Bilkent University

Department of Computer Engineering

CS319 Term Project

Katamino

Analysis Report

Group 1H

Ege Özcan, Mustafa Bayraktar, Simge Tabak, Yağız Mertol, Zeynep Gözel
Supervisor: Eray Tüzün

Progress Report
October 25, 2018

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Term Project course CS319.

1. Introduction

In our project for CS319, we will implement the geometrical mind game Katamino. Katamino is a 2D mind game, where the player tries to fill the board with the given blocks in different colours and geometrical shapes. The classic Katamino game is setup by dividing the board into two parts. After that both players are given five blocks and 4 filler blocks, the first player to fill their section with the given blocks is the winner of the game.[1] The main objective of this project is to make this a desktop game using the fundamentals of object oriented programming. We will use Java to implement this real life puzzle game on desktop.

2. Overview

At the beginning of the game, player chooses the game level and category from the main screen and begins the game. Different boards for easy, medium and hard levels will appear according to the player's selections. Player should fill the board with the given blocks. Board designs will feature pixel art that will give an extra joy to the main target audience that are children and also adults. The users can add their own pixel art as an additional feature as well. Mouse is used to drag the blocks from the given blocks onto the board. After the block is chosen by using left click, it can be rotated ninety degrees clockwise by using the right click.

In addition, a time counter will keep count of the time from the beginning of the level. A move counter will keep count of the moves done by the user in the current level. These will be used to determine the score of the player at the end of the game. An offline leaderboard will allow the user to see their score and compare it with other users that played on the same device. Moreover, we will implement a hint system, which will guide the user towards a solution. As explained above in detail, our contributions to the classical Katamino game are mainly allowing user to play the game on different pixel art boards and create their own board. Time

challenge and move counts also make our game different from the classical Katamino game.

3. Functional Requirements

- Start a new game
- Enter user name
- Create board using level creator
- Draw board picture
- Save custom board
- Select board
- Import board
- Export board
- Rotate block
- Drag a block to the board
- Drop a block the board
- Use hint button
- Display leaderboard
- Exit game
- Return to the main menu from anywhere in the game

3.1 Additional Requirements

After the first iteration of our analysis report and considering the feedback we received in our demo, we decided to add a new feature to our game, which allows the player to import and export the boards they designed. This will make the game more fun since you can play with your friend's board and your friend can play with the board you designed. Moreover, we will change the colors we used into more vivid and distinguishable colors.

4. Non-functional Requirements

- Performance: All user inputs should be acknowledged within 0.5 second
- Performance: CPU usage should be less than 20%
- Usability: Can be played without a tutorial
- Reliability: A system crash should not result in data loss
- Portability: We want our game to work on most common operating systems such as Windows, Linux, MacOS .
- Board text files must be smaller than 2KB

5. System Models

5.1. Use Case Model of Katamino HAK3

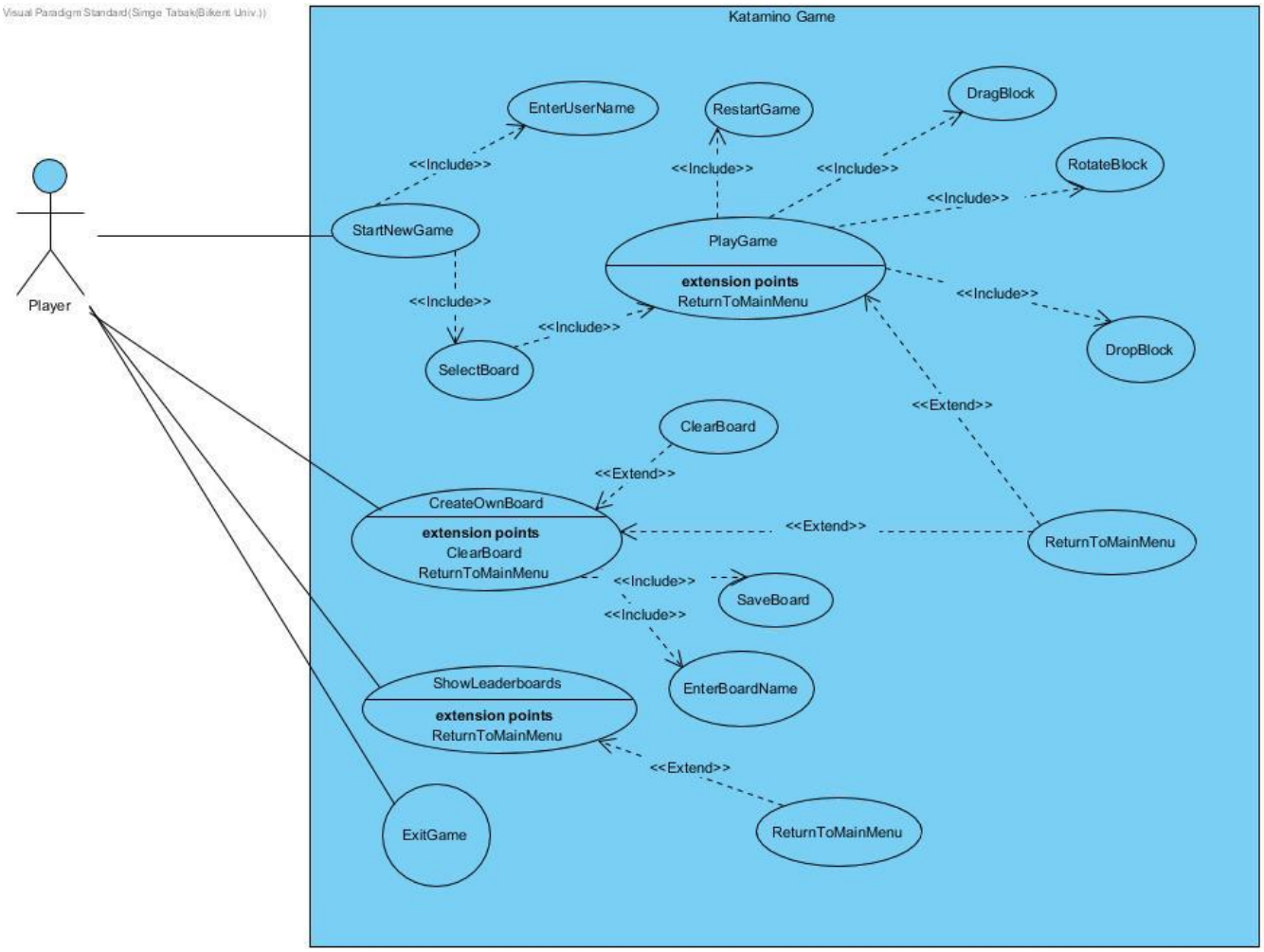


Figure 1: Use Case Model

Use Case Scenario 1

Name: Start a new game

Participating Actor: Player

Entry Condition: Player must be in the main menu

Exit Condition:

Player clicks exit button

Player finishes the game successfully

Flow of Events:

1. Player clicks "Play Katamino" button from the main menu.
2. Player enters name.
3. Player selects a board.
4. Player drags a block to the board.
5. Player drops the block to replace the block.
6. Player repeats steps 4-5.
7. Player finishes the game.
8. Controller ends the game

Special Requirements: None

Use Case Scenario 2

Name: Create board using level creator

Participating Actor: Player

Entry Condition: Player must be in the main menu

Exit Condition:

Player clicks exit button.

Player click save board button.

Flow of Events:

1. Player clicks "Create Your Board" button from the main menu.
2. Player draws a board.
3. Player tests the board.
4. Player saves the board.
5. Player enters board name.

Special Requirements: None

Use Case Scenario 3

Name: Show Leaderboard

Participating Actor: Player

Entry Condition:

Player must be in the main menu

Game must be over

Exit Condition:

Player clicks exit button.

Flow of Events:

1. Player clicks "Show Leaderboard" button from the main menu.
2. Leaderboard is shown.

Alternative Flow of Events:

1. The game ends successfully.
2. Leaderboard is shown.

Special Requirements: None

5.2 Dynamic Models

5.2.1 Activity Diagram

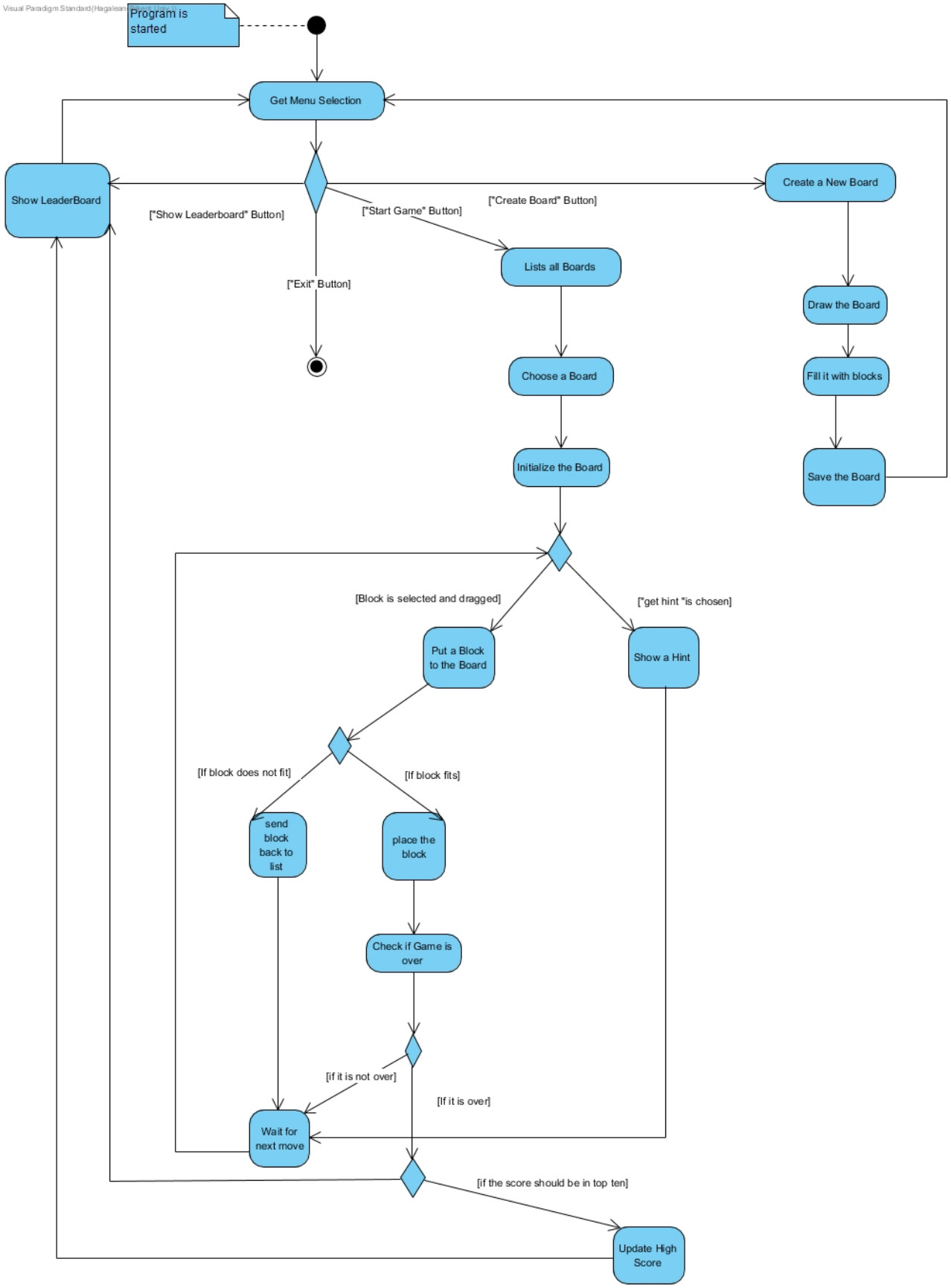


Figure 2: Activity Diagram

At the beginning, the program displays a menu which includes start a new game, create a new board, show leaderboard and exit buttons. Then it waits for an input from the user. There are different cases at that point. At each button selection, the user will be faced with different scenarios.

If the user chooses to start a new game, the program lists all playable boards to the user and waits for a selection. After the user selects a board from the list, the program initializes the board. This board will be the game which includes the blocks and the shape of the board, hint button and a timer.

Now the user may either try to put a block to the board or get a hint.

- If the user decides to put a block to the board, the program checks if it is suitable for the place where the user wanted to put.
    - If the block fits, it is placed on the board and then program checks if the game is over or not.
        - If it is over program updates the leaderboard with the data of the game and shows it to the user. This leaderboard sends the user directly to the main menu.
        - If it is not over, the program waits for a new move.
    - If the block does not fit, the program sends the block back to the block list, and then waits for a new move.
- If the user decides to get a hint, the program shows a hint to the user and then starts to wait for a new move.

Another case is the user choosing to create a new board. After the user clicks on the create a new board button, the program displays a new empty board to the user to

draw a new shape. After the user draws the board, this board is checked with blocks. At the end, the program saves this board.

If the user clicks on the show leaderboard button, the program displays the leaderboard to the user and then it will send the user to main menu.

The last case is clicking on the exit button. If the user clicks on the exit button, program will shut down.

5.2.2 Sequence Diagrams

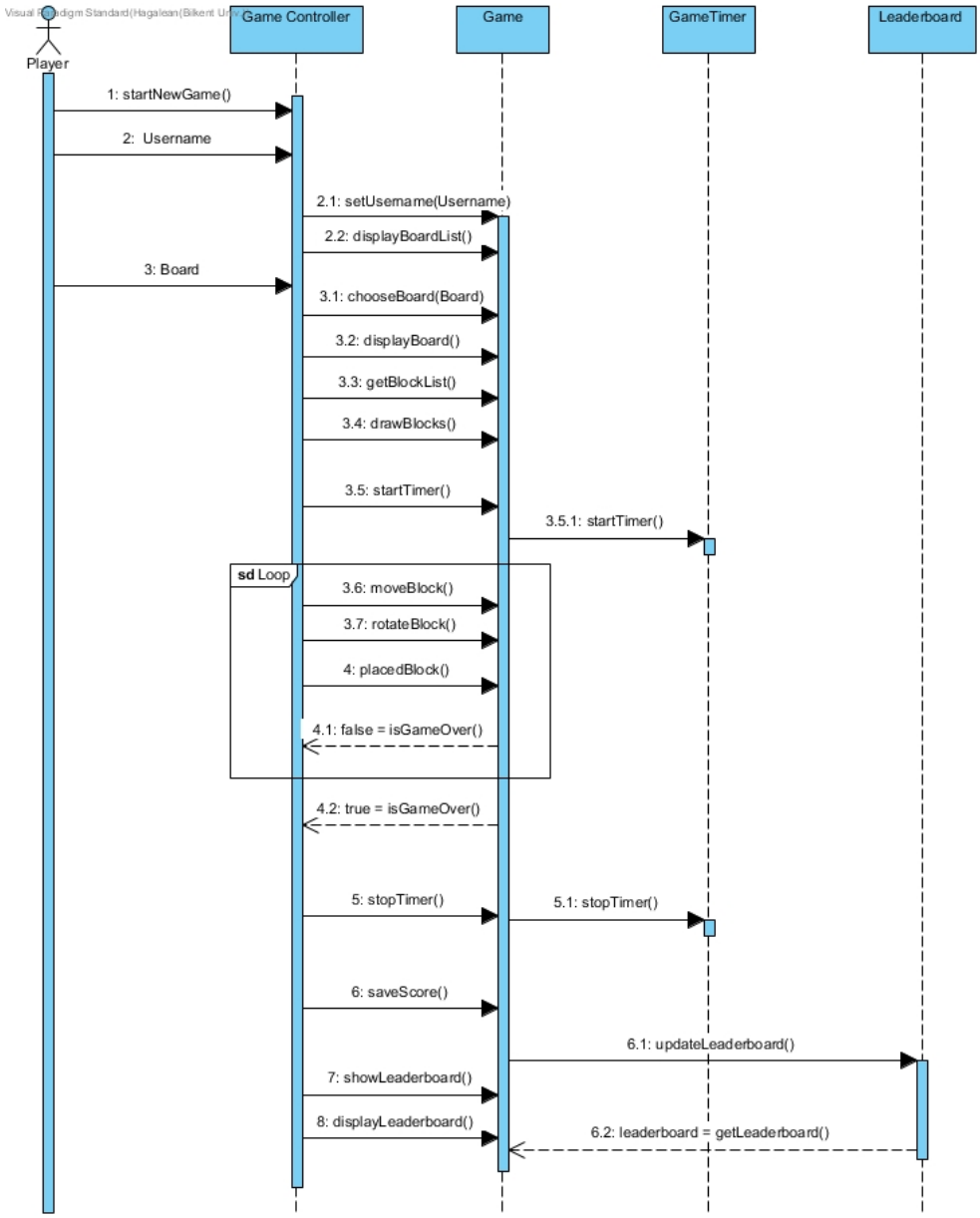


Figure 3: Sequence Diagram for Playing a level of Katamino HAK3

Scenario: User starts the game

The user wants to play the game so he/she presses the play game button on the main menu. Then the user is shown a list of different boards and then chooses one of these boards. The game class gets the block list of the chosen board and draws these blocks to the block pool. And then the timer is started using the GameTimer class. After this the user can move and rotate blocks as they wish until the game is over. When the game is over the user will see the leaderboard.

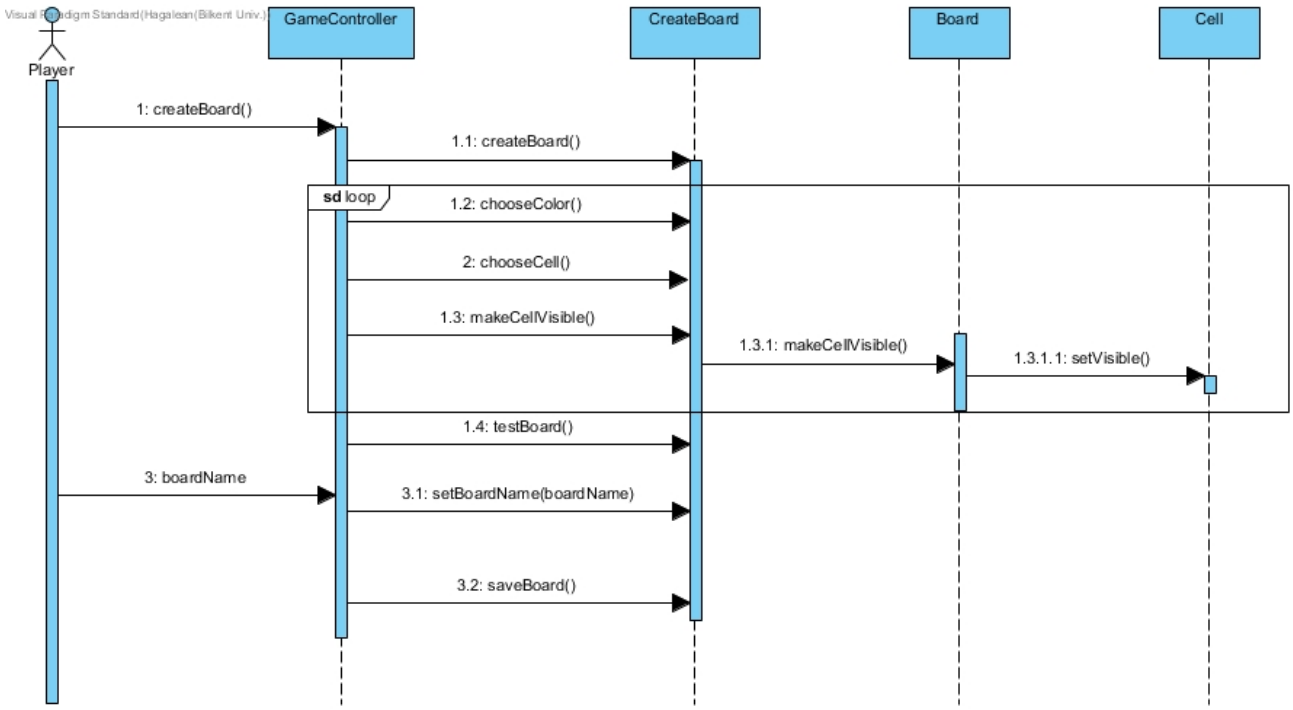


Figure 4: Sequence Diagram for Creating a Custom Board using BoardCreator

Scenario: User wants to create a custom board

The user wants to create a custom board so they press the create your board button on the main menu. Then the user is given an empty grid with game squares that they can fill and colour. After the user fills the squares that they desire they can test the board by using infinitely many blocks. After the testing is complete the user can either save the board or edit the board further.

5.2.3 State Chart Diagrams

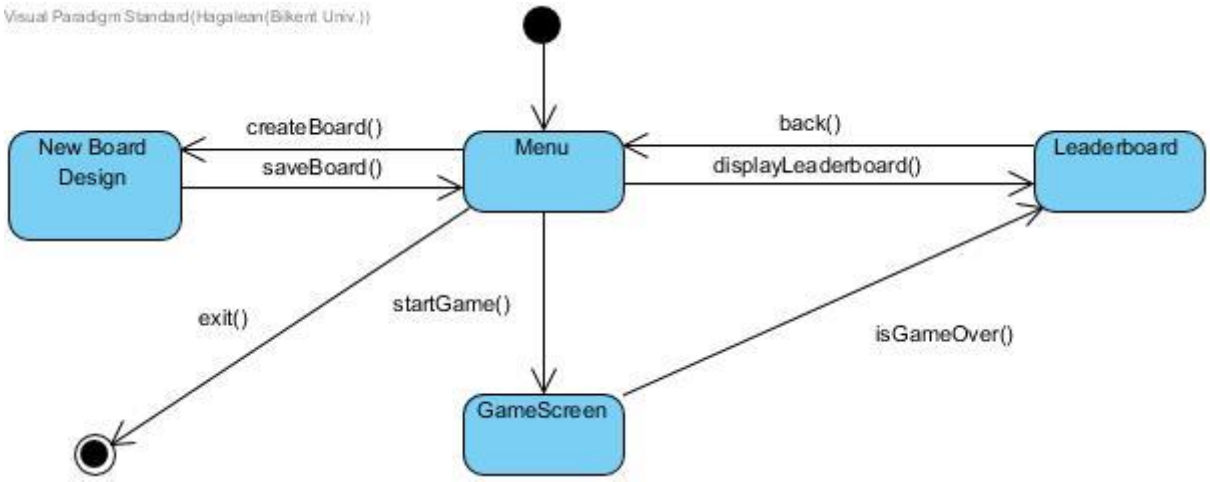


Figure 5: State-chart diagram for game

Initially the game is in the menu state. The program displays the menu and waits for user input. The state of the program changes depending on the user selection. If createBoard() function is called by the program, the program goes into the new board design state in which custom boards can be created. The program stays in this state until the saveBoard() function is called.When the saveBoard() function completes execution the program goes back into the Menu state.If user selection makes the program call the startGame() function, the state of the program becomes GameScreen. This is the state in which the game is played. The program stays in this state until the isGameOver() function returns True. When the the isGameOver function returns True the state of the program changes to Leaderboard. This state can also be reached from the Menu state when the displayLeaderboard() function is called. Leaderboard state is the state in which the high scores are displayed. When the back() function is called in the Leaderboard state, the program goes back to the Menu state.

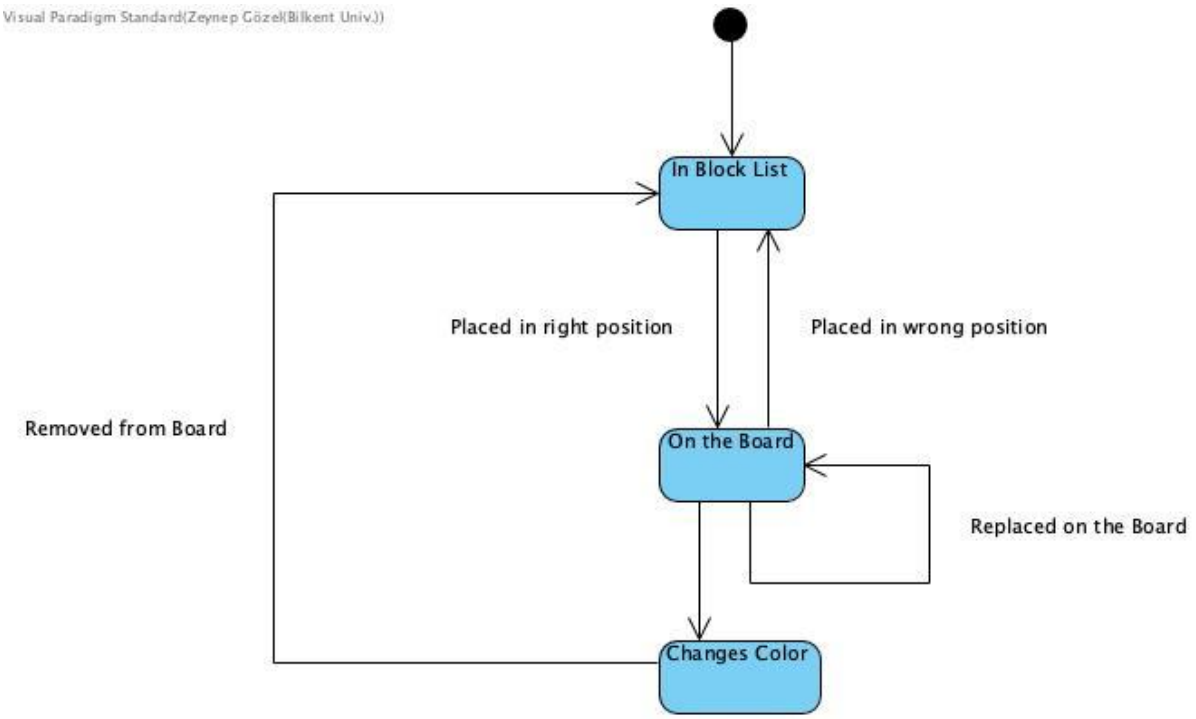
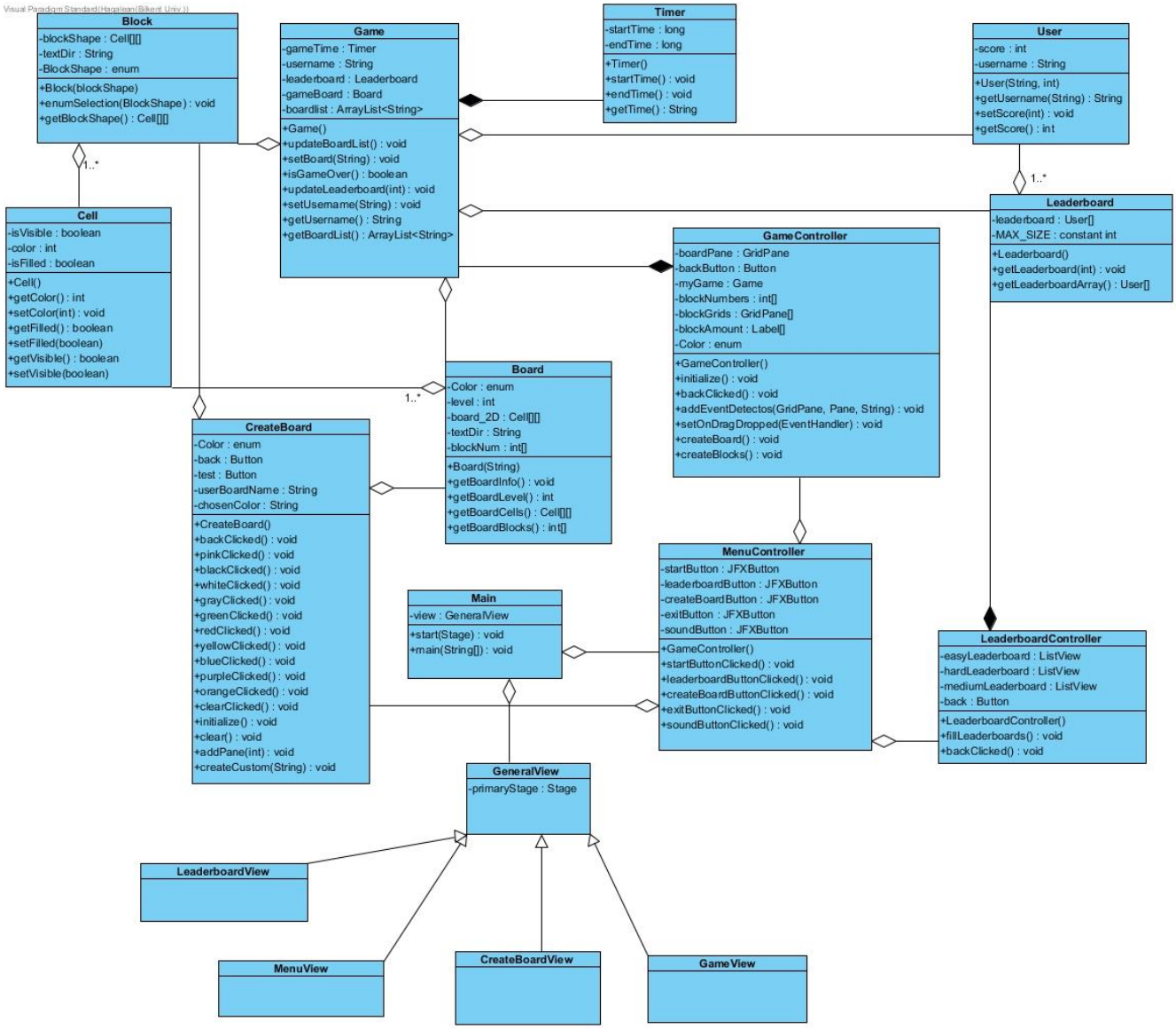


Figure 6: State-chart diagram for a block

When the program is in its GameScreen state, the Blocks that correspond to every level start in the "In Block List" state. This means that initially every block that will be used to solve the board is on the block list and no block is on the board. A block can change its state to "On The Board" if it has been placed in a valid position on the board (A valid position for a block means that every part of the block is placed in an empty and visible cell on the board) A block that is in "On The Board" state can change position on the board, therefore the "On The Board" state is a self looping state. A block that is in "On The Board" state can stay in that state until the level ends or can go back to its original state of In Block List if it is removed from the board.

5.3 Object and Class Model



Main Class

As shown above, the class model of Katamino has 17 classes in total. Main class is the class that allows the player either to start a new game, or create a board or show leaderboard at the beginning of the game. In addition, the player can also set the volume or exit the game.

MenuController

This is the class that contains the functional bits of the Menu. The buttons defined in the MenuView text file become functional and codable elements in this class. The actions of the buttons are declared in this class. This class also contains a GameController which allows the User to start a new game from the Menu.

MenuView

This is the class that contains the GUI information related to the Main Menu of the game. The information is stored in text file.

Game Class

Mainly, Game Class allows the user to play the game and checks whether the game is over or not. It starts the timer when the game begins and controls the moves and rotations of the locks.

GameController Class

GameController class is used to collect input from the user and manipulate the game object according to that. GameController gives a meaning to the GameView class and according to the actions of UI Elements on the view, by using listeners, modifies the contents of the board object of the specific game object.

GameView Class

This class consists of user interface elements for the specific game screen of Katamino HAK3. The user interface data is stored in a text file that interacts with

JAVAFX. This view majorly contains the grid of the game board and the blocks to be placed on the board.

Board Class

The Board class contains the shape of the board as a 2 dimensional cell array. It also has the block information of a board. Moreover, it contains which blocks the board is made up of.

Cell Class

Cell Class is used to make a whole board from cells. Each cell can be thought of as a pixel in a pixel art. The specific shape of ach board is obtained by setting certain part of a 20x20 square grid visible. This class is responsible for determining if a cell is filled or not filled, visible or not visible.

BoardCreator Class

BoardCreator class is used for controlling the creation of custom user boards. T he player fills the squares by choosing a colour and this class makes the specified squares coloured for the user.

CreateBoardView

This class contains the User Interface information of the Create Board screen of the game in text file. It contains a fillable 20x20 grid a colour picker and other UI elements.

Block Class

Block class has one to many associations with Board class, meaning that every board contains many blocks. We generate different shaped blocks by using this class and numerate these blocks to distinguish them (e.g. Block “1” refers to simple 1x1 block).

Timer Class

Timer class is used to control the time of the game. The timer starts when the game screen appears and ends when the player finishes the game successfully. We use getTime() function to determine the point and ranking of the player.

Leaderboard Class

The leaderboard class reads the leaderboard from a text file found in game folders and copies the leaderboard into a user array called leaderboard. This process is done using the getLeaderboard method. After the start-up of the game every change to the leaderboard is done on the user array. Before the termination of the program the latest leaderboard is copied from the user array to the text file using the updateLeaderboard function. Every time the user wants to see the leaderboard, the displayLeaderboard function of this class iscalled.

LeaderboardController

This is the class responsible for governing the actions of the UI elements in the LeaderboardView class. This class is responsible for filling the leaderboard as well as detecting back button clicks and taking the user back to the main menu if a back button click is detected.

LeaderBoardView

This is the class that contains the GUI information related to the Leaderboard screen of the game. The information is stored in text file.

5.4 User interface - navigational paths and screen mock-ups

Some screens shown in this section are previous mock-up pages since they are not implemented yet.

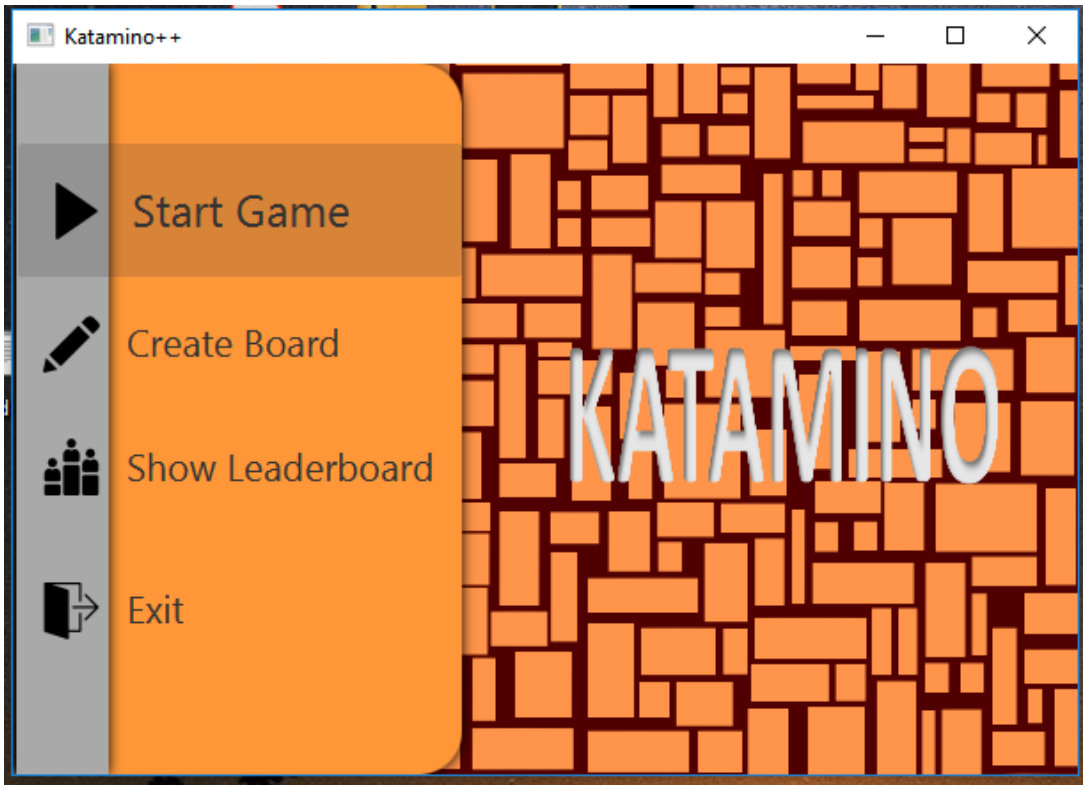


Figure 7: First screen of Katamino

Figure 7 shows the first menu screen of Katamino. There are four buttons for the user to go on. First: "Start Game", second: "Create Board", third:"Show Leaderboard", four: "Exit". The user can set the sound with a sound set button. (This will be implemented, not shown in figure.) To quit the game, the user clicks on "Exit".

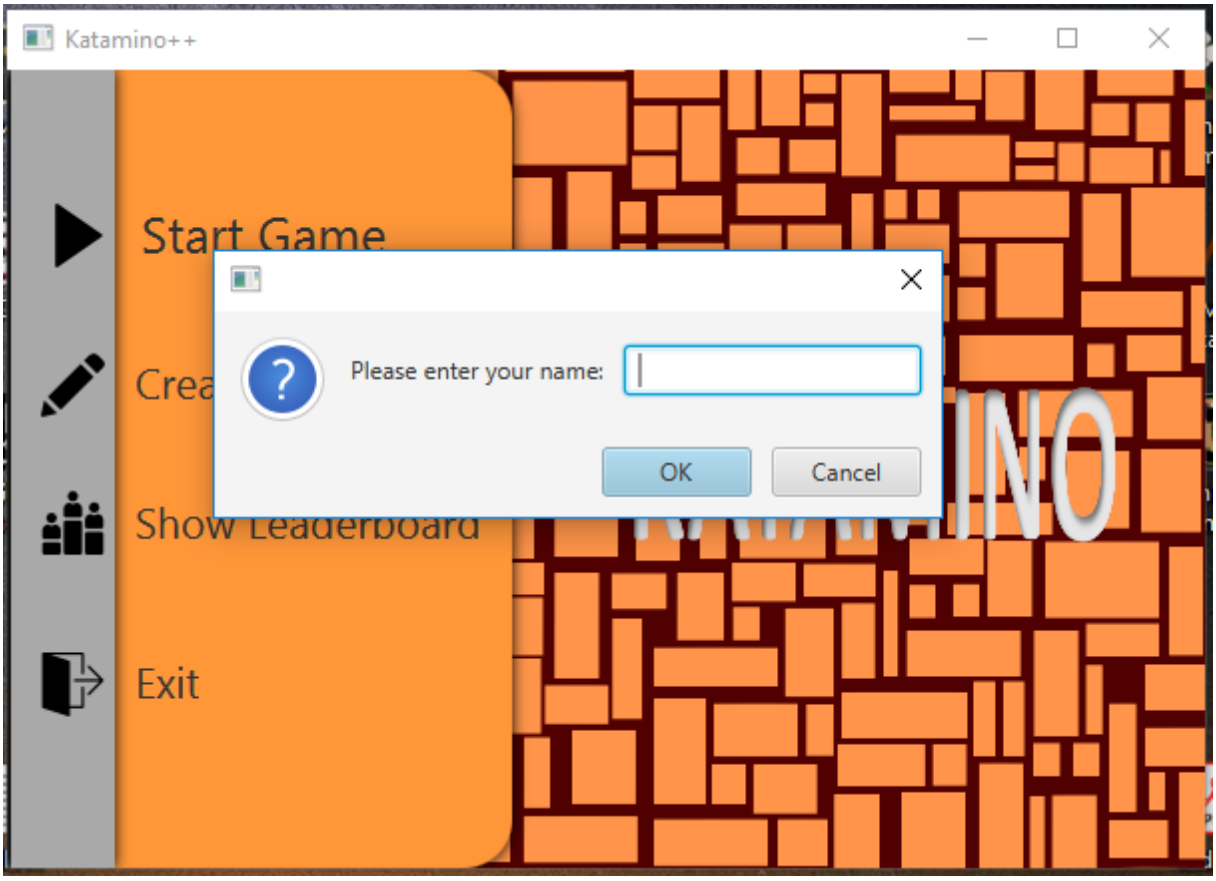


Figure 8: Name request

When the user presses "Play Katamino" button in figure 1, a text area pops up for the user to enter his/her name. The user is free to enter any name into the text area. (For example if there is already a user with username "Ege" in the leaderboard Ege can enter his username as Ege when he plays the game again.)

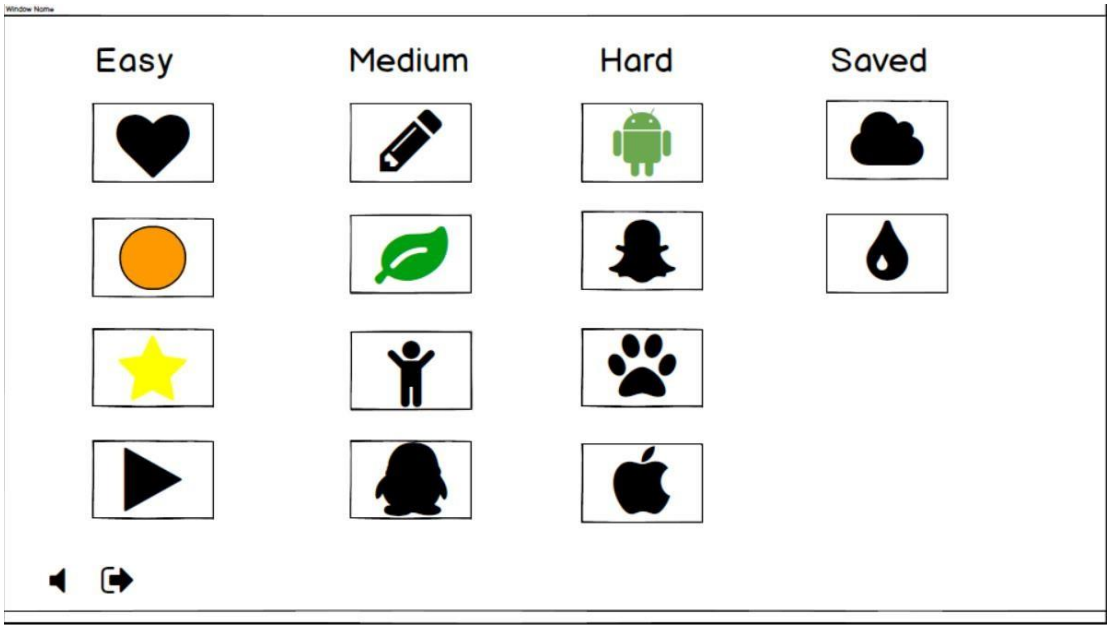


Figure 9: Level screen of Katamino

Figure 9 shows the level selection screen of Katamino. This screen appears after the user enters his/her name in the main menu. There are 4 columns of boards. Every column represents a difficulty level and every box under these columns gives a preview of the level. For example, if the user selects the first box in the easy column, they will be taken to the Heart level in which the user will be asked to fill a heart shaped board with given blocks. The last column contains custom levels that are created by the user.

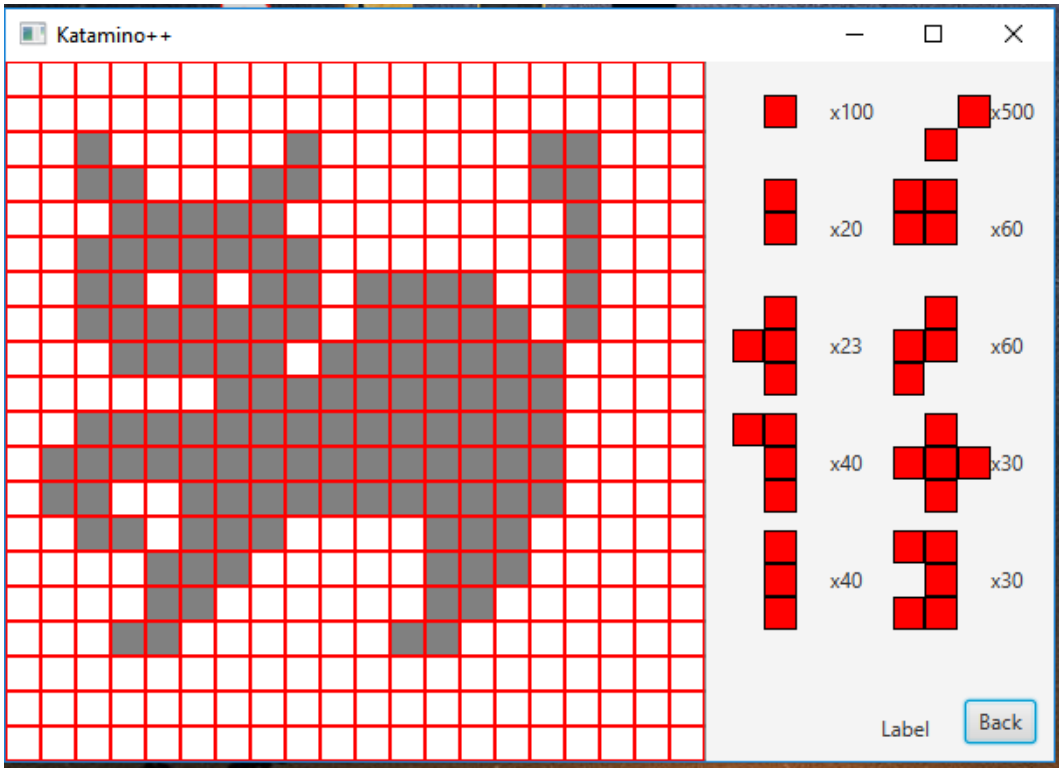


Figure 10: Beginning of the game

Figure 10 shows the very beginning of the Katamino game, after a level is chosen from the choices shown in figure 9. The timer shown in upper left begins instantly when this screen appears, meaning the game is started. At the beginning of the game move count shown in upper right is zero. Each time the user places a block to the board, it increases by one. The fair grey part of the board is in a special shape and only that part can be filled in using the blocks from the "Blocks" section in right. The bulb shown in bottom right is the hint button. The bulb, move counter, and time will be implemented.

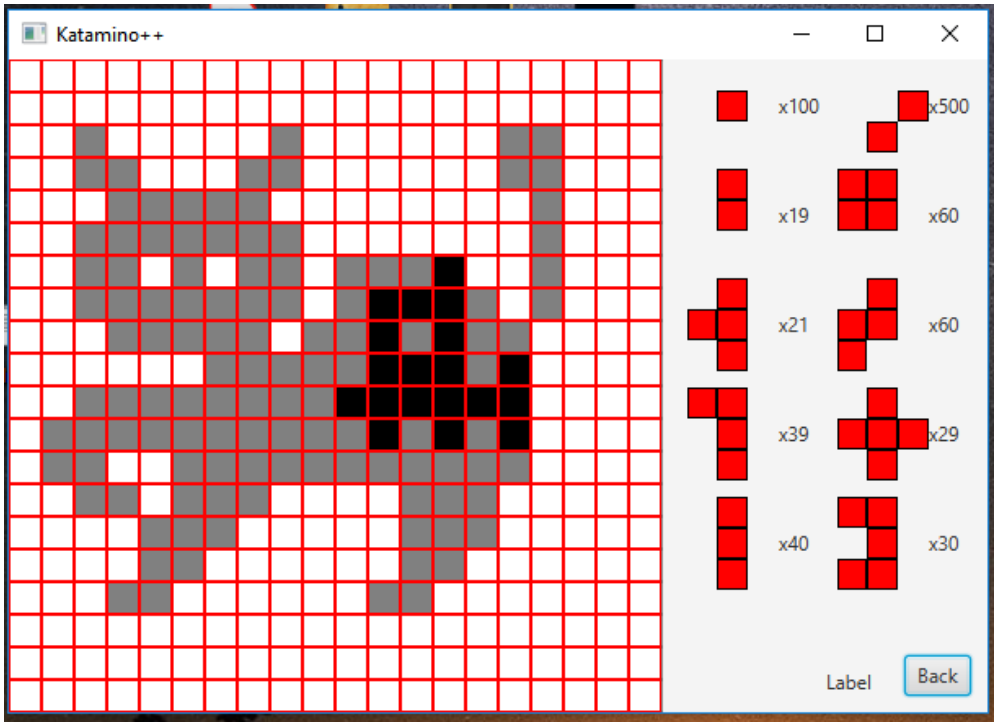


Figure 11: Playing the game

Figure 11 shows the ongoing game. The fair grey board gets coloured when the user places a block on it.

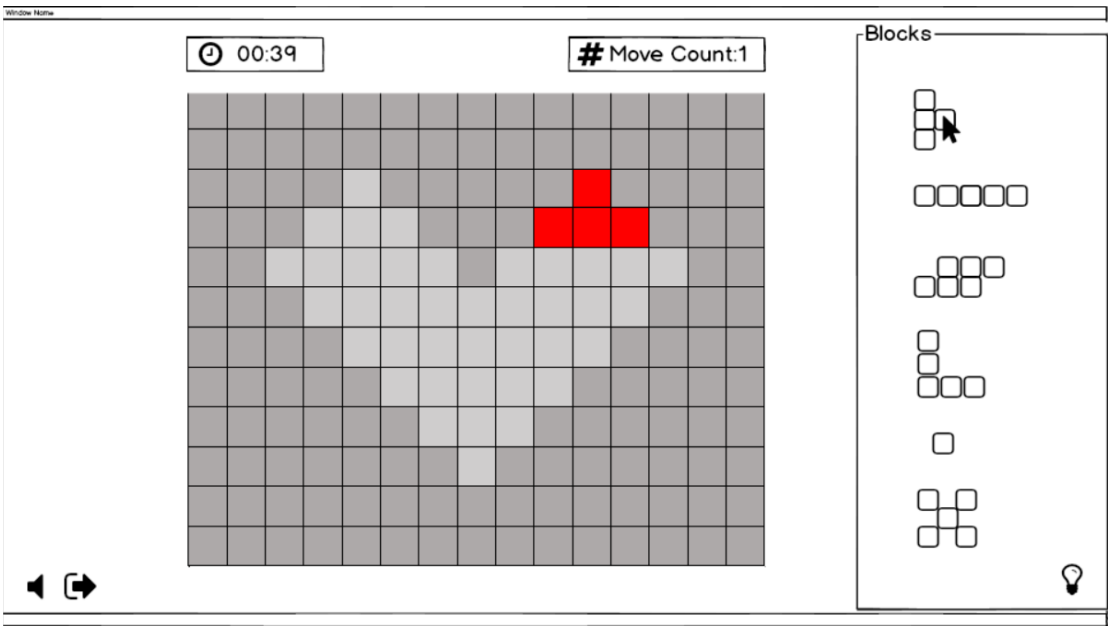


Figure 12: Rotating a block

Figure 12 shows a block rotated clockwise for 90° after the user has firstly left clicked on it to choose it and second time right clicked to rotate it.

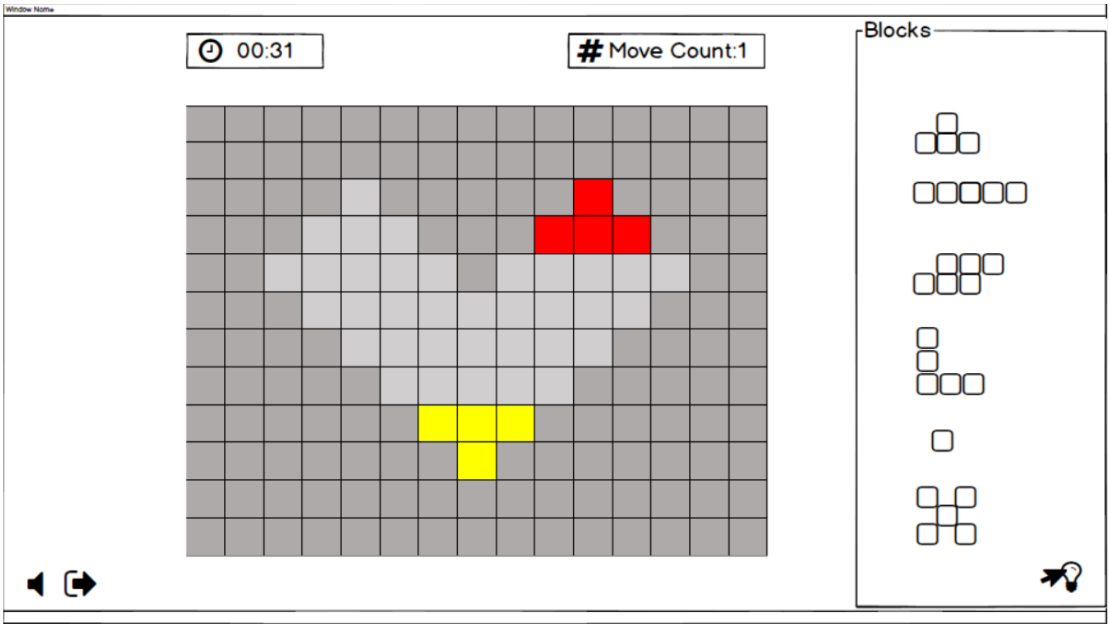


Figure 13: Highlighted hint

Figure 13 shows a part of the board highlighted as a hint after the user clicks on the bulb icon.

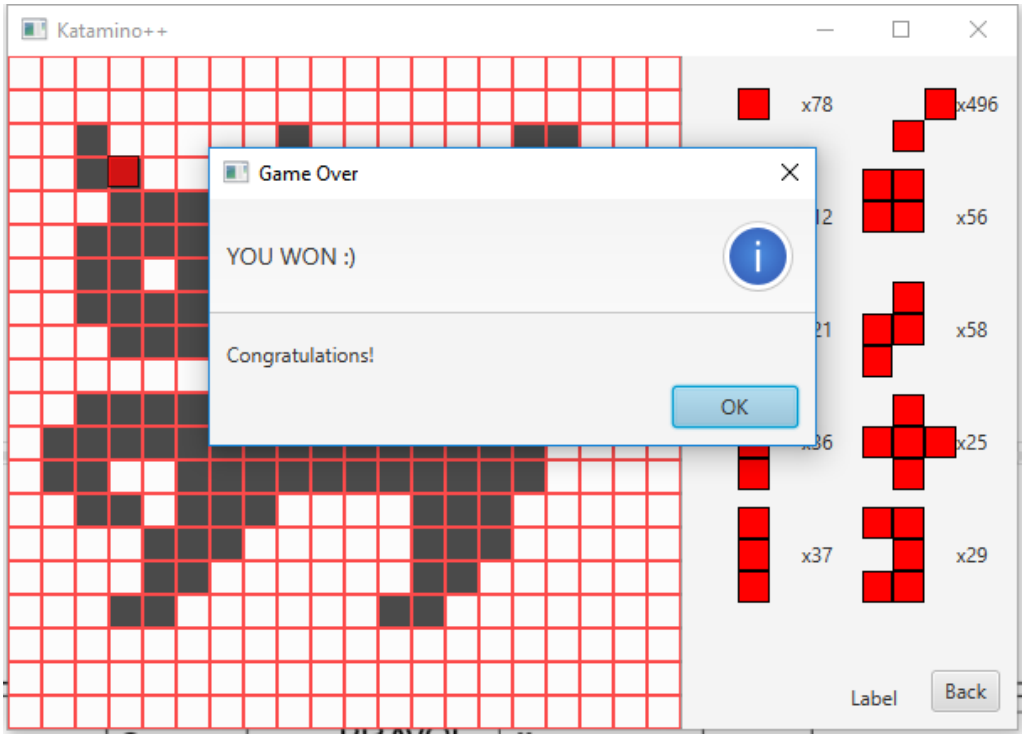


Figure 14: Completed game

Figure 14 shows the completed Katamino Game. The fair grey part of the board gets completely coloured when the blocks are placed correctly and fill all the board. The timer shows the time passed until the beginning of the game. Move count shows the total move counts. When the user completes the game, an alert box appears as shown.

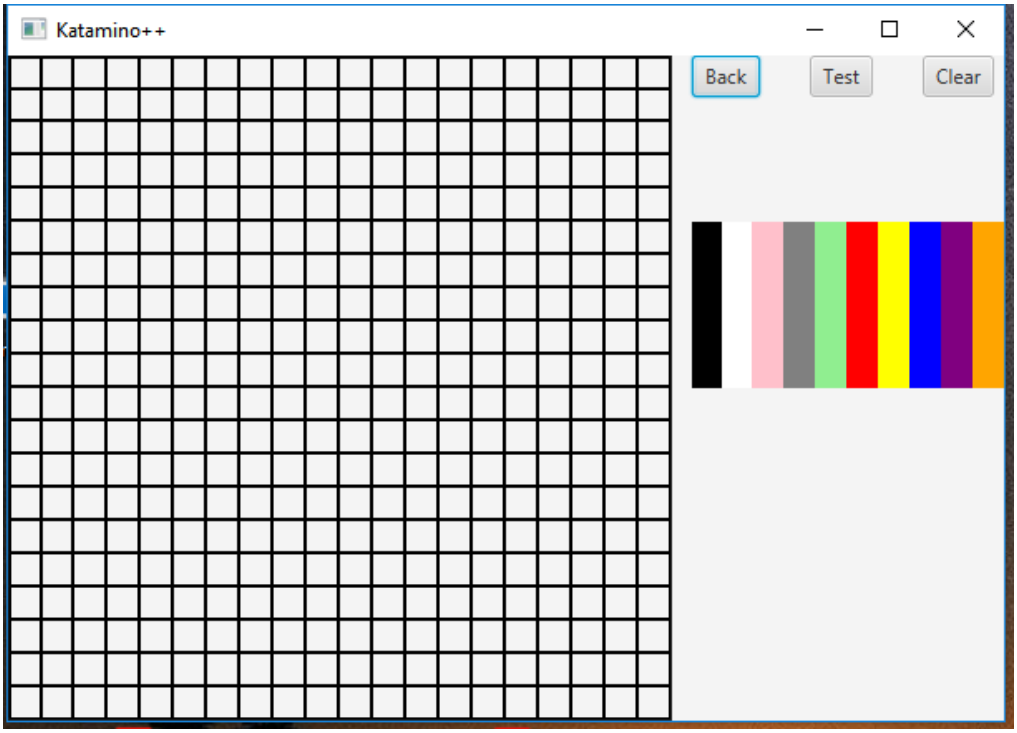


Figure 15: Player's Own Board Design

The board design screen shown in figure 15 appears when the user presses "Create Board" button on the first screen shown in figure 7. The user chooses a colour from the right and draws his/her own board by clicking the grids one by one. The user clicks "Test" button in bottom right to test and save the board. To start from scratch, the user clicks on "clear" button.

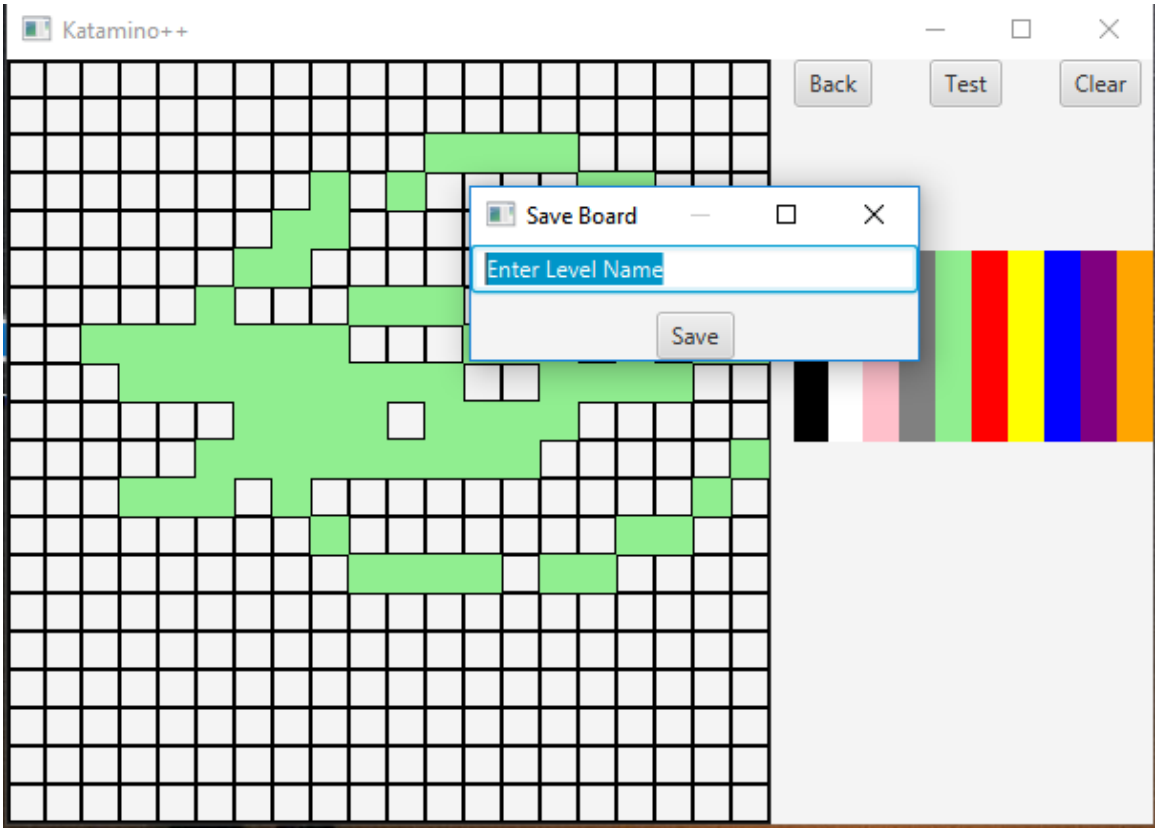


Figure 16: Entering board's name

Screen in figure 16 appears after the user creates his/her own board and clicks on the "test" button. If the board passes test, meaning that it is fillable, a text field pops up for the user to enter the board's name.

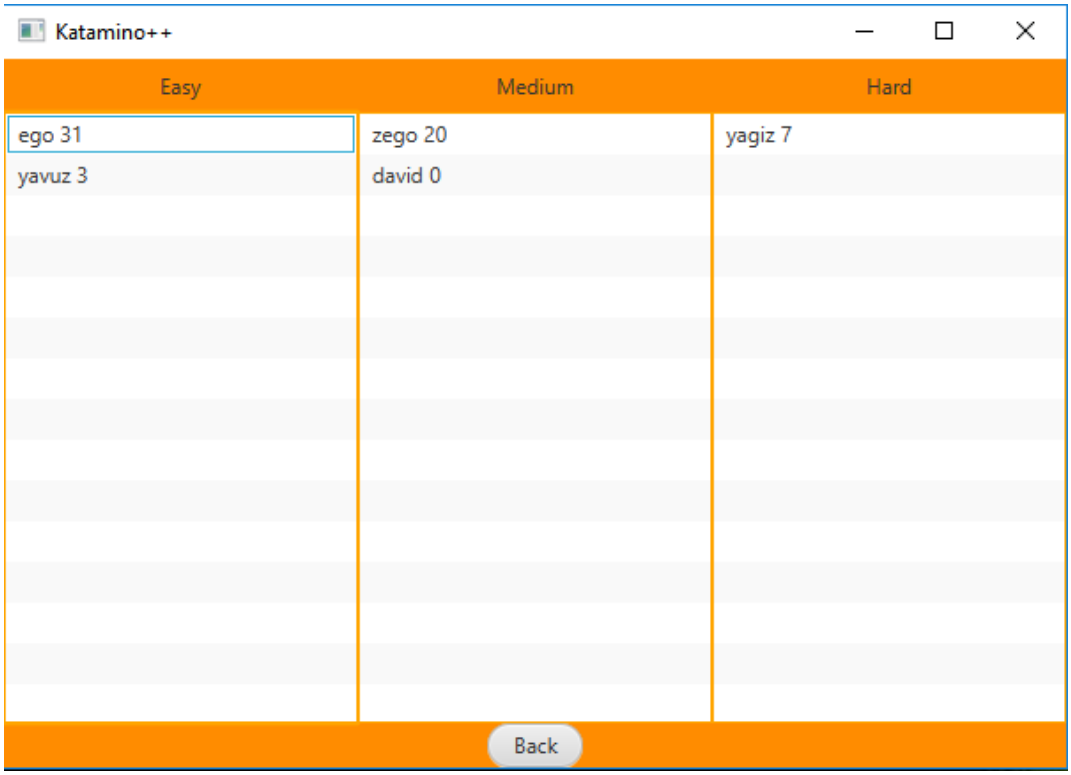


Figure 17: Leaderboard

Leaderboard screen shown in figure 17 appears when the user presses the "Show Leaderboards" button from the first menu screen in figure 1, or appears automatically after the game ends. This leaderboard is offline and shows the leaderboards of different levels separately.

6 Improvement summary

We updated all of our diagrams based on feedback and based on the experiences we gained during the initial coding process. We fixed logical errors in our use case diagram and also simplified the aesthetics of it to make it more readable. We updated our sequence diagrams so that they show each step of a sequence in a clear way. The biggest change we did was to the Object Class diagram: during the coding phase we saw many errors and shortcomings in the program architecture we foresaw, therefore we had to implement many changes to this class structures and relationships. As a result, our Object Class diagram went through great changes. We also added some new diagrams to make things easier to understand. We added

the activity diagram for the whole program which provides an easy to follow map of the game. We also added state chart diagrams for the important parts of our game so that important details are not left in the dark.

7 Glossary & References

1. "Katamino." Game Rules, www.thegamerules.com/en/board-games/family/katamino-907-detail.
2. Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13- 047110-0.