



Bilkent University

Department of Computer Engineering

CS 319

Object-Oriented Software Engineering

Term Project

Section 2

Group 2F

Monopoly

Final 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	2
2.	Lessons Learnt	3
3.	Work Allocation	4
4.	User's Guide	5
4.1.	Play Game	6
4.2.	Load Game	7
4.3.	Player Specialization	7
4.4.	In-Game Screen	8
4.5.	In-Game - Property Screen	9
4.6.	In-Game - Pause Screen	10
4.7.	Options	11
4.8.	Credits	12
4.9.	How to Play	12
4.10.	Mode Editor Screen	12
5.	Build Instructions	13

1. INTRODUCTION

As Group 2F who is building a computer version of the game Monopoly, we stand close to finishing the game. Our implementation is working well at the moment and the core game itself is playable. All major requirements of the game such as buying, selling, movement, etc. are working so the gameplay is quite fine at the moment. The UI is also functioning well and we are able to perform operations like saving, loading and quitting the game.

As to the part of the implementation we have not yet finished, we are still working on finishing some parts of the UI (which has working functionality). We also haven't gotten to finishing either the logic or the UI for the new game modes (blitz and reverse) and the mode editor for allowing users to customize their game. We also don't have the new map types that we have mentioned before. We thought it best to make these a lower priority as they could have cost us time in building the most crucial parts of the game. This is why we saved them for last and it has paid off as we have built a game that is playable. Not finishing these unique features of our game can cost us originality or uniqueness and so we will try to finish these as soon as possible.

2. LESSONS LEARNT

During the analysis, design and implementation stages of this project, we have learned many things, some much more harshly than others.

Firstly, we have seen that not all of our finalized design plans and especially class designs translated so well onto the coding process. We've encountered many difficulties implementing the design patterns we tried to apply to our classes. It is clear that even some of the best design choices can create problems during the implementation and that small flaws can definitely lead to bigger problems being created. We should have been more realistic rather than optimistic and should have prepared better for it. Either our designs could still be improved or we could have worked out the complications earlier.

We also definitely learned that we need to make a better plan of action and distribute tasks in a much more organized way. Throughout the project we always remained in communication and we tried to be clear on what we meant as much as possible which worked well but sadly wasn't enough. We did not manage the distribution of our tasks very well, meaning some people got too much work sometimes and sometimes we didn't split it into the necessary tasks. It is especially hard to manage tasks when we do not pay attention and decide to give everyone the job of knowing everything about all the code. Instead of separating the implementation into certain tasks and code segments for everybody, we all focused on learning most of the code, which did end up helping with the finish of the project but definitely cost us time and efficiency. All in all, we should have given certain tasks to each member and had one or two middlemen that helped bring it all together afterwards.

We also learned that, although it should be done with small steps, coding is a process that should be started with the beginning of the design stage. This is one of our failures. We focused very hard on our design and while we did come up with a very doable and good design (that still had its flaws) but since we focused so much on it, it wasn't until the end of the process that we really started to work on our implementation, which did cost us some time.

3. WORK ALLOCATION

Adil:

- Analysis report → class diagram
- Design report → object diagram
- Façade and Singleton design patterns
- Most of the logic and some part of the UI in the implementation

Doğa:

- General designs and formatting of reports
- Analysis report → introduction and functional requirements
- Design report → system decomposition and design goals
- Location classes in the implementation
- Final report
- Presentations and demo video

Burak:

- Analysis report → some part of functional requirements
- Analysis report → non-functional requirements
- Analysis report → activity and state diagrams
- Design report → design tradeoffs and boundary conditions
- Design report → class interfaces
- Card Strategy Pattern design and implementation

Ömer Faruk:

- Analysis report → use case diagrams and use case explanations
- Analysis report → sequence diagrams
- Design report → persistent data management
- Help to Adil during the implementation of the UI and logic

Osman Batur:

- Analysis report → mockups
- Design report → hardware software mapping and access control security
- Design report → packages
- Most of the UI

4. USER'S GUIDE

As the .jar file for the game (the deliverable) has not been implemented yet, in order to open the game, users can follow the Build Instructions section to download and install the game. Then they can follow this guide to proceed.

1. Choose one of the following and press the button:
 - a. **Quit:** Makes you close the window of the game
 - b. **Play Game:** Play a new or saved game
 - c. **Mode Editor:** Edit and customize maps and modes to make your own version of the game
 - d. **Options:** Change technical features of the game like sound and light
 - e. **How to Play:** Opens the How to Play document to explain how the gameplay works
 - f. **Credits:** Credits of the game



Figure 4: Main Menu Screen

4.1. Play Game

When the user clicks the **Play Game** button in the **Main Menu**, this screen will come up. The user can do the following:

1. Specify **the player count**, and **the game mode** from the drop down menu below the name. There will be default placeholders for **the player count**, and **the game mode** fields in their own respective drop down menus.
2. The user may choose the **Back** button to navigate to the **Main Menu**.
3. The user can choose the **New Game** button to specify the game properties further in the **Player Specification** screen.
4. The **Load Game** button will take the user to the **Load Game** screen.

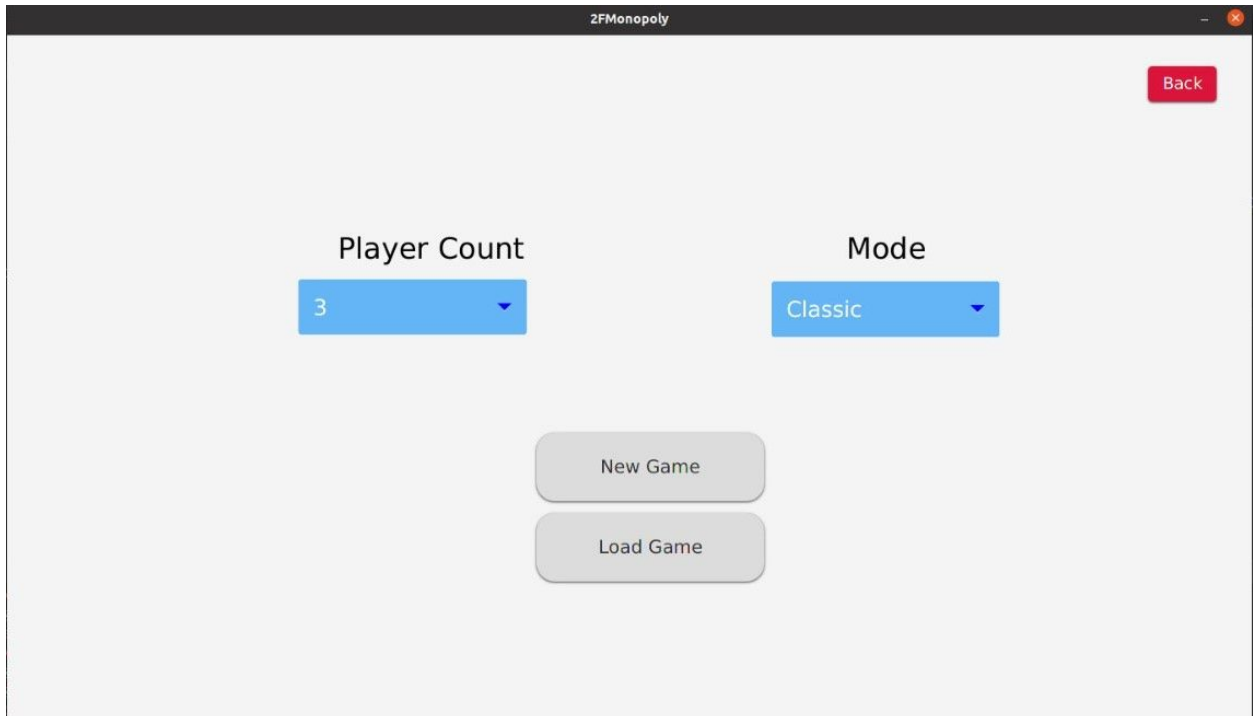


Figure 4.1: Play Game Screen

4.2. Load Game

In this screen, the user can do the following:

1. Select a previously **saved game** to continue playing.
 - a. 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.
2. The user may choose to navigate to the **Play Game** screen by clicking the **Back** button.

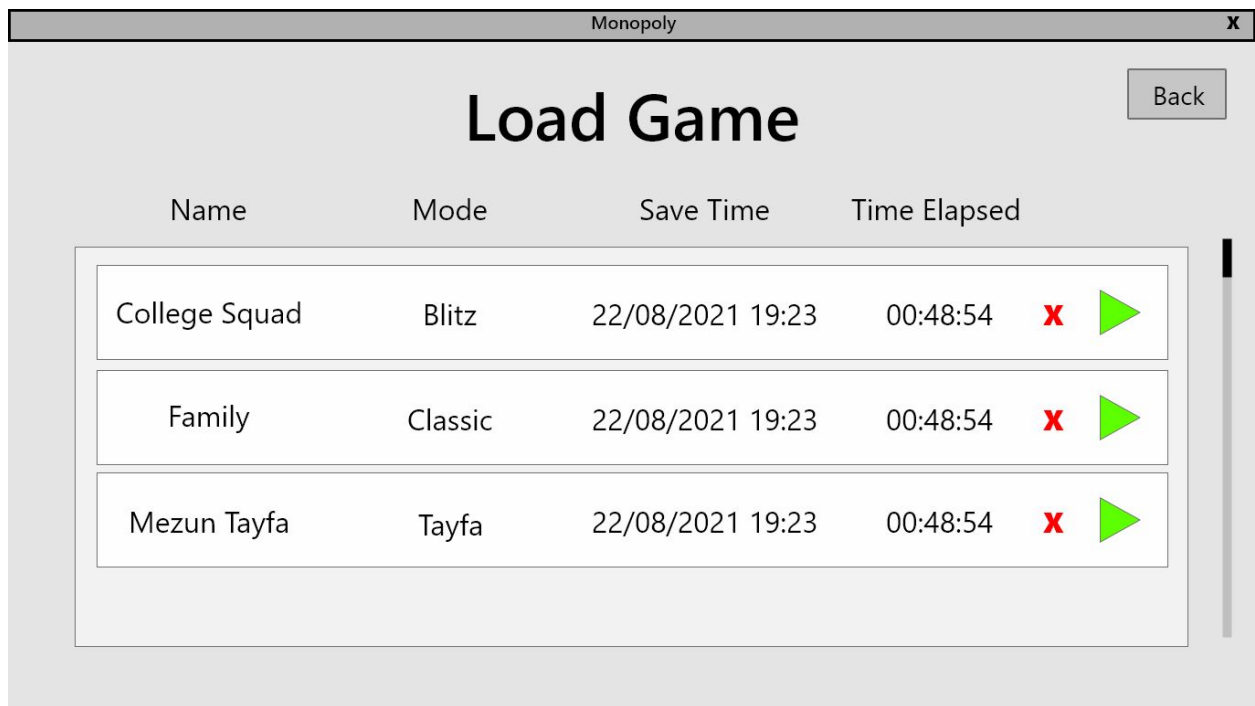


Figure 4.2: Load Game Screen

4.3. Player Specialization

In this screen, the user can do the following:

1. Specify the **names** of each player, choose their **tokens**, and confirm their changes.
 - a. The game cannot be started before confirming all changes.

2. Click the **start** button to enter the **in-game** screen.
3. The **back** button navigates to the **Play Game** screen.

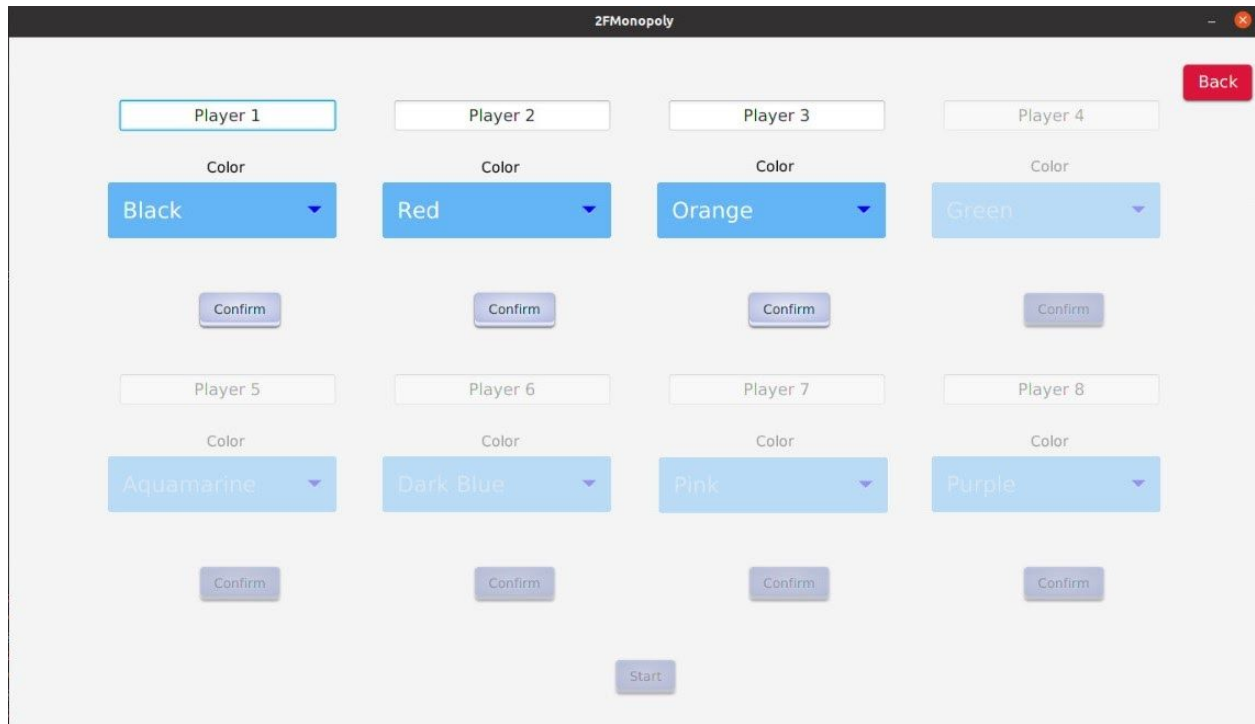


Figure 4.3: Player Specialization Screen

4.4. In-Game Screen

When the game starts, it is a player's turn. In this turn, the following will happen:

1. Each player gets a **unique color** with the **tokens** they have chosen and it forms the color of their **name** and **cash value**.
2. A **sideways black triangle** will appear next to the player whose turn is on.
3. The **pause** button on the right can open a dialog to the pause menu where the game can be closed or continued, etc.
4. 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.

5. The user can **roll the dice** and then continue to take actions such as buying houses. The user must click **end of turn** to end his or her turn and pass the turn to the next player. The user can make **further arrangements** as long as he/she does not roll the dice.
6. 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.
7. 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**.

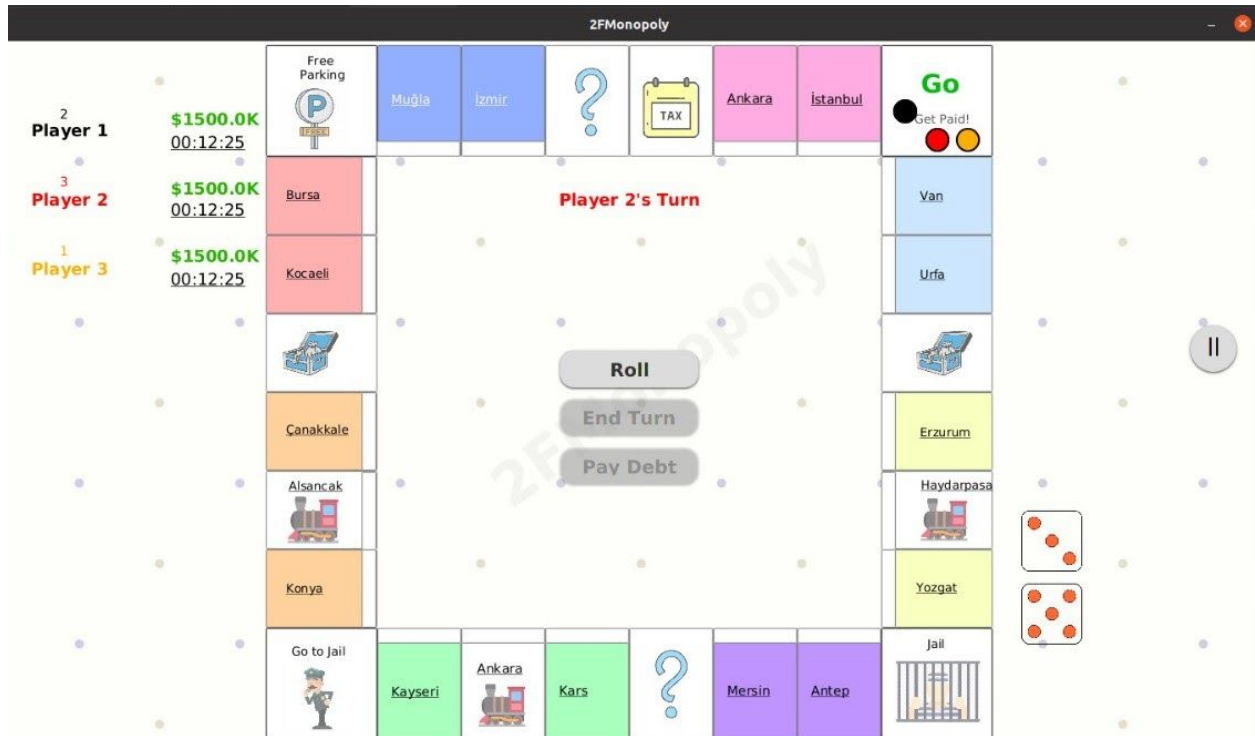


Figure 4.4: In-Game Screen of the Classic Mode

4.5. In-Game – Property Screen

1. **Buy:** Buy the property you have arrived on.
2. **Sell:** Sell the buildings on the property.

- a. in the **sell** option the buying player must further confirm the transaction.
3. **Mortgage**: Mortgage the property.
4. **Unmortgage**: Unmortgage the property.
5. **Build**: Build buildings on the property.
6. **Close**: Press the **X** button to close the property deed image.

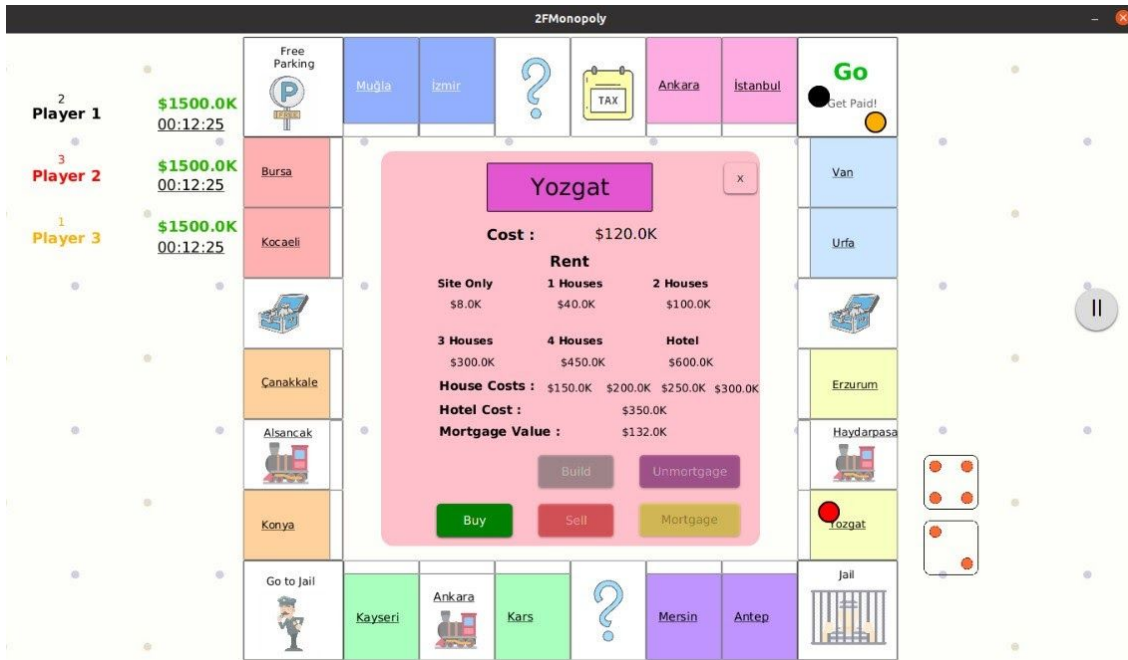


Figure 4.5: In-Game - Property Screen

4.6. 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 do the following:

1. **Save Game**: Saves the game
 - a. This option opens a window where the name of the game has to be entered to save the game.
2. **Back to Game**: Continues playing
3. **Quit Game**: Quits the game without saving.

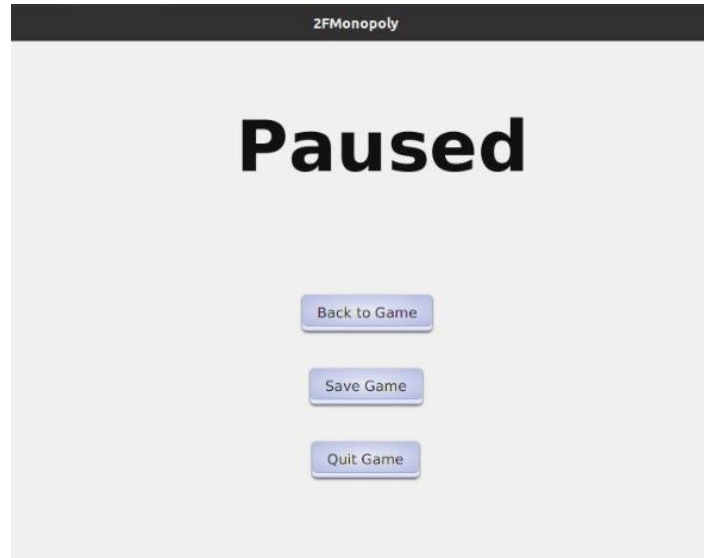


Figure 4.6: Pause Menu Screen

4.7. Options

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.

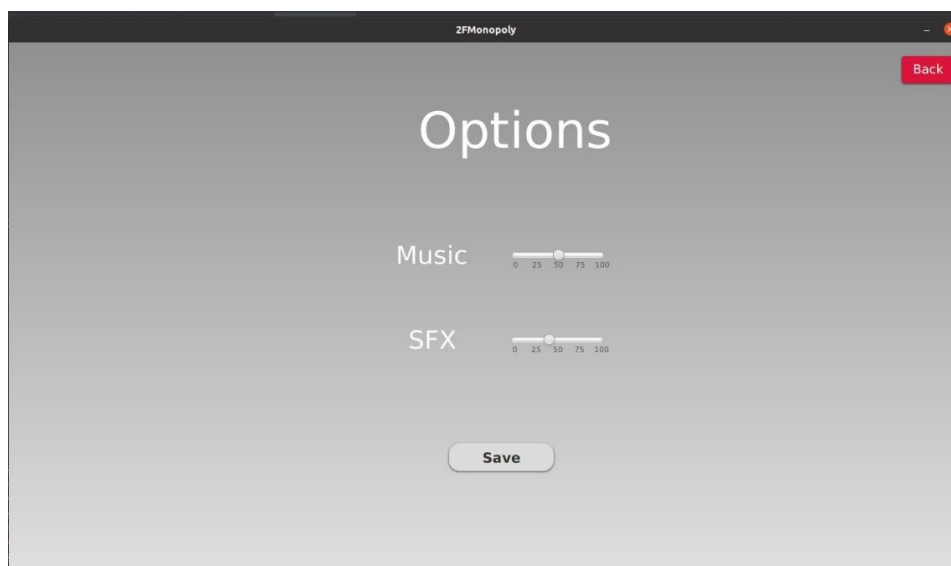


Figure 4.7: Options Screen

4.8. Credits

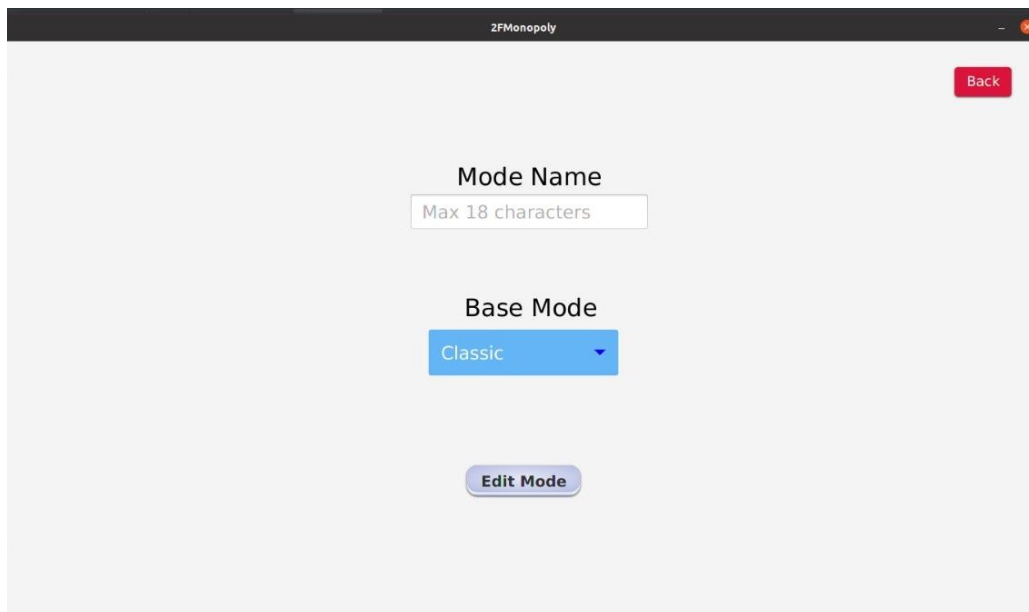
In this screen, the user can see the credits of the **Monopoly** game. The **back** button takes the user back to the **Main Menu** screen.

4.9. How to Play

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.

4.10. Mode Editor Screen

1. **Mode Name:** Specify name for the previously created mode
2. **Base Mode:** Select a base mode.
3. **Edit Mode:** Takes user to the **Mode Editor – Specialization** screen.
4. The back button takes the user back to the **Main Menu**.



The screenshot shows a web browser window with the title '2FMonopoly'. The page has a light gray background. In the top right corner, there is a red 'Back' button. The main content area contains three elements: a 'Mode Name' label above a text input field with a placeholder 'Max 18 characters'; a 'Base Mode' label above a blue dropdown menu currently showing 'Classic'; and an 'Edit Mode' button at the bottom.

Figure 4.10: Mode Editor

5. BUILD INSTRUCTIONS

For someone to build this game with the source code, they must execute the following steps:

1. Clone the source code of the game from GitHub:
<https://github.com/mericadil/2FMonopoly.git>
2. Open the project in your favorite IDE (we use IntelliJ as our IDE and it is recommended)
3. Make sure that you have installed at least JDK 8 and JavaFX library. JavaFX library must be set as the dependency of the project.
4. Build and run the game.

When the project is cloned and opened in the IDE, it should look like this:

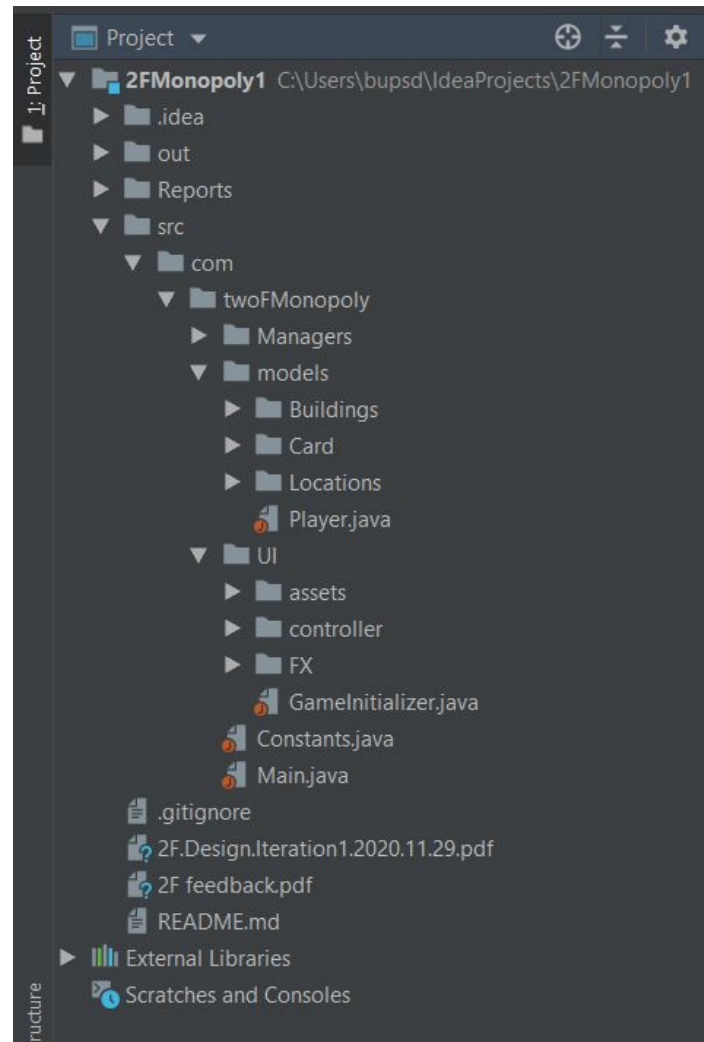


Figure 5: Project Layout in IDE

Note: The main function is in the Main.java class which is visible in Figure 5.