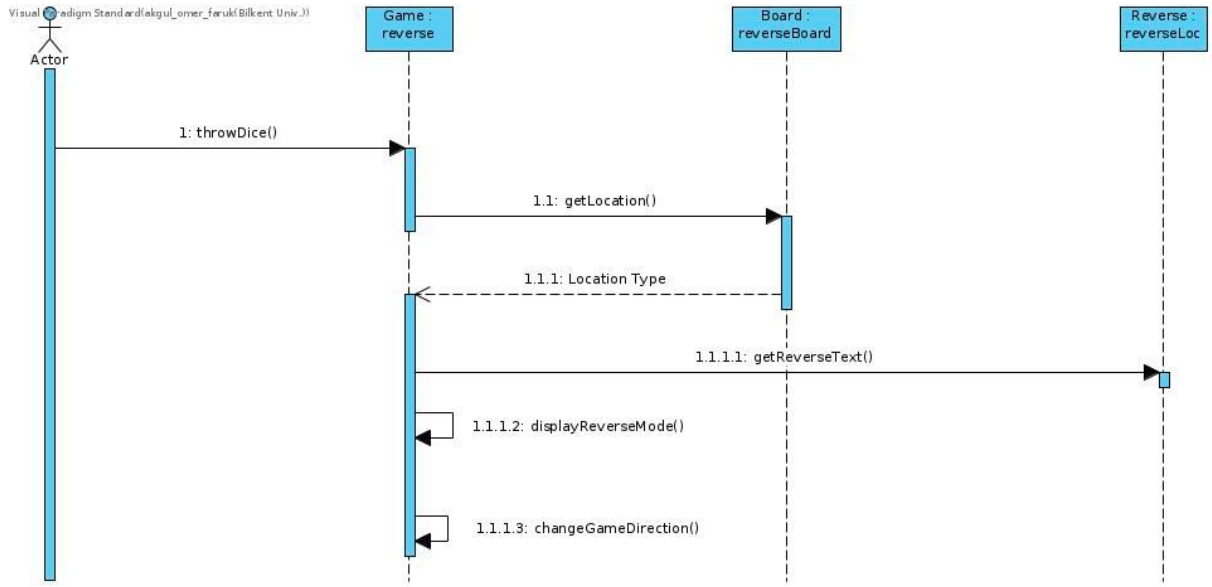

Figure 5.2.10: Sequence Diagram of Reverse Game Mode

When the player comes to the reverse block and the direction is changed, all the players start to move back until one of the players comes to the reverse block again.

5.2.11. Blitz Game Mode

The following state diagram represents Blitz mode. Conceptually, Blitz mode is inspired from the Blitz Chess. In this diagram, the main purpose is to display the differences of Blitz mode from Classic mode. The activities that a player can do in a turn is the same as a Classic game. Therefore, a turn of a player is abstracted in the diagram.

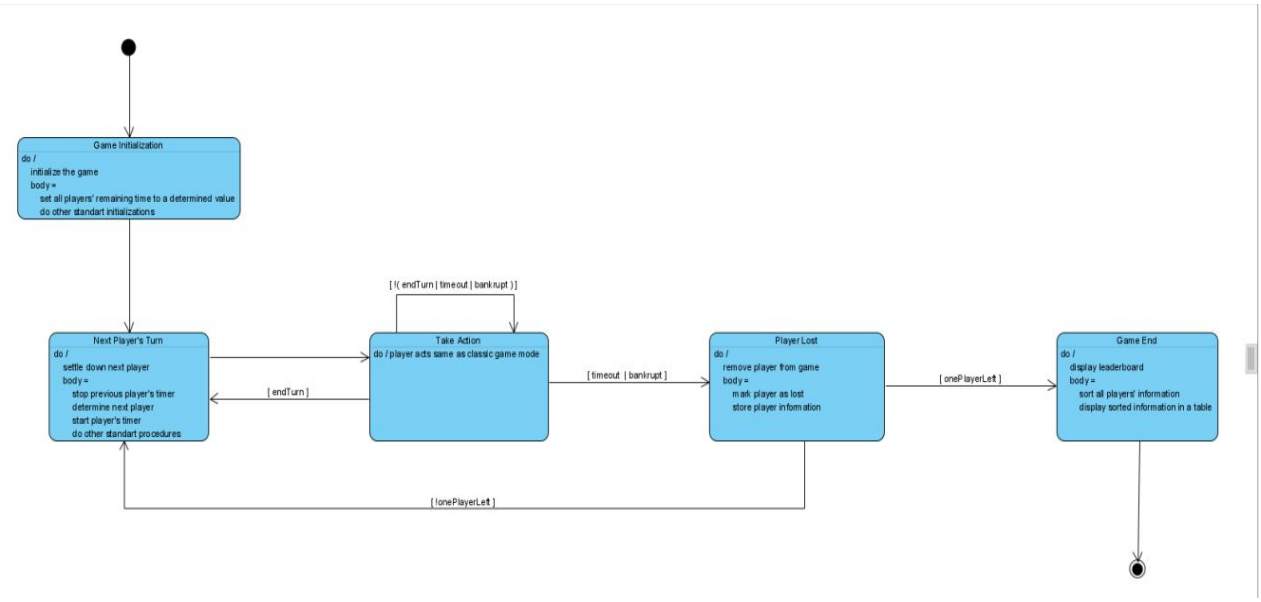

Figure 5.2.11: Blitz Mode State Diagram

Initialization of the Blitz mode is nearly the same as Classic mode. Additionally, each player is given some amount of total time. When the turn comes to a player, this player's remaining time starts decreasing. During his/her turn, the player can take action just like the Classic game (i.e. rolls dice, moves the token and acts accordingly). After the current player decides to end the turn, the current player's timer stops and the next player's turn and timer starts. Players lose the game if they go bankrupt or their time ends. When a player loses, his/her current wealth (determined from money and owned locations) is saved. The game continues until one player remains unlost. When there is only one player left in the game, there is no point in playing. So, the game ends and a leaderboard prepared from players' wealth is displayed. Note that a wealthy player whose time is up at an early stage of the game can still win the game.

5.3. Object and Class Model

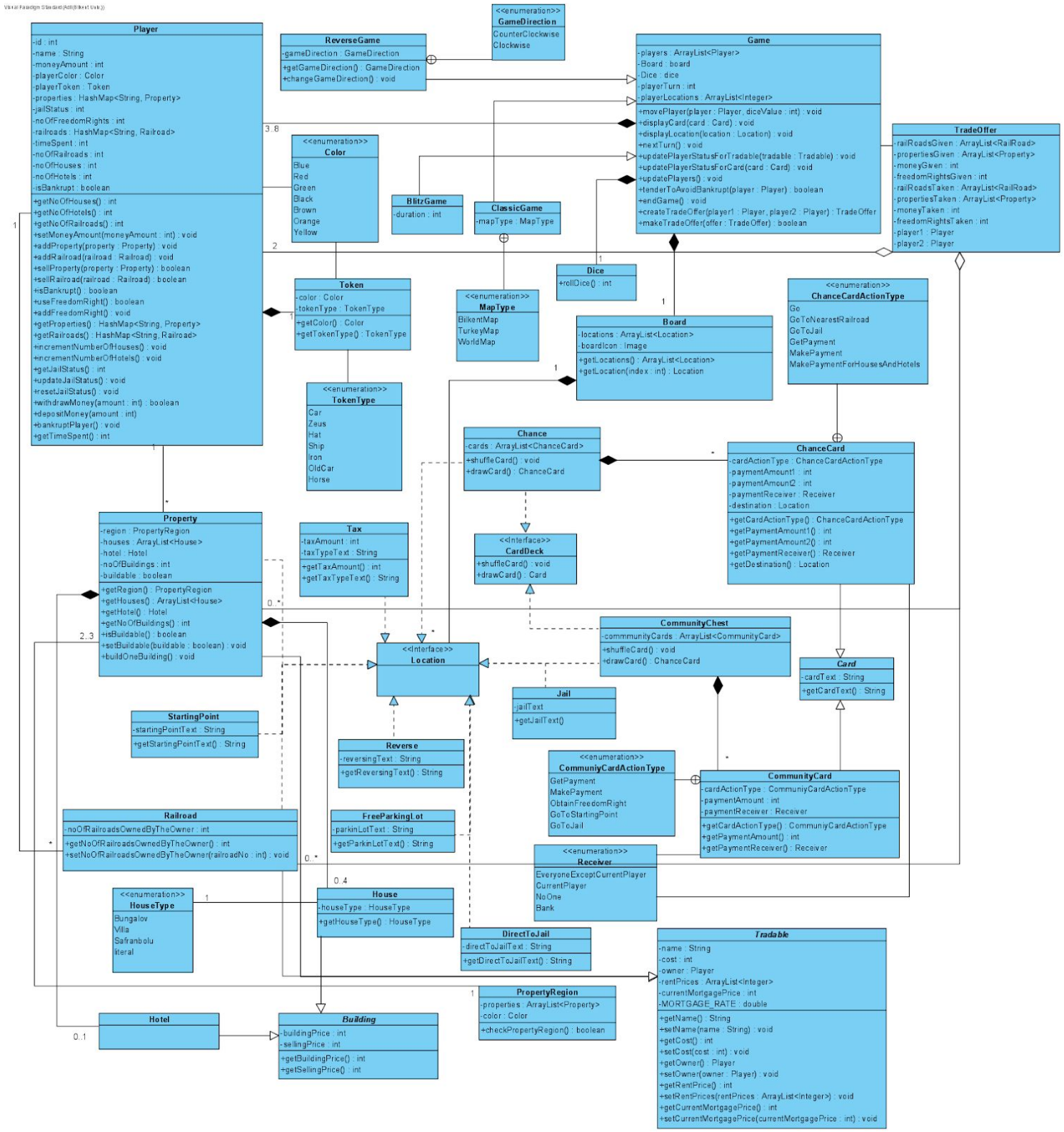

Figure 5.3: Object Class Diagram

Class Descriptions

The class diagram of the “Monopoly” game is stated at the Figure above.

Classes

The main class of the Monopoly game is the Game class. Game class is responsible for the functionalities of the game. It contains players, dice, board and player’s locations. The game class has 3 subclasses which represents the different Game modes.

ClassicGame

ClassGame class is the first subclass of the Game class which is responsible for the classic game mode. It contains 3 existing constant maps which are BilkentMap, TurkeyMap and WorldMap.

BlitzGame

BlitzGame class is the second subclass of the Game class which is responsible for the blitz mode. It has duration property different from its parent class, it specifies the total duration for the game.

ReverseGame

ReverseGame class is the third subclass of the Game class which is responsible for the reverse mode. It has a gameDirection property different from its parent class. Also, it contains inner enumeration which specifies the game direction of the class. The GameDirection enumeration can only take two values that specify the game direction which are CounterClockWise and ClockWise.

Player

As can be deducted from its name Player class is responsible for players and their current status. All players, therefore, player instances have different colors and tokens. This class contains all game instances owned by the player and related with the player. Player’s money, properties, railroads, freedom rights are kept by this class.

Dice

Dice class is defined for the dice. In order to move, players should roll the dice and they may move through the board with respect to the facing values of the dice. The rollDice method returns an integer number between 2 and 12 inclusively.

Board

Board class is responsible for the game board which means the construction of the game map and orders of the locations. It contains a location array. The first (0) index of the location array represents the starting point and last index represents the ending point of the board(map).

Token

This class is used to visually represent current players. The tokens are colored with respect to the player's color. And players may choose one of the provided tokens by the game.

Location

Location is an interface used to group all locations. It has 9 sub-classes which are the different kinds of locations on the board and the game. The location abstract class does not contain any properties or methods. Board class uses arrays that consist of location classes.

Property

Property class is a subclass of the location class. It is responsible for the properties that exist in the game. It contains all information about the properties such as its name, price, renting price and current owner.

PropertyRegion

PropertyRegion class is responsible for the property regions. It keeps the properties with the same region. It is used to check the availability of building houses/hotels and updating the rent prices when all properties that belong to the same region are owned by the same player.

Railroad

Railroad class is a subclass of the location class, hence, it is a location. It is responsible for the railroads that exist in the game. It contains all information about the railroads such as its price, rent price. Since the rent price of the railroads change according to the number of railroads owned by its owner, it keeps the number of railroads owned by its owner.

Tax

Tax class is a location. It is responsible for the tax locations that exist in the game. There are several tax locations with different back stories and amounts. Therefore, it keeps the text which represents the back story, and the amount the player should pay.

Jail

Jail class is a location. It is responsible for the jail areas in the game. At default maps, there is only one jail location where players go when they are imprisoned.

StartingPoint

StartingPoint class is a location. It is the first location and starting location in the game. When users pass or stop at the starting point, they receive money.

Reverse

Reverse class is a location that exists only in the reverse mode. When a player comes to the reverse location, the direction of the game changes.

Chance

Chance class is a location and card deck that exists in the classic mode and the blitz mode. This class implements the CardDeck interface. When a player comes to the chance location, the player draws a chance card from the deck and the related actions are made.

CommunityChest

CommunityChest class is a location and card deck that exists in the classic mode and the blitz mode. This class implements the CardDeck interface. When a player comes to the community chest location, the player draws a community card from the deck and the related actions are made.

ChanceCard

ChanceCard class is a card because it implements Card interface. It contains specific information about an action which will be made. It keeps the payment amount, destination, payment receiver and card action type which is an inner enumeration that is defined in the ChanceCard class. According to the card action type, the game decides the type of action and makes the action according to the properties which are payment amount, destination and payment receiver.

CommunityCard

CommunityCard class is a card because it implements the Card interface. It contains specific information about an action which will be made. It keeps the payment amount, payment receiver and card action type which is an inner enumeration that is defined in the CommunityCard class. According to the card action type, the game decides the type of action and makes the action according to the properties which are payment amount and payment receiver.

FreeParkingLot

FreeParkingLot class is a location which is responsible for the free parking lot area. When a player comes to this point, the player takes all of the money in the middle.

5.4. User Interface

5.4.1. Main Menu

When the user gets into the game, they will see the main menu first. From this screen, the user can select one of the following options they are presented with.

1. **Play**

   This will take the user to the further options of the game mode, and the player count.

2. **Mode Editor**

   This button will take the user to the mode editor.

3. **Options**

   When users click onto the **Options** button, they will go to the **Options** menu.

4. **How to Play**

   This button will open up the **How to Play** instructions.

5. **Credits**

   This button will take users to the **Credits** screen.

6. **Quit**

   The user closes the software and **quits** the game.

Figure 5.4.1: Main Menu Screen

5.4.2. Play Game

When the user clicks the **Play Game** button in the **Main Menu**, this screen will come up. In this screen, the user can specify **the player count**, and **the game mode** from the drop down menu right below the name. There will be default placeholders for **the player count**, and **the game mode** fields in their own respective drop down menus. The user may choose the **Back** button to navigate to the **Main Menu,** and choose the **New Game** button to specify the game properties further in the **Player Specification** screen. The **Load Game** button will take the user to the **Load Game** screen.

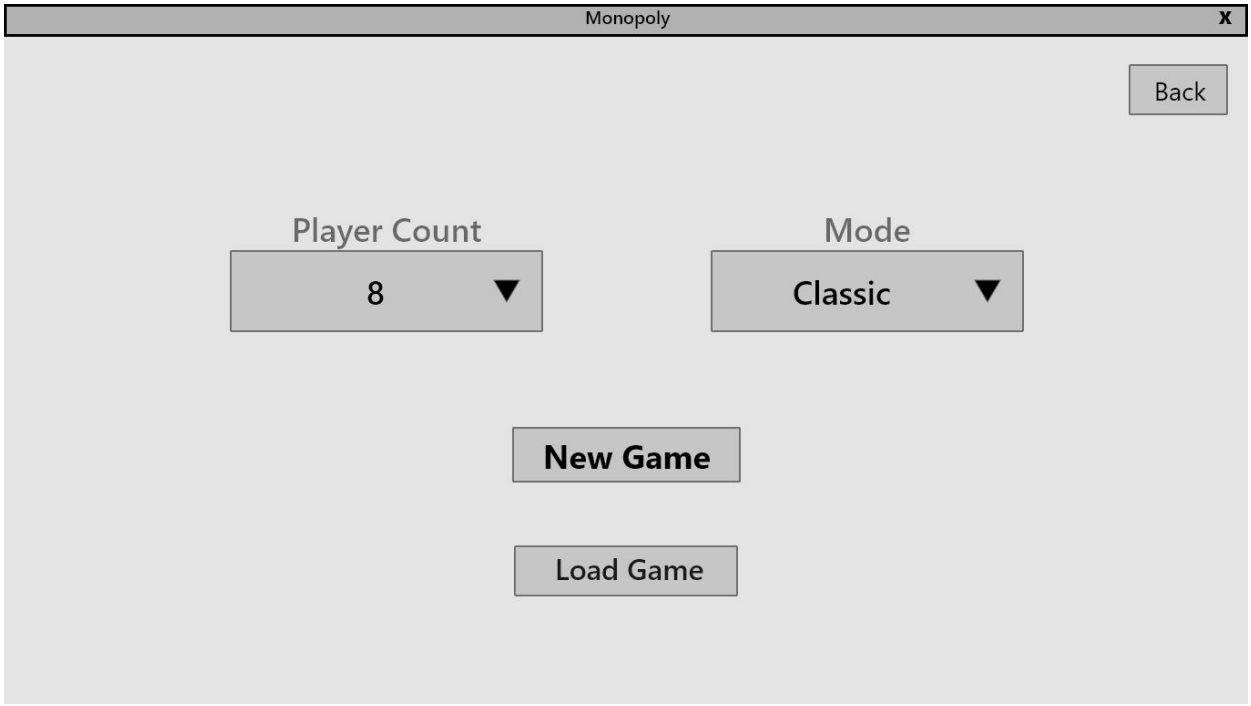

Figure 5.4.2: Play Game Screen

5.4.3. Load Game

In this screen, the user can select a previously saved game to continue playing. The saved games have 4 main properties. Name, mode, save time, and time elapsed. The user can delete the saved game by clicking on the **X** button or can continue to play the game by clicking at the green horizontal triangle next to the **X** button. The user may choose to navigate to the **Play Game** screen by clicking the **Back** button.

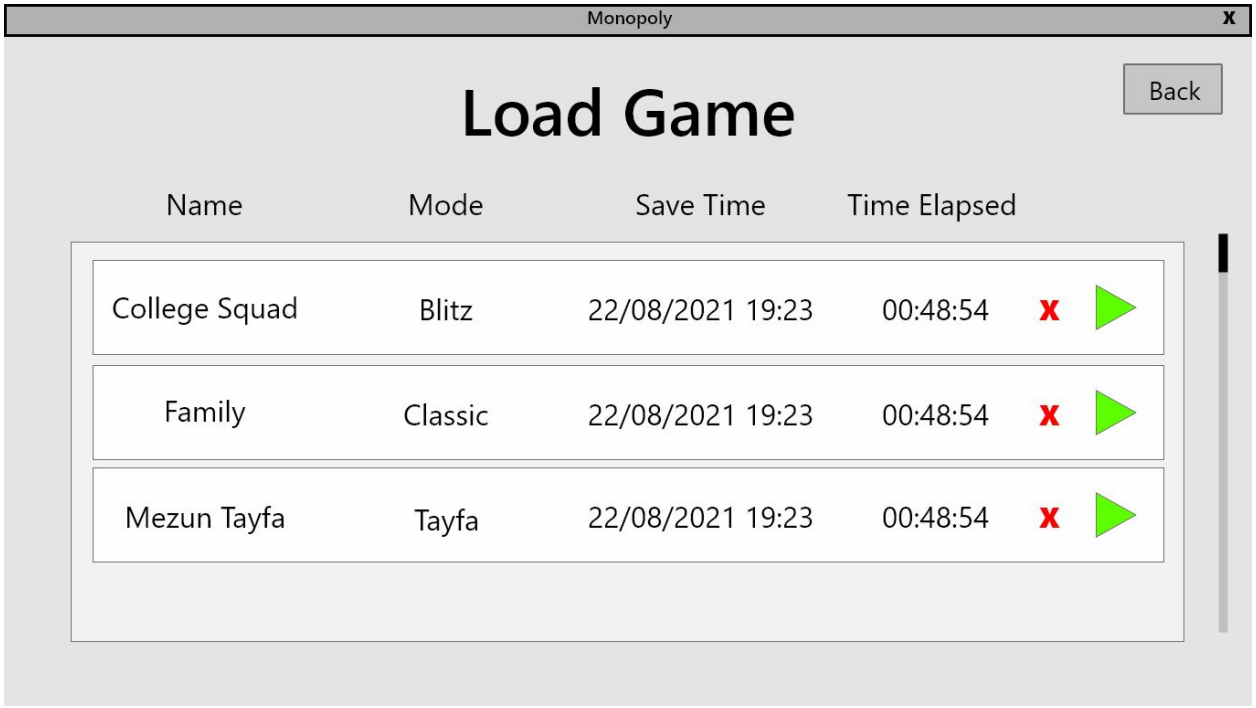

Figure 5.4.3: Load Game Screen

5.4.4. Player Specialization

In this screen, the user can specify the names of each player, choose their tokens, and confirm their changes. In this screen, it can be seen that all of the players are not ready, they have not confirmed their names or tokens. In this case, the user cannot **start** the game. When all of the players are ready, the user can start the game. The **start** button navigates to the **in-game** screen. The **back** button navigates to the **Play Game** screen.

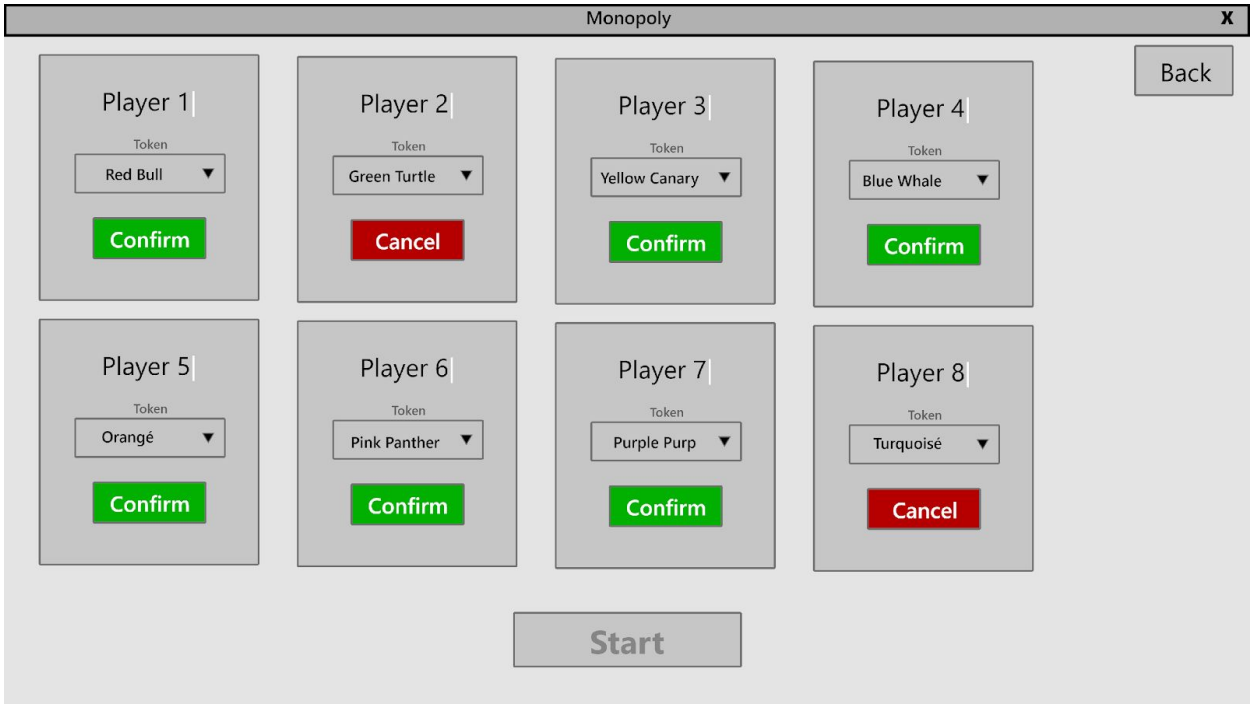

Figure 5.4.4: Player Specialization Screen

5.4.5. In-Game Screen

In this screen, the players can see the table, their cash value, who owns the cities, which cities are in the same group and so forth. However, **one must not forget that** this is only a **mock-up**, therefore, these visualizations are not exactly the implementation wire frames, and it is simplified to give the basic understanding of the implementation.

Each player gets a unique color with the tokens they have chosen and it forms the color of their name and cash value. There appears a giant sideways black triangle next to the player whose turn is on. On the right of the screen, there is the pause button, which opens a dialog to close the game, and perhaps some other features we might implement. The center colors of the properties on the table show which property group they belong to. The upper blocks on each of the properties show the owner of the property and have the same color as the owner's token color. Tokens are shown as little disks at the left lower corners of properties.

As we are implementing this game as a multiplayer game on a single local computer, we cannot control who is taking actions, we only assume. The user can roll the two dice and then

continue to take actions such as buying houses. The user must click **end of turn** to end his turn and pass the turn to the next player. The user can make further arrangements as long as he/she does not roll the two dice. The parallelogram on the upper side of the roll button is the **chance** card holder and on the lower side of the roll button is the **community chest** card holder. The places on the board's side are the **chance** and the **community chest** places respective to their symbols. The **tax** place is the place where newcomers pay tax.

In this mock-up, there are only 8 places on each side of the board, however, there will be 10 when we implement the full game. We will further discuss this in the design reports. When the user clicks on a property placed on the board, he/she can access the information about the property, and can take access if he/she has the proper authorization. This will be shown in the **In-Game – Property Screen**. By clicking at the pause button at the center right, the user will navigate to the **In-Game Pause Screen.**

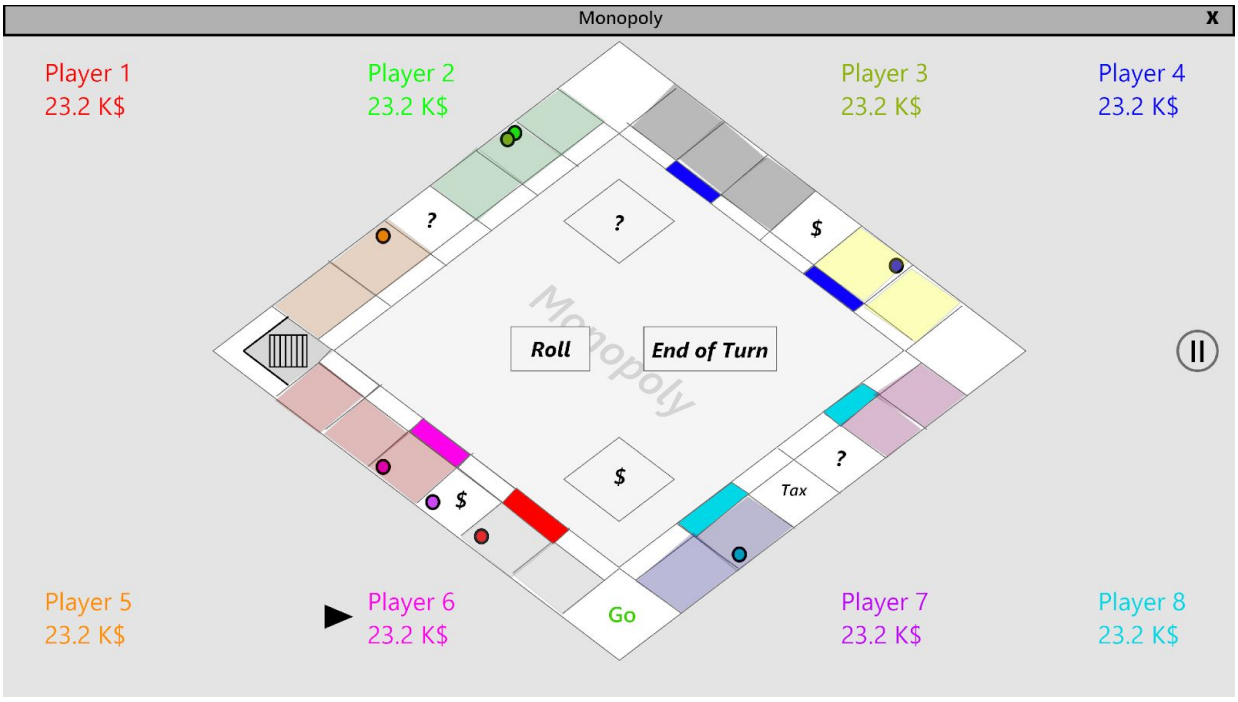

Figure 5.4.5: In-Game Screen of the Classic Mode

5.4.6. In-Game – Property Screen

The place's name is on the top row of the property image. The background color of the place name is the color of the owner's token. The cost of the property, the rent incomes in each different building states, mortgage values and the building costs are shown in this screen. As now the turn is on the **Player 6,** the player can mortgage the property, sell the property, or build buildings on the property.

The **mortgage** option does not require any further confirmation, however, in the **sell** option the buying player must further confirm the transaction. They can also get out of the property screen by clicking the **X** button on the top right of the property card.

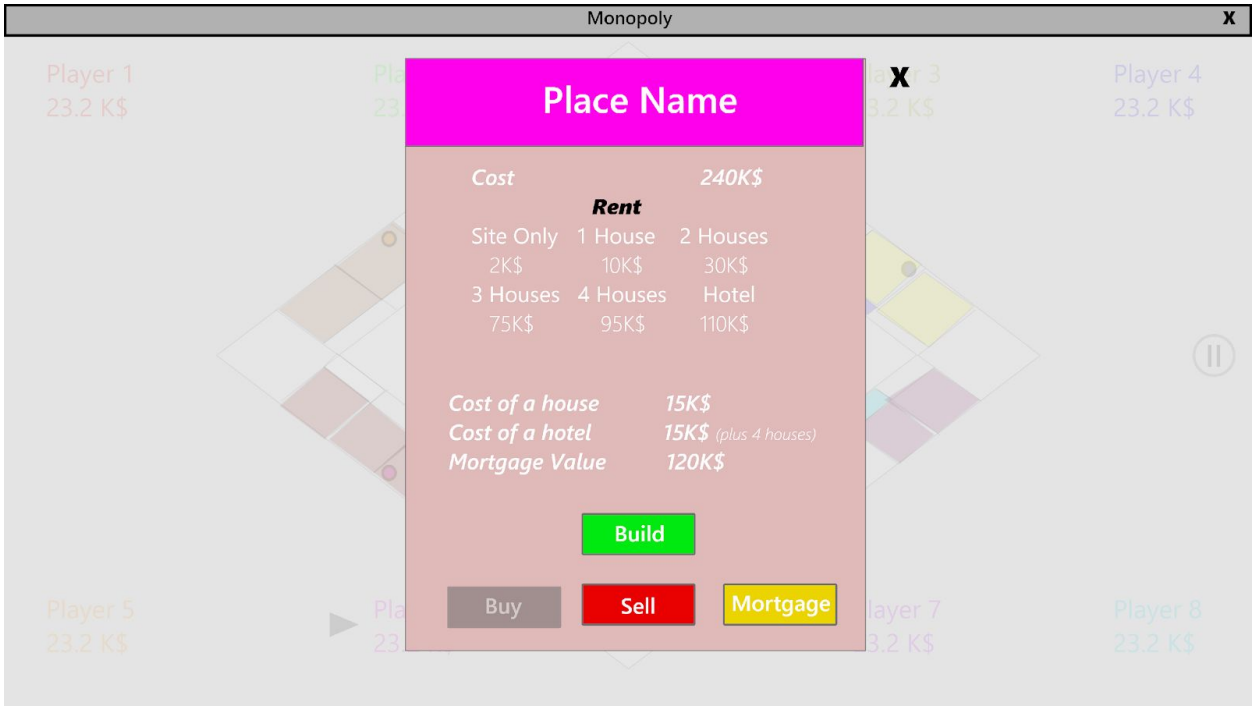

Figure 5.4.6: In-Game - Property Screen

5.4.7. In-Game – Build Screen

As there are not any buildings on the current property, **Player 6** can only build a house. After the build **Player 6** can close the build screen by clicking the **X** button on top right of the property card.

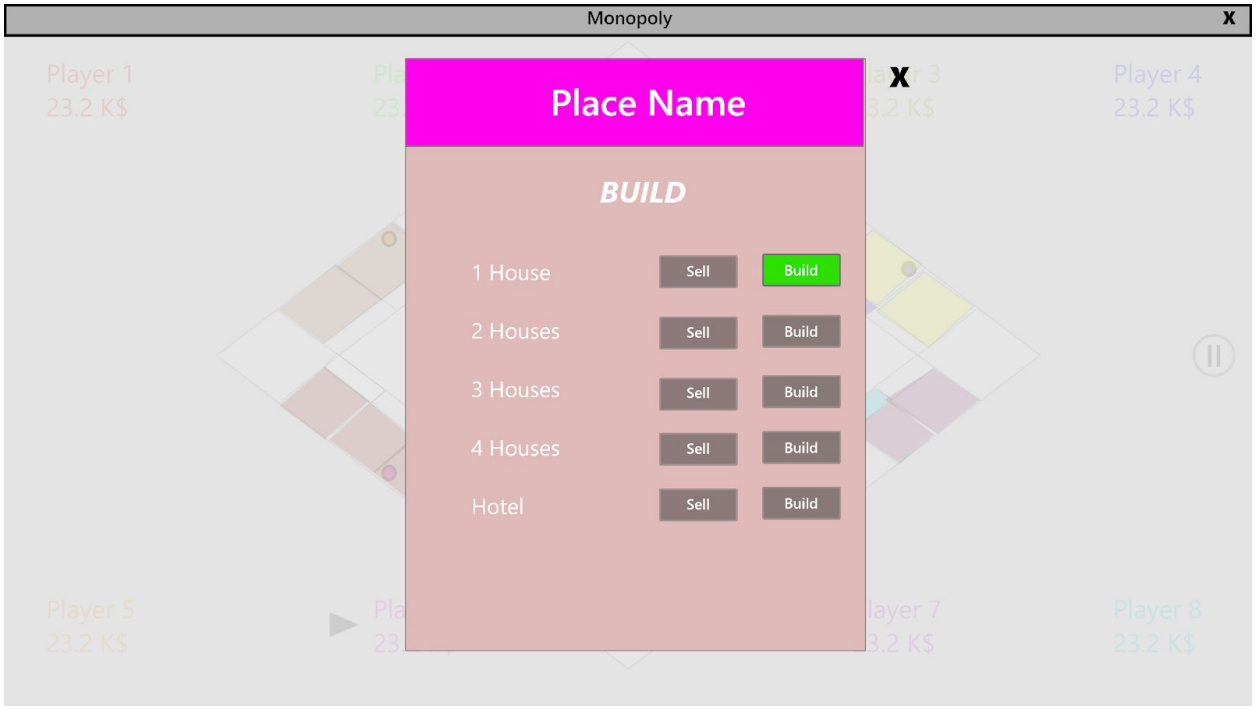

Figure 5.4.7: In-Game - Build Screen

5.4.8. In-Game - Pause Screen

By clicking at the pause button on the **In-Game** screen, the user will navigate to this screen. In this screen the user can save the game, continue playing, or quit the game without saving.

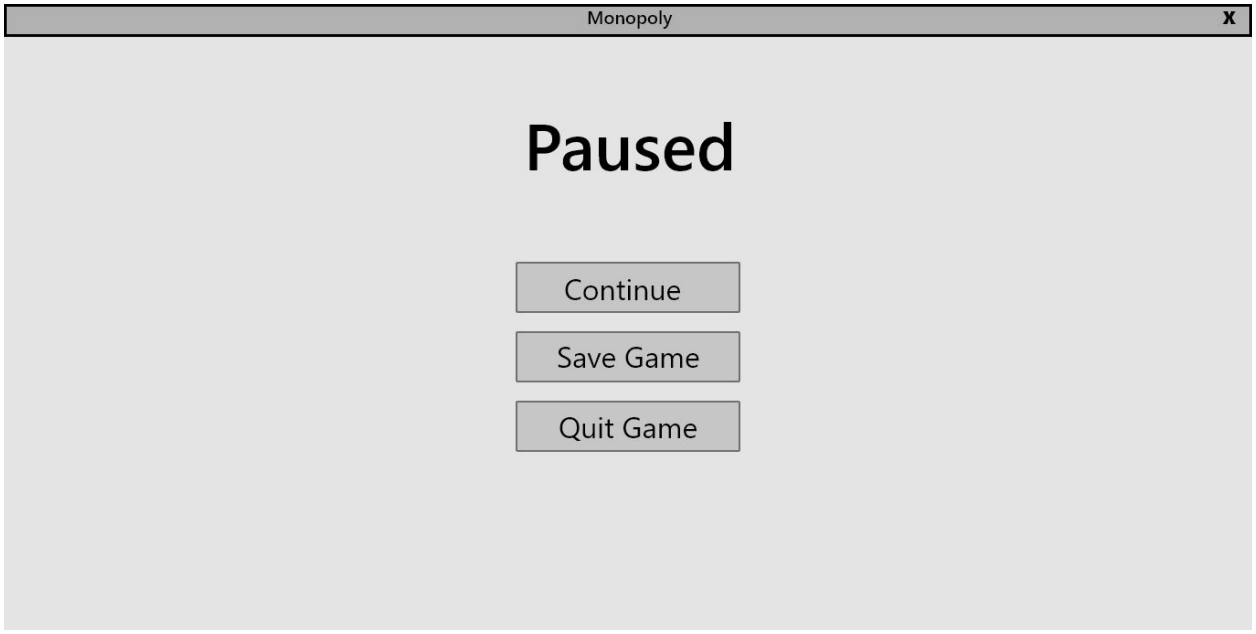

Figure 5.4.8: Pause Menu Screen

5.4.9. Options Screen

In this screen, the user can adjust the volumes of both the music and the SFX. The **save** button saves the current info from the sliders, and the **back** button returns to the **main menu** screen.

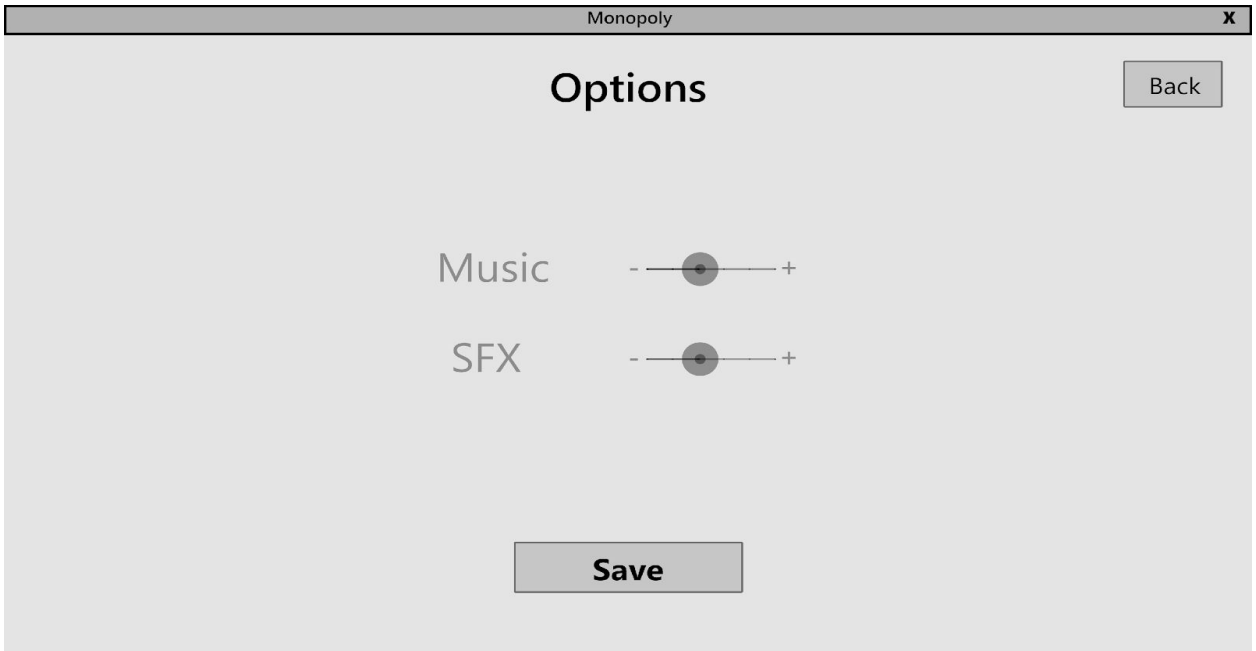

Figure 5.4.9: Options Screen

5.4.10. Credits Screen

In this screen, the user can see the credits of the **Monopoly** game. The **back** button takes the user back to the **Main Menu** screen.

Figure 5.4.10: Credits Screen

5.4.11. How to Play Screen

In this screen, the user can see the rules, the general information about the objects in the game and introductions to the different tools. The **back** button takes the user back to the **Main Menu** screen.

Figure 5.4.11: How to Play Screen

5.4.12. Mode Editor Screen

The user can specify a name for the previously created mode and can select a base mode to work on. The **Edit Mode** button takes the user to the **Mode Editor – Specialization** screen. The back button takes the user back to the **Main Menu.**

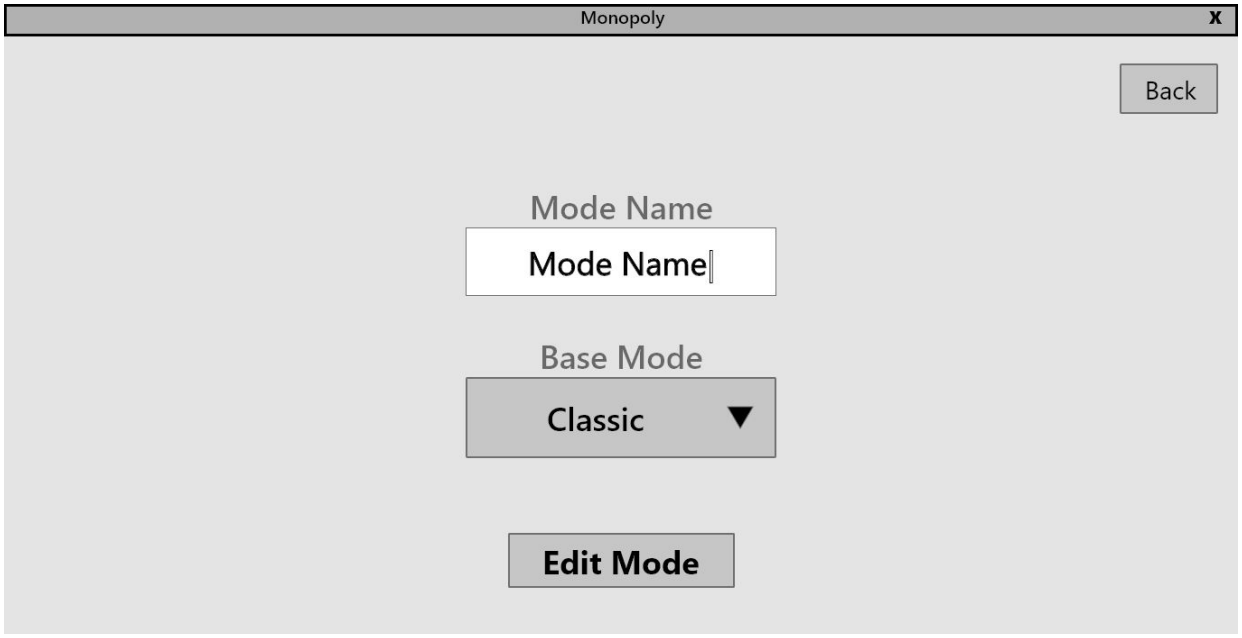

Figure 5.4.12: Mode Editor

5.4.13. Mode Editor – Specialization Screen

By choosing different base modes, the user gets more permission to specify the game in the way that he/she wants. This in-game tool enables users to edit the given six modes. Then, the default game board of this mode will appear. On this board, users will be able to modify each location's name and money related numeric values. For example, users may edit a property's name, rent, price and building costs.

As it is seen in Figure 15, the user can specify only salary and start capital in the **Classic** mode, but the user can specify the blitz period in the **Blitz** mode as it can be seen in the Figure 16, and also they can change the tax calculation in the tax mode. However, users cannot change the functionality of a location. For example, they can't convert a property location to a Railway or a Jail, or they can't change the three turn waiting time in Jail.

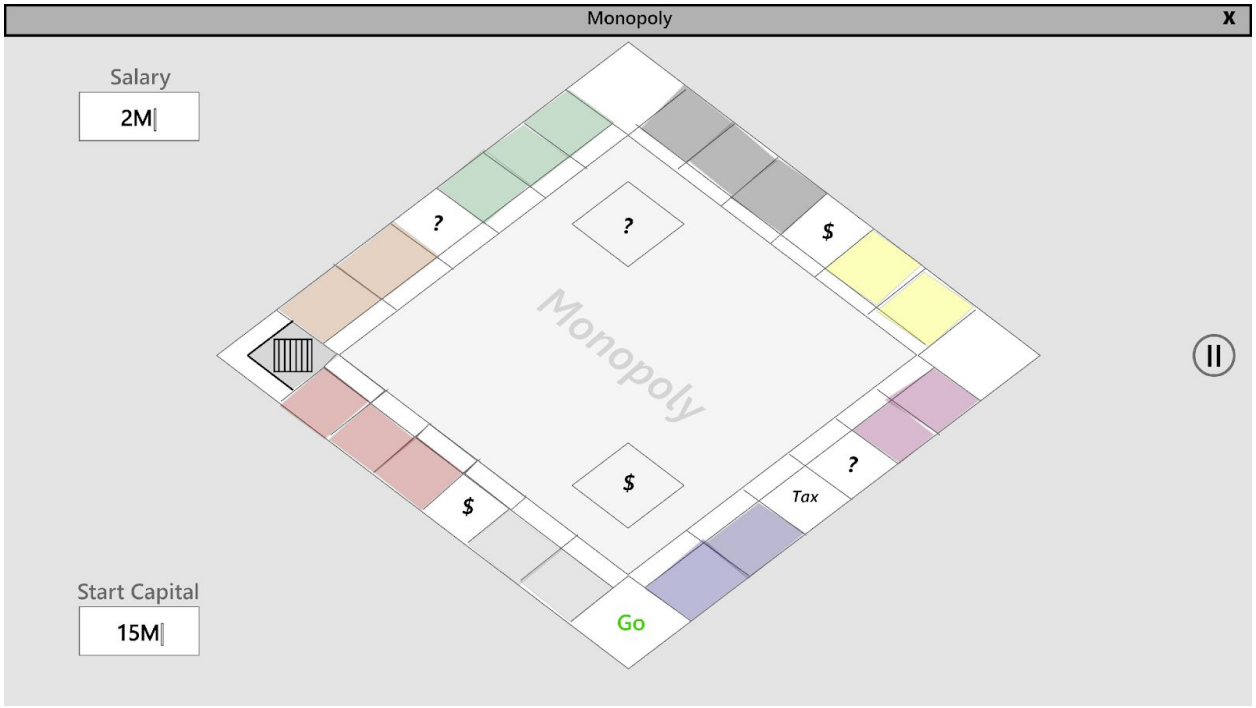

Figure 5.4.13.1: Mode Editor – Specialization – Classic Base Mode

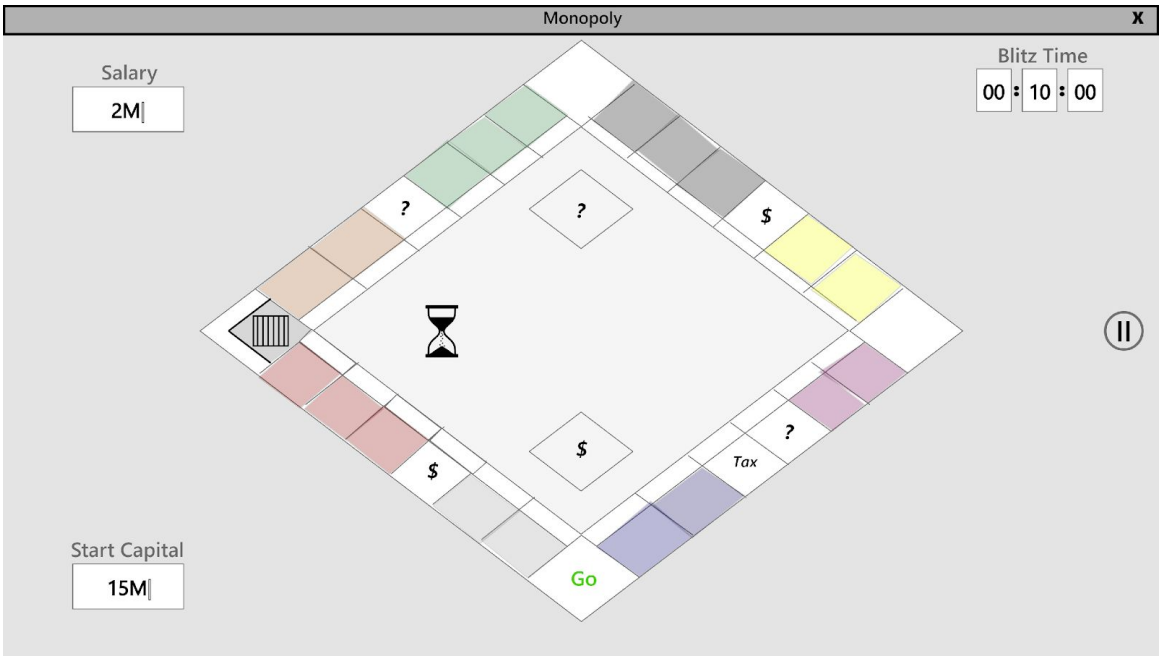

Figure 5.4.13.2: Mode Editor – Specialization – Blitz Base Mode

By clicking on the can-be-specialized properties, the user can change the monetary values, as it opens the property focus screen with text fields. That way the users can interact with the property values without any further knowledge. By clicking at the **Cancel** button, the user will discard any changes that he/she has made. The **Save** button saves the current status of the property.

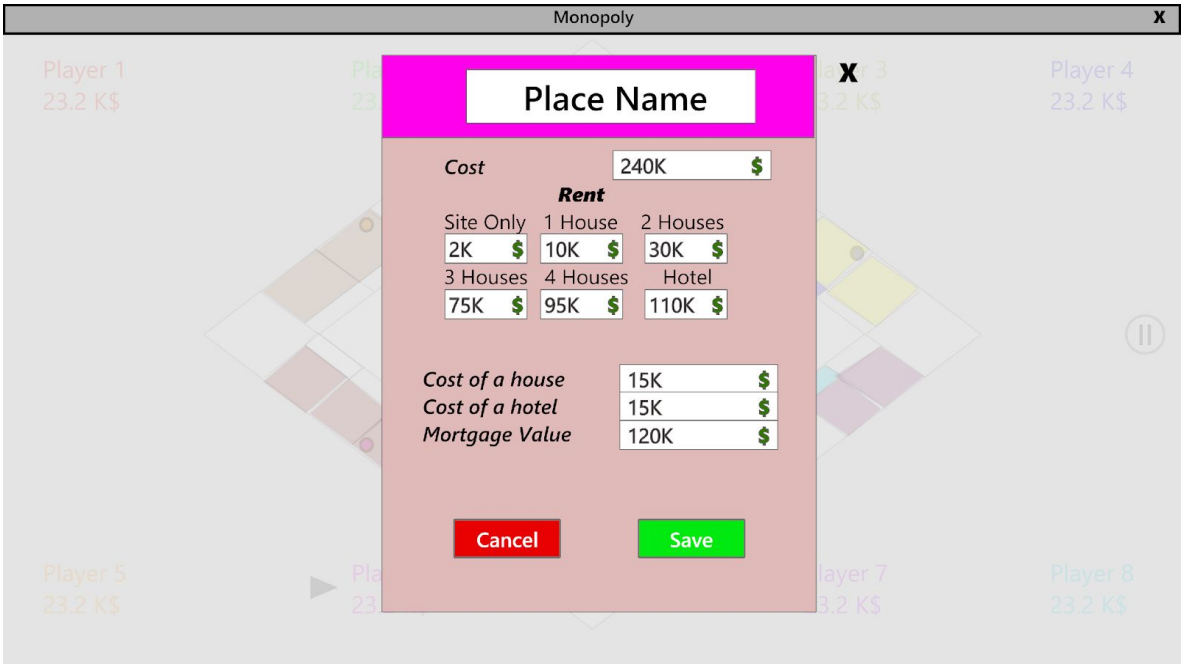

Figure 5.4.13.3: Mode Editor – Property Focus

The pause screen of the mode editor is as below. The user can save the mode or quit the editor. The pause screen is navigated through the pause button on the **Mode Editor – Specialization** screen.

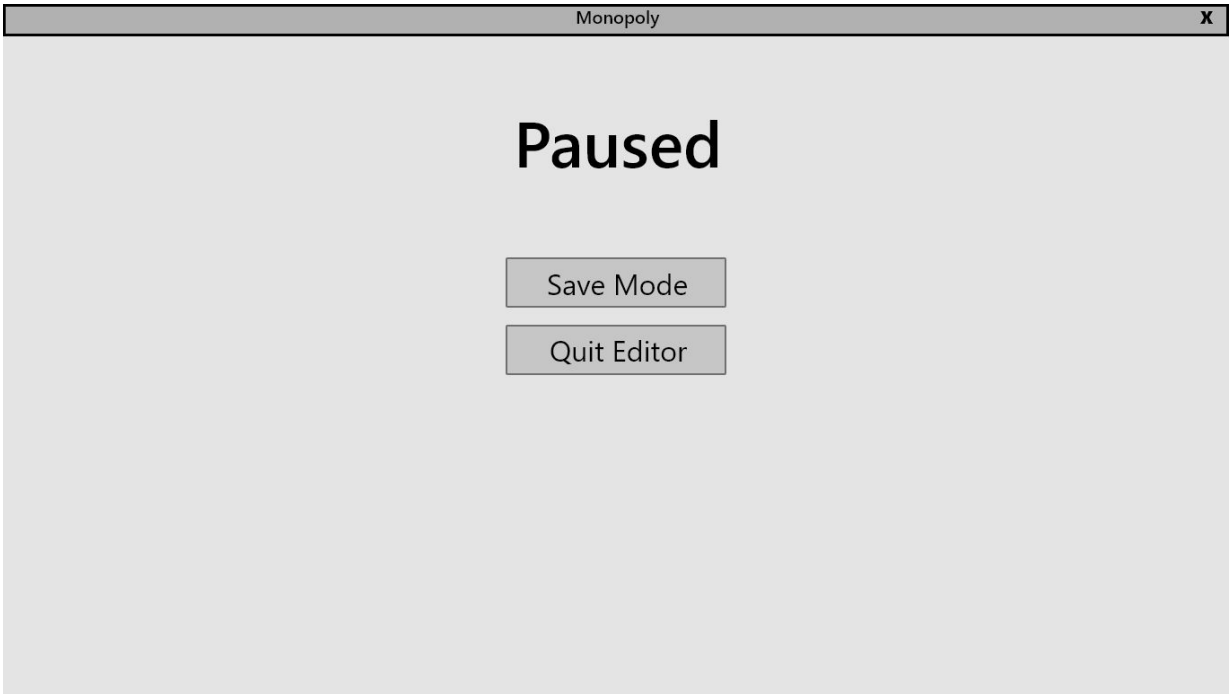

Figure 5.4.13.4: Mode Editor – Pause Focus

5.4.14. Different Mode Mock-ups

Below are some of the modes we are implementing in our term project. These are **Blitz**, **Classic**, and **Reverse**. The mock-ups except the **Classic** mode are almost identical to the **Classic** modes mock-up. However, there are certain visual differences that appear on the screen, as they are different modes indeed and we would like to remind that to the players.

For the **Blitz** mode, each user can see their own remaining time right over their player names or under their cash values.

In the **Reverse** mode, there are two reverse icons in the middle of the board, pointing clockwise and counter-clockwise respectively with different colors to indicate the current flow

direction of the game. Also, there are reverse icons on some of the places, on which the users change the flow direction of the game.

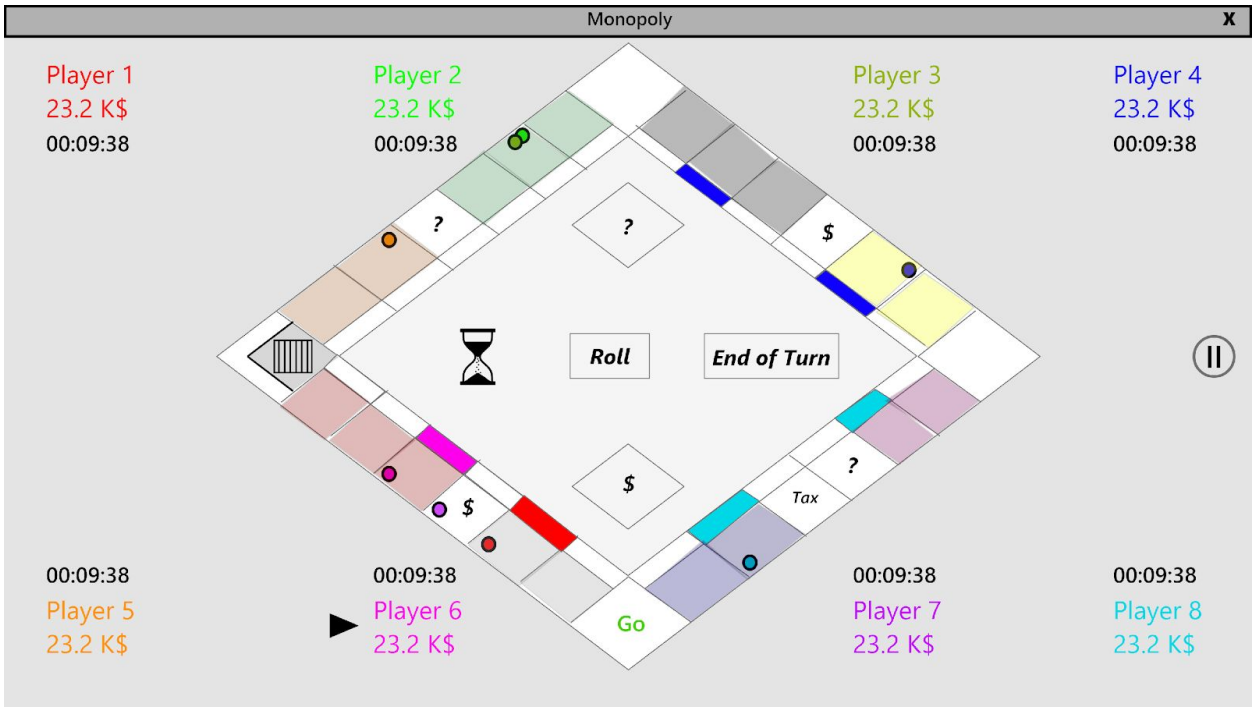

Figure 5.4.14.1: Blitz Mode

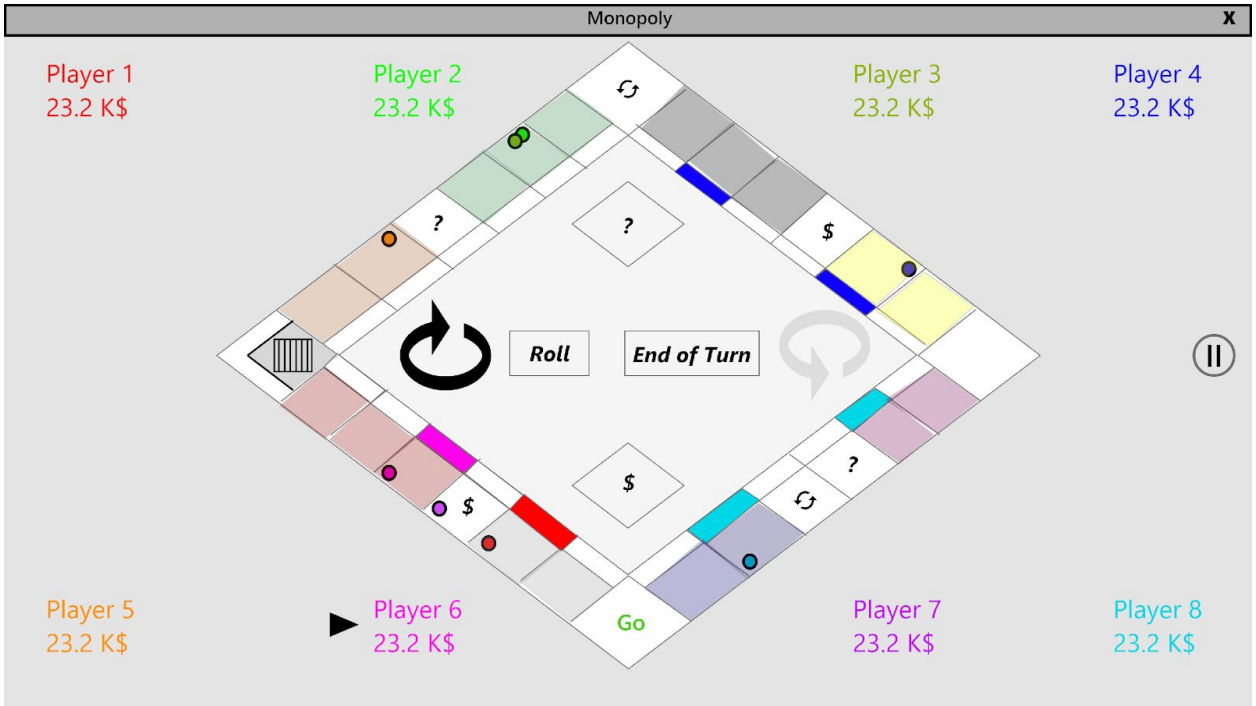

Figure 5.4.14.2: Reverse Mode

6. APPENDIX

6.1. Appendix A

The properties on our default map for all modes will be cities of Turkey. Below is the chart that shows the properties and their corresponding colors and rents:

Table 6.1.1: Turkey Properties and Their Rents

			Rent					
Property	**Cost**	**M'tg**	**Site**	**1 hse**	**2 hses**	**3 hses**	**4 hses**	**Hotel**
Van	60	30	2	10	30	90	160	250
Şanlıurfa	60	30	4	20	60	180	320	450
Erzurum	100	50	6	30	90	270	400	550
Diyarbakır	100	50	6	30	90	270	400	550
Yozgat	120	60	8	40	100	300	450	600
Gaziantep	140	70	10	50	150	450	625	750
Mersin	140	70	10	50	150	450	625	750
Hatay	160	80	12	60	180	500	700	900
Samsun	180	90	14	70	200	550	750	950
Kars	180	90	14	70	200	550	750	950
Kayseri	200	100	16	80	220	600	800	1000
Konya	220	110	18	90	250	700	875	1050
Çanakkale	220	110	18	90	250	700	875	1050
Afyonkarahisar	240	120	20	100	300	750	925	1100
Adana	260	130	22	110	330	800	975	1150

İzmit	260	130	22	110	330	800	975	1150
Bursa	280	140	22	120	360	850	1025	1200
Antalya	300	150	26	130	390	900	1100	1275
Muğla	300	150	26	130	390	900	1100	1275
İzmir	320	160	28	150	450	1000	1200	1400
Ankara	350	175	35	175	500	1100	1300	1500
İstanbul	400	200	50	200	600	1400	1700	2000

The Turkey Cities map is based on the classic game, therefore there are no different kinds of properties.

The World Cities map is only available for the classic mode. Below is the chart that shows the properties and their corresponding colors and rents:

Table 6.1.2: World Properties and Their Rents

			Rent					
Property	**Cost**	**M'tg**	**Site**	**1 hse**	**2 hses**	**3 hses**	**4 hses**	**Hotel**
Venice	60	30	2	10	30	90	160	250
Barcelona	60	30	4	20	60	180	320	450
Amsterdam	100	50	6	30	90	270	400	550
Seoul	100	50	6	30	90	270	400	550
Delhi	120	60	8	40	100	300	450	600
San Francisco	140	70	10	50	150	450	625	750
Cape Town	140	70	10	50	150	450	625	750

Singapore	160	80	12	60	180	500	700	900
Dubia	180	90	14	70	200	550	750	950
Hong Kong	180	90	14	70	200	550	750	950
Vienna	200	100	16	80	220	600	800	1000
Sydney	220	110	18	90	250	700	875	1050
Rome	220	110	18	90	250	700	875	1050
Bangkok	240	120	20	100	300	750	925	1100
Tokyo	260	130	22	110	330	800	975	1150
İstanbul	260	130	22	110	330	800	975	1150
Los Angeles	280	140	22	120	360	850	1025	1200
Montreal	300	150	26	130	390	900	1100	1275
Berlin	300	150	26	130	390	900	1100	1275
Paris	320	160	28	150	450	1000	1200	1400
New York City	350	175	35	175	500	1100	1300	1500
London	400	200	50	200	600	1400	1700	2000

The World Cities map is also based on the classic game, therefore there are no different kinds of properties.

The Bilkent map is only available for the classic mode. Below is the chart that shows the properties and their corresponding colors and rents:

Table 6.1.3: Bilkent Properties and Their Rents

			Rent					
Property	**Cost**	**M'tg**	**Site**	**1 hse**	**2 hses**	**3 hses**	**4 hses**	**Hotel**
East Campus Library	60	30	2	10	30	90	160	250
U Building	60	30	4	20	60	180	320	450
D Building	100	50	6	30	90	270	400	550
Dorm 81/82	100	50	6	30	90	270	400	550
M Building	120	60	8	40	100	300	450	600
V Building	140	70	10	50	150	450	625	750
EB Building	140	70	10	50	150	450	625	750
SB Building	160	80	12	60	180	500	700	900
T Building	180	90	14	70	200	550	750	950
H Building	180	90	14	70	200	550	750	950
C Building	200	100	16	80	220	600	800	1000
NC Building	220	110	18	90	250	700	875	1050
FF Building	220	110	18	90	250	700	875	1050
FB Building	240	120	20	100	300	750	925	1100
G Building	260	130	22	110	330	800	975	1150
FC Building	260	130	22	110	330	800	975	1150
B Building	280	140	22	120	360	850	1025	1200
SA Building	300	150	26	130	390	900	1100	1275
Main Campus Library	300	150	26	130	390	900	1100	1275

EA Building	320	160	28	150	450	1000	1200	1400
A Building	350	175	35	175	500	1100	1300	1500
EE Building	400	200	50	200	600	1400	1700	2000

The Bilkent map is a slight variation of the classic game; the properties are the campus buildings and not cities.

6.2. Appendix B

Turkey Railroads and Canals

The railroads on our default map for all modes will be from the Turkey Cities map. Below is the chart that shows the railroads and their corresponding colors and rents:

Table 6.2.1: Turkey Railroads with Rents

Railroad Station	Cost	M'tg	Rent
Haydarpaşa Station, İstanbul	200	100	25 or 50 or 100 or 200
Ankara Station, Ankara	200	100	25 or 50 or 100 or 200
Merinos Station, Bursa	200	100	25 or 50 or 100 or 200
Alsancak Station, İzmir	200	100	25 or 50 or 100 or 200

The Turkey Cities map is based on the classic game so we used railroads for that map.

World Railroads and Canals

The World Cities map contains canals. Below is the chart that shows the canals and their corresponding rents:

Table 6.2.2: World Railroads with Rents

Canals	Cost	M'tg	Rent
Suez Canal	200	100	25 or 50 or 100 or 200
The English Channel	200	100	25 or 50 or 100 or 200
The Bosphorus	200	100	25 or 50 or 100 or 200
Panama Canal	200	100	25 or 50 or 100 or 200

The World Cities map is based on the classic game and the canals are similar in type to railroads.

Bilkent Restaurants/Cafeterias

The Bilkent map contains cafes and restaurants. Below is the chart that shows the cafes and their corresponding rents:

Table 6.2.3: Bilkent Railroads with Rents

Cafes and Restaurants	Cost	M'tg	Rent
Speed	200	100	25 or 50 or 100 or 200
Mozart Cafe (B)	200	100	25 or 50 or 100 or 200
Cafe In	200	100	25 or 50 or 100 or 200
Tabldot Cafeteria	200	100	25 or 50 or 100 or 200

The Bilkent map is a slight variation of the classic game; there are cafeterias and restaurants instead of railroads.

6.3. Appendix C

These are the community chest cards that are included in the Turkey Map (other maps will be similar only with different names or prices for locations):

- Advance to "Go". (Collect $200)
- Bank error in your favor. Collect $200.
- Doctor's fees. Pay $50.
- From sale of stock you get $50.
- Get out of Jail Free -This card may be kept until needed.
- Go to Jail. Go directly to jail. Do not pass Go, Do not collect $200.
- Grand Opera Night. Collect $50 from every player for opening night seats.
- Holiday. Fund matures. Receive $100.
- Income tax refund. Collect $20.
- It is your birthday. Collect $10 from every player.
- Life insurance matures – Collect $100
- Hospital Fees. Pay $50.
- School fees. Pay $50.
- Receive $25 consultancy fee.
- You are assessed for street repairs: Pay $40 per house and $115 per hotel you own.
- You have won second prize in a beauty contest. Collect $10.
- You inherit $100.

6.4. Appendix D

These are the chance cards that are included in the game (maps will be similar only with different names or prices for locations):

- Advance to "Go". (Collect $200)
- Advance to Çanakkale. If you pass Go, collect $200.
- Advance to Gaziantep. If you pass Go, collect $200.
- Advance token to nearest Utility. If unowned, you may buy it from the Bank. If owned, throw dice and pay the owner a total 10 (ten) times the amount thrown.
- Advance token to the nearest Railroad Station and pay the owner twice the rental to which he/she is otherwise entitled. If Railroad is unowned, you may buy it from the Bank. *(There are 2 of these.)*
- Bank pays you a dividend of $50.
- Get out of Jail free. This card may be kept until needed, or traded/sold.

- Go to Jail. Do not pass GO, do not collect $200.
- Make general repairs on all your property: For each house pay $25, For each hotel pay $100.
- Pay speeding fine of $15.
- Take a trip to the Haydarpaşa Railroad Station. If you pass Go, collect $200.
- Advance token to İstanbul. You have been elected Chairman of the Board. Pay each player $50.
- Your building loan matures. Collect $150.
- You have won a crossword competition. Collect $100.

7. GLOSSARY & REFERENCES

[1] Monopoly.fandom.com. 2020. *Monopoly Wiki*. [online] Available at: <https://monopoly.fandom.com/wiki/Main_Page> [Accessed 23 October 2020].