



CS 319 - Object-Oriented Software Engineering
Final Report

Monopoly

Group 2-H

Aziz Ozan Azizoğlu	21401701
Mert Sayar	21601435
Veli Can Mert	21602394
Hammad Khan Musakhel	21801175
Gökhan Mullaoglu	22001086

Contents

Implementation Process	3
Changes in Design	5
User's Manual	5
Work Allocation of the Team	11
Lessons Learnt	12

1) Implementation Process

Implementation of the Monopoly game started right after the submission of iteration 2 of analysis and design reports. For the implementation process, we shared the MVC architecture among our group members and started implementation. Implementation is being done in parallel to our class, state, sequence and activity diagrams. Right now, we are in the phase of GUI implementation of our project.

Mert Sayar and Aziz Ozan Azizoğlu were in charge of model and controller parts of the architecture where Hammad Khan Musakheil, Veli Can Mert and Gökhan Mullaoğlu were in charge of the view (GUI) part. First, the model part of the project which consists of Board, Player, Cell and Card objects were implemented. Afterwards, the controller object which is the GameManager object in our case was implemented. The GameManager can be determined as the brain of the game which controls and computes the necessary operations for the monopoly game since all of the cell handling, paying rent, buying property, mortgaging or drawing card operations were implemented inside the GameManager class. Also, the GameManager object was also responsible for the communication between the GUI and the game system. The one-direction communication between the model and view/controller(from controller to GUI) were handled by observer interface as GUI classes subscribed to the observer arraylist of the GameManager class and specific commands were sent to all observers from the controller. It can be any command which requires user interaction. For instance, if a player lands on a property which is not owned by any other player, the controller will send listeners a string message “buy”. Whenever the listener receives the message, GUI class will enable the “Buy Property” button. After this communication, we will need a reverse way communication in order to handle the user activity. In order to handle the user activity, each class which listens to the controller will have an instance of the controller which is designed as Singleton. For instance, if the user presses the “Buy Property” button, the related function from the controller instance will be called.

Beyond our plans, the implementation process is struggling for us since the group was not able to meet even once because of the extraordinary circumstances. Communication via digital platforms are not as efficient as face to face meetings. Another disadvantage was that the group members did not know each other beforehand and the team was formed via Slack Channel since the members were unable to join a group. However, we are working hard in order to overcome these cons and complete our Monopoly Game as it should be. We see this

implementation process of the Monopoly Game as a crucial opportunity to be prepared for our professional lives and to gain experience.

2) Changes in Design

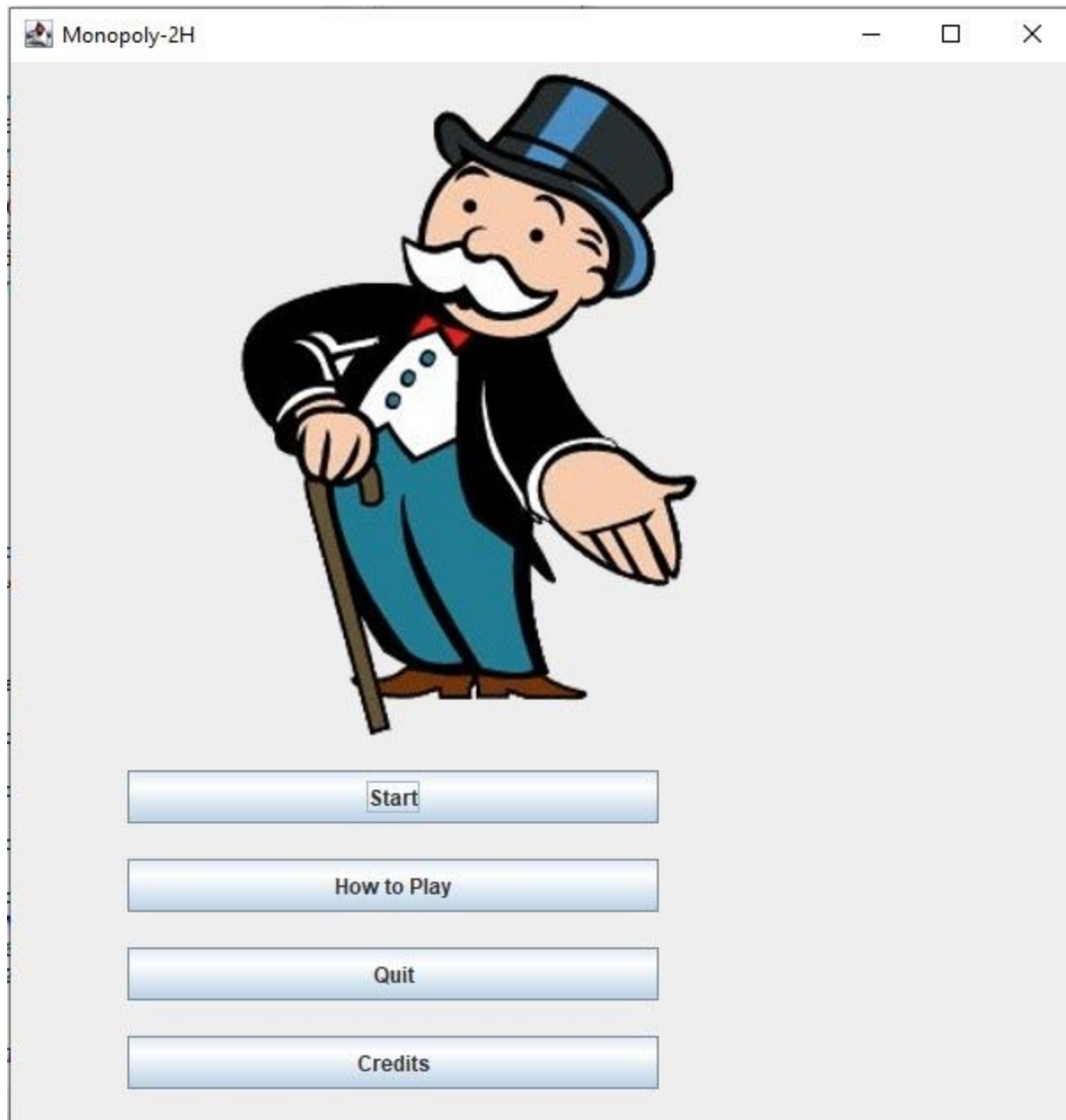
In our design report, we said that we would have two boards and a bridge in between these boards and it would be a new mechanic. However, we couldn't have this feature in our game because it would change the game mechanics more than we expected so we decided to not have this feature. Additionally, we added a different mechanism to our design; players can only build house/hotel on and mortgage properties when they are on that property cell. This change can affect the game mechanics.

In the design report, we said that we would use the javafx library package but we used java swing and java awt in our graphical user interface design. They are more functional than javafx for our game design.

3) User's Manual

The Monopoly game would be readily available at the Git repository for public access; one can download the zip file, extract the file containing the source code, and then run it on a JDK that needs to be installed beforehand; the JDK must have the Java and Java swing libraries for proper execution of the game.

Following the execution of the game, the user would be presented with a Main Menu as shown below:

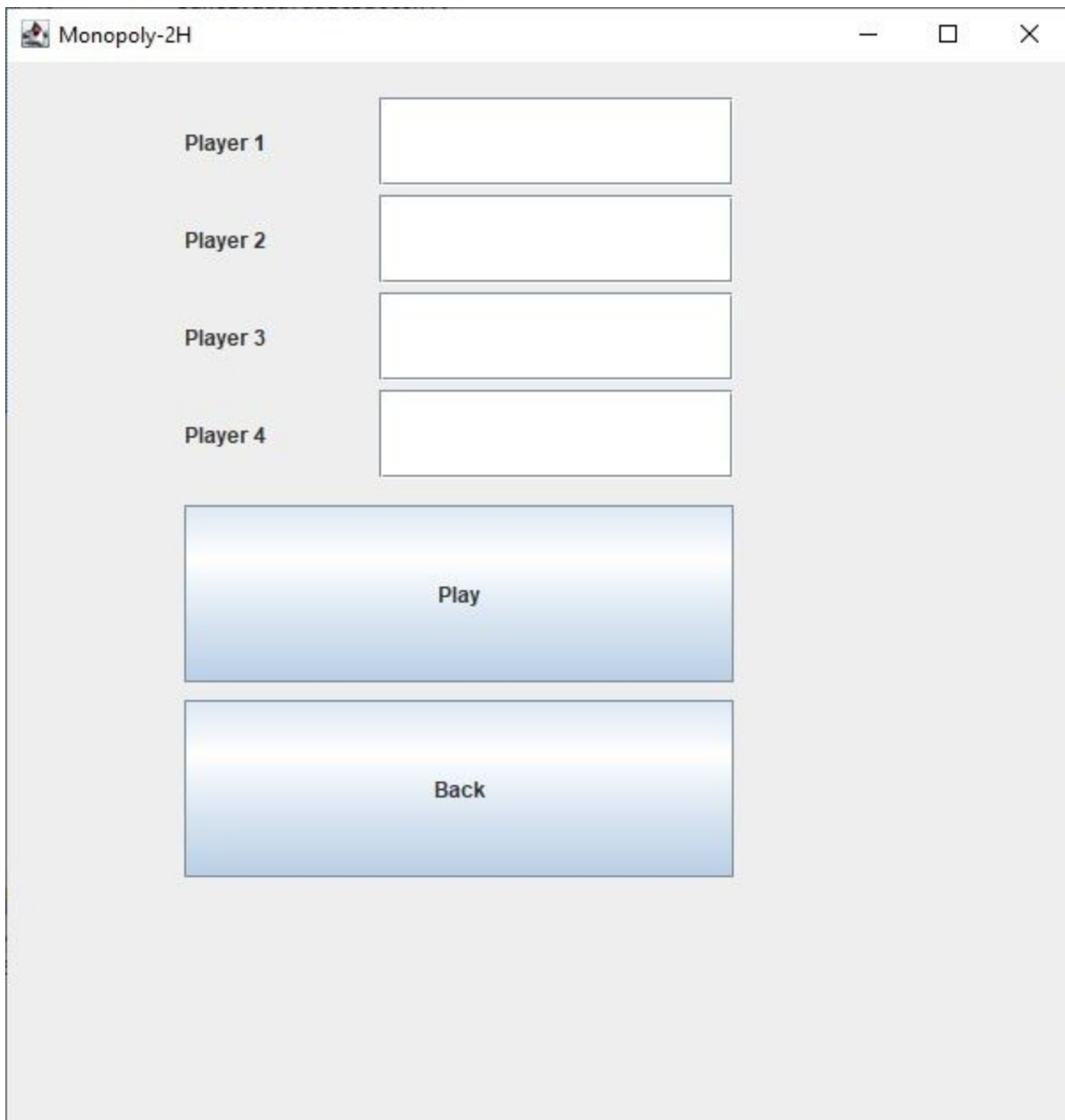


From the Menu, the user can direct oneself to the area of the game they desire to proceed. The Main Menu only has four buttons: Start, How To Play, Quit, and Credits. The button labels are self explanatory themselves thus to game is expected to be pretty basic and understandable for starters. The Credits button would lead to a window where the contributions of the developers would be stated and will have a back button as well to return to the Main Menu;

Aziz Ozan Azizoğlu
Hammad Khan Musakhel
Veli Can Mert
Gökhan Mullaoglu
Mert Sayar

Back

The start button would take the user to the window where the names of the users would be required to start off the game. The screenshot below shows the display of that window:



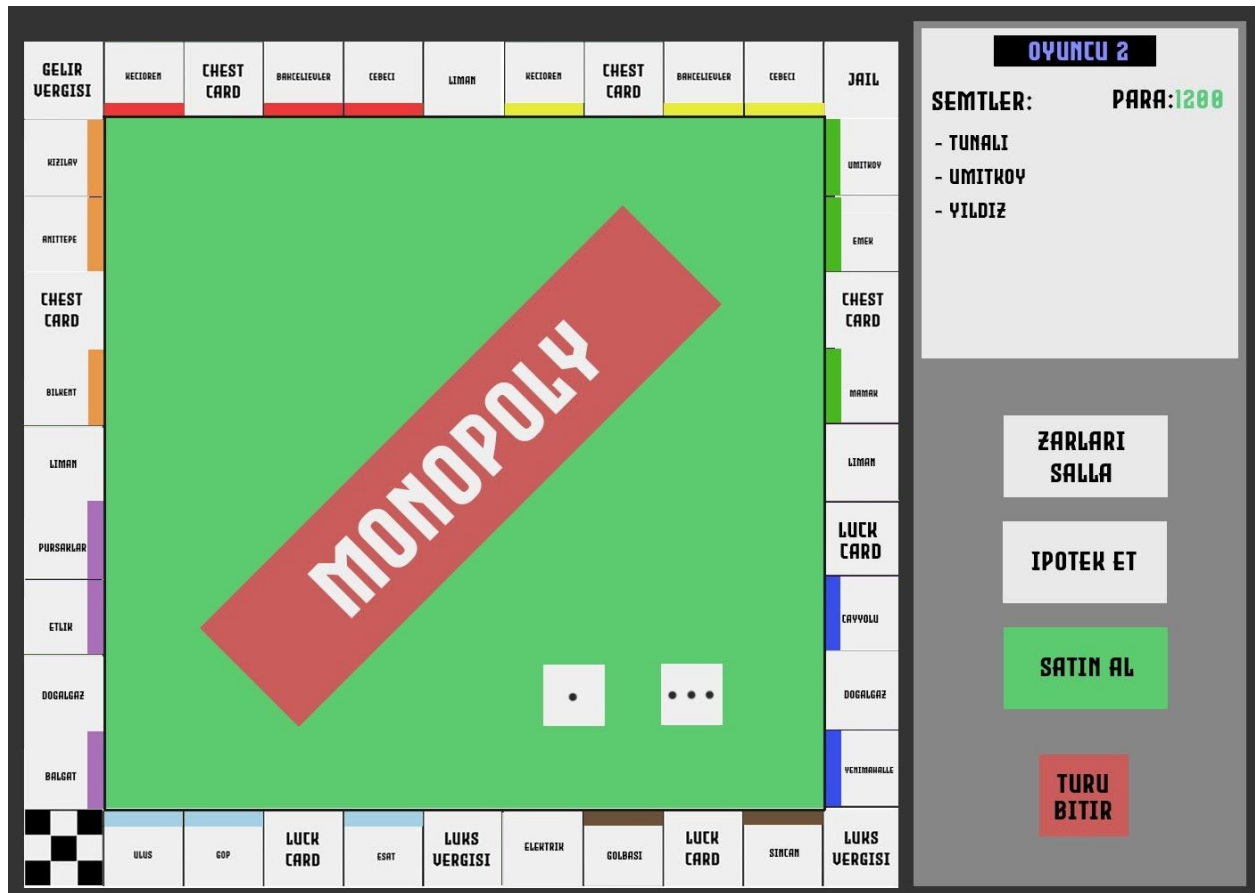
The screenshot shows a window titled "Monopoly-2H" with standard Windows window controls (minimize, maximize, close). The window has a light gray background. On the left side, there are four labels: "Player 1", "Player 2", "Player 3", and "Player 4". To the right of each label is a white rectangular input field. Below these input fields are two large, light blue buttons with a gradient and a drop shadow. The top button is labeled "Play" and the bottom button is labeled "Back".

Player 1	<input type="text"/>
Player 2	<input type="text"/>
Player 3	<input type="text"/>
Player 4	<input type="text"/>

Play

Back

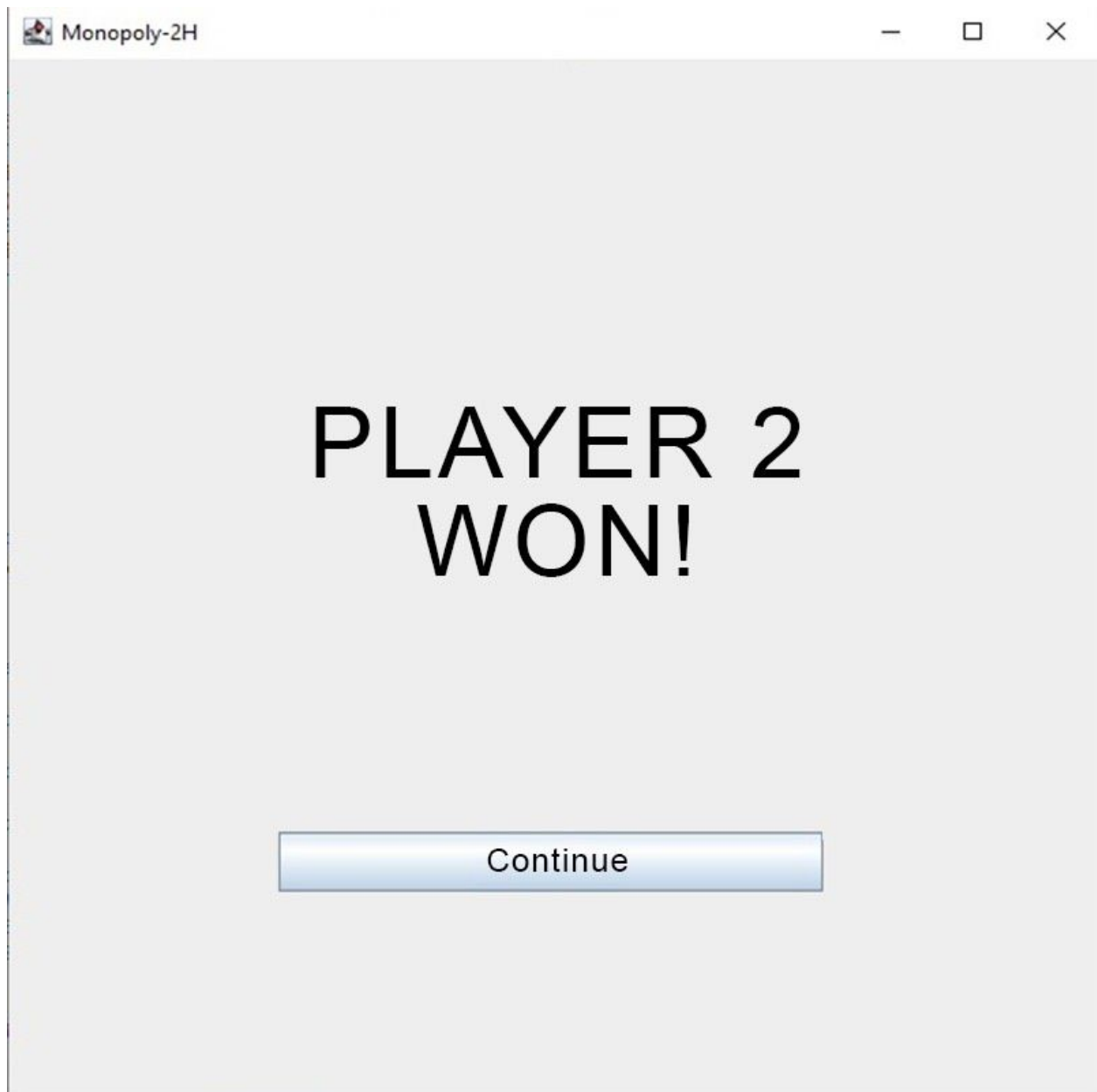
The names entered would be the names assigned to each player during gameplay time and that is how each player would be able to keep a record of their property, money, and much more. The name is a string literal, so one can use any type of characters. The Play button would proceed with the gameplay and present the user with the board. The board is the crucial and most important part of the game. It is where the users are going to play and perform activities to compete and gain the victory. The game board presented below is in animate form as we are still in the middle of finalising a unique and intriguing board:



The board would be also very self explanatory as one can easily assess it. If one needed help with the understanding of the game/board rules and procedures, one can choose the ‘How To Play’ button from the Main Menu; it would lead the user to a window with rules and procedures. The board will have the specific buttons and associations to proceed with the game. We have counselled to keep the game board immensely elaborate and work day and night on it as it is the

most integral GUI feature of the game. We aim to keep the users engaged and impressed so that they can revisit our game for further entertainment.

The end screen of the game would be like this:



The continue button would take the user to the Main Menu again where the user can choose to play again or quit the game entirely.

Monopoly is a relatively simple game with fixed features and rules all over the world. The aim is to have a non-crashing, bug free software for the game with an intriguing board.

4) Work Allocation of the Team

Aziz Ozan Azizoglu

- Design of the class diagram, for analysis report.
- Design of application domain class diagram, explanations for each class attributes and methods in design report.
- Implementation of model and controller architecture of the monopoly game and implementation of the communication between the view and controller architecture.

Hammad Khan Musakhel

- Design of Sequential Diagrams for the Analysis Report, Rules and Regulations of the game (Introduction Part) for the Analysis Report.
- High Level Software Architecture for the Design Report: Subsystem decomposition, Hardware/Software Mapping, Persistent Data Management, Access, Control & Security, and Boundary Conditions.
- Implementation of the GUI/View component of the Monopoly architecture.
- User's Manual for Final Report.

Veli Can Mert

- Design of activity diagrams and explanation of activity diagrams for analysis report.
- Explanation of purpose of the system, design goals and subsystem services for design report.

- Implementation of the GUI (mainly windows and game board) of the Monopoly architecture.

Gökhan Mullaoglu

- Worked on design of the project architecture, made decisions for the design patterns, design trade-offs and external & internal packages.
- Worked on the Low-level software design part for the design report, explanation of the model and controller architecture attributes and methods.
- Worked on the “Game Design” and ideation part of the project. Designed the Game Board, User Interface/View component and mockups as well as made design decisions for the game rules and limitations.

Mert Sayar

- Design of use case diagram, solution domain class diagram and state diagrams for analysis report.
- Design of application domain class diagram, subsystem decomposition and design pattern integration such as observer, singleton and factory patterns for design report.
- Implementation of model and controller architecture of the monopoly game and implementation of the communication between the view and controller architecture.

5) Lessons Learnt

In conclusion, we used MVC (Model View Controller) architecture to implement this Monopoly Game project. Aziz Ozan Azizoğlu and Mert Sayar were in charge of the Model and Controller part of the project. Hammad Khan Musakhel, Veli Can Mert and Gökhan Mullaoglu were in charge of View implementation by using java.swing and java.awt language instead of javafx libraries because they found these libraries more functional and understandable to

implement the graphical user interface. Since all group members are not living in Ankara and we have some restrictions due to the pandemic, we faced some hardships to meet, discuss and code face to face. We've learnt that we should divide the duties as everyone makes the same effort. Although some tasks take a really long time, some tasks are not that much compelling. Since some of us have low spec computers and IntelliJ IDEA requires more processing power, we preferred Eclipse for implementation. Other than that, it is very important to have a defined class diagram in order to have the perfect layout for the commencement of the project. UML diagrams are very integral in software development as it provides the developer with the blueprint of the entire software development requirements. Sequence diagrams are very helpful to visualize an object's actions in any scenario.