

**School of Engineering**

**Electrical and Computer Engineering Department**

**Design and Development of a Multi-Player Minesweeper Game**

*A project submitted*

*in partial fulfillment of the requirements for the EEE210 course in SEng*

**by**

Asset Baisalov (Student ID 201437287)

Darkhan Baizhan (Student ID 201525145)

**Evaluated by**

Dr. Aresh Dadlani

**Spring 2019**

**ACKNOWLEDGEMENT**

Our team would like to express great appreciation to Dr Aresh Dadlani for teaching students the foundation of Java programming, also for improving our work ethics and working style at Java environment. Also we want to thank our TA for fast response to the questions about course. Finally, we want to thank our parents for their constant support and guidance, as well as for caring for us.

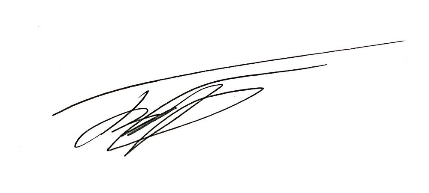
**UNDERTAKING**

This is to declare that the project entitled “Design and Development of a Multi-Player Minesweeper Game” is an original work done by undersigned, in partial fulfillment of the requirements for the course entitled “EEE 210: Software Engineering” at EEE Department, School of Engineering, Nazarbayev University.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

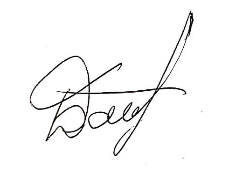
Student 1

Asset Baisalov (NU student ID 201437287)



Student 2

Darkhan Baizhan (NU student ID 201525145)



**ABSTRACT**

The project challenges students to find way to take information about relatively new for them information and implement it to the project. The project contains the POP and OOP programming at the same time, also there was graphical representation of information, taking the input from IO devices, matching the information, work with it and giving the result. Creating the chained programs where each level affects the next one shows work ethic of software engineers. Also the dividing the responsibility creates the software engineers from the students. There was used the functions from the java libraries, also there were created the own algorithms to achieve the goal.

Table of Contents

LIST OF FIGURES

1.   Introduction

Content

*2.1*     *Level A*

*2.2*     *Level B*

*2.3*     *Level C*

*2.4*     *Level D*

3.  Conclusion

Major Contributions

References

# LIST OF FIGURES

Figure  ‎2.2 UML Class Diagram for Level B.................................................... 9

Figure  ‎2.3 UML Class Diagram for Level C.................................................... 11

Figure  ‎2.4 UML Class Diagram for Level D.................................................... 12

**Introduction**

The “MineSweeper”  is once an iconic game, which was popular on Windows machines in early 2000s. It is well known game which has dozens of clones created by programmers all around the globe. However, the multiplayer mode in a “MineSweeper” is a new challenge and perhaps the breath of fresh air to the game which used to be on almost every computer on the Earth. During the process of creation of MineSweeper, the theoretical background gained in lectures are checked and implemented. This is the main motivation and significance of the project.

The principles of the original game are simple, the N x N board with the boxes which either has or has not the mine is initialized. Then the player is given the flags which are equal to the  number of mines present on board. In order to win the game player should flag all the mines or open all empty boxes.

The problems faced during the creation project are related mostly with the graphical implementation of the code, some concerns are related with the implementation of flags and recursive opening of the nearby empty boxes.

The successful execution of the principles of the game and the playable graphics are expected. The main priority during the creation of game was for user to have full gaming experience with all the basic functions working correctly, however not the design part.

# Content

The project description was given in four different levels, so the game development itself was divided into four parts as well.

**Level A**

In A-level were  created the purifying method, which could be called by recursion, the main aim of the method is “open” the neighbor cells, which are not bombs. The borders for this method are the numerical values, which contain the information about the number of bombs at neighborhood. To avoid the exceptions, which could be caused by uncontrolled moving through the cells, there were written additional rules for border lines. Also, there was a problem with infinite moving between two points, therefore, there were added additional “if” rule:  if (arr\_h[i][j] == '\_' && arr\_d[i][j] != '\_'), which stops the program, if previous cell was revealed.

**Opportunity to choose the level:** there opportunity to choose the level of the game, there three levels: 1 (15% chance that at the cell bomb will be planted), 2(20%),  3(30%), therefore, the user could regulate the intensity of the game.

The main part of the program is in the do while loop, it was made for opportunity to start the new game without exiting the console. Also there is choice between starting the new game or not. Also to avoid the problem with the case, there was used .equalsIgnoreCase.

There two layers, first is layer which showed to the user (boxed) and the second contains the information for the program with the info about empty places, bomb location and the number of bombs at neighbour cells.

At the console the user could see the progress of layer changing and the number of mines.

A the process of coordinates intaking, there could problem with mismatching of the input type, there was used the try catch block. After the taking the coordinates the *purifying* method is called which check is this empty cell or not. In case of empty space, it starts to reveal the neighbour cells.

The winning condition: open all cells which are not bombs.

The losing condition: open the bomb.

**Level B**

**Main functions:** the 3 classes Position, Grid and MineSweeperMain was created in order to make the algorithmic design implementation easier. During the creation of B-level the following logic was used: on Grid the are 9 x 9 positions so using the Aggregation relationship between classes, the above logic was executed. In Position class in order to describe the several  (more than two) states of Position (e.g. Explosion,Nothing,Mine,AllExplosion) the Enumerations was used ( Position class aggregates PositionsState enum) So, abovementioned object-oriented paradigms served to make code more readable and easier to implement further functions and interact them with each other. For the graphical part of gaming board the Grid class  which is responsible for the creating thed board and placing the mines on it extended the JFrame, being subclass of the JFrame allowed Grid to use its methods to create the board, set the size for it. Moreover, it allowed to add the MouseMotionListener and MouseListener which made the program sensitive to mouse motion and mouse clicks. In the designing interface part, since in our program each box did not represented separate button, everything was drawn on the Grid.

The box was nothing more than a drawing which was performed by “paintcomponent” overridden function which was inherited by BackGround class from the JPanel class.

Moreover, one of the main functions, besides the obvious two nested for loops in order to initialize array of Positions and check the surrounding Positions for the presence of mines, there are two functions which call each other recursively until they would find the Position surrounded with bombs. They are revealNear(int x, int y) and show(int x, int y), the former one uses the same nested for loops as the counter of neighbouring miner, however revealNear(int x, int y)  then calls show(int x, int y) method which opens the cells if it was not opened before

**Challenges:**

One of the main difficulties was caused because of above mentioned logic of the graphics used (without buttons using the drawings only) which caused the necessity in method which converts the pixel coordinates into the elements of array depending on the width of the each box and the spacing between them.

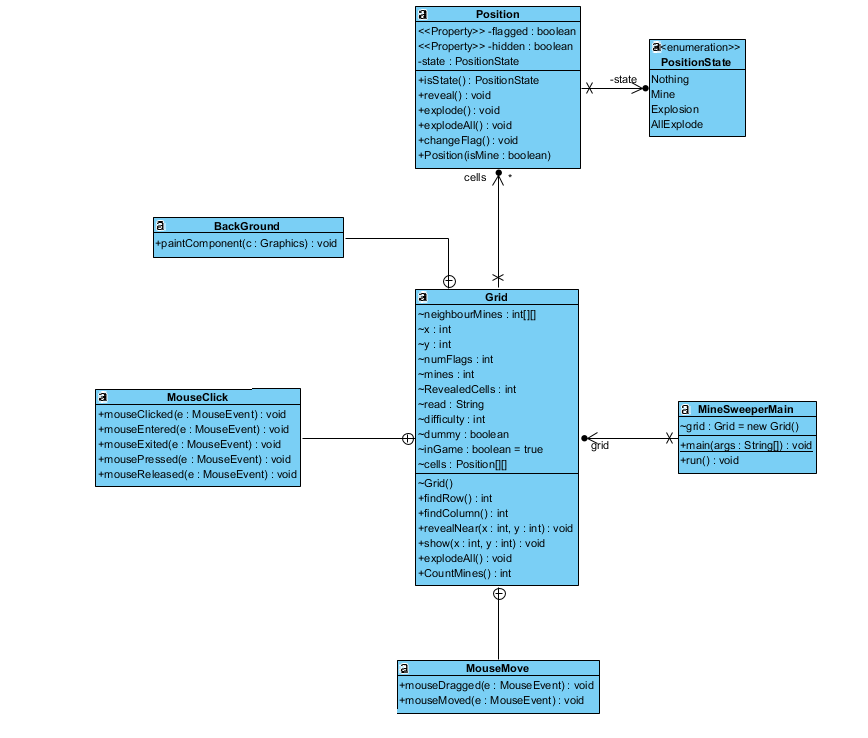


Figure 2.2. The Level B UML diagram

**Level C**

The main difference between Level B and Level C is implementing the multiplayer. The new functions allows two users play at the same desk and create a competitive atmosphere. The main changes were done at the Grid.java class, There were added data fields for storing the name of each player, number of flags for each player and the integer “clock” which allows to switch between the users. There were changed several parts of mouseclicking places and Jpanel, where some part of code was divided into the two flows, which allows track the winning and losing conditions for each of the player. Also there were added the additional fields to show the number of the remaining flags or scores.

There were problems with changing the order of players. Therefore, there were added boolean flags that allows to change the clock only if the previous player made an action, which affects the game process (the misclicks are not counted).

There two winning conditions: if the player clicks on the last safe cell or puts the flag on the last bomb. It could be the intensive game with bomb neutralization or passive game with trying to not have a deal with bombs.

Losing conditions - clicking on the bomb.

In B-level the 3 classes “Position”, “Grid” and “MineSweeperMain” was created in order to make the algorithmic design implementation easier. During the creation of B-level the following logic was used: on Grid the are 9 x 9 positions so using the Aggregation relationship between classes, the above logic was executed. In Position class in order to describe the several  (more than two)

“when there’s a red, green or blue wavy line, then something must be wrong.” Other proofreading pointers follow in the next subsections.

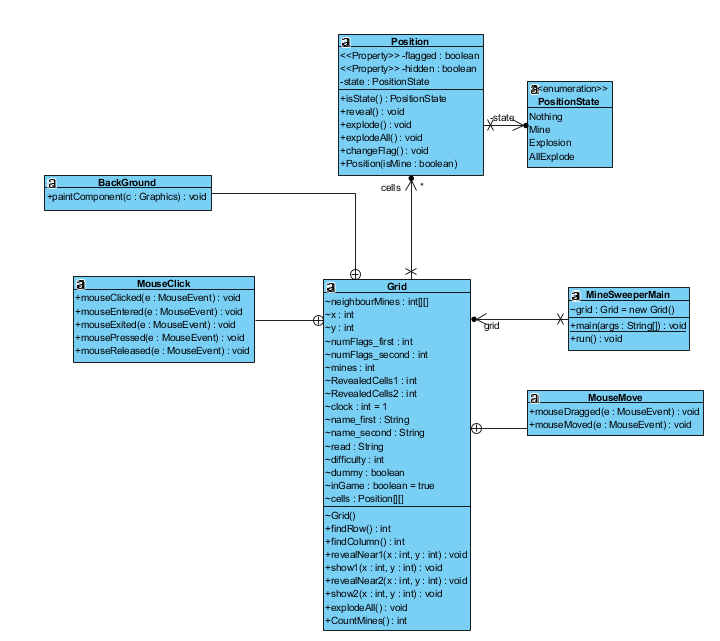


Figure 2.3 The Level C UML diagram

**Level D**

**Main functions:**Inheriting the winning and losing conditions from level-C  the program was changed in order to be able to have a board of various dimensions. The implementation in order to fulfil the given condition was straight forward. (To create the separate integer type variables row,column) which changed the 9 digit ( 9 x 9 dimension from C level) into the row and column in the nested loops throughout the code.

**Challenges:** The main challenge while implementing this Level was caused again by the use of drawing logic instead of button logic. Since the dimensions are dynamic and in order to have the more capacity for the dimensions up to (40 x 40), the dynamic width of square boxes were chosen. Their dynamic nature was implemented simply using the if statements for the specific number of rows(3 cases width = 30 when row < =22; width = 20, when row>22 and width = 15, when the row >=33) ( since, usually screens has less height than width). However, this solution caused a new challenge, which is the need of  dimensions and coordinate alignment of flags, digits and bombs for every new width. The solution for the new faced challenge was straightforward as well: to write the different if statements to check the 3 different cases of widths and align correspondingly ( bombs, flags and digits ( the number of neighbour positions containing mines)

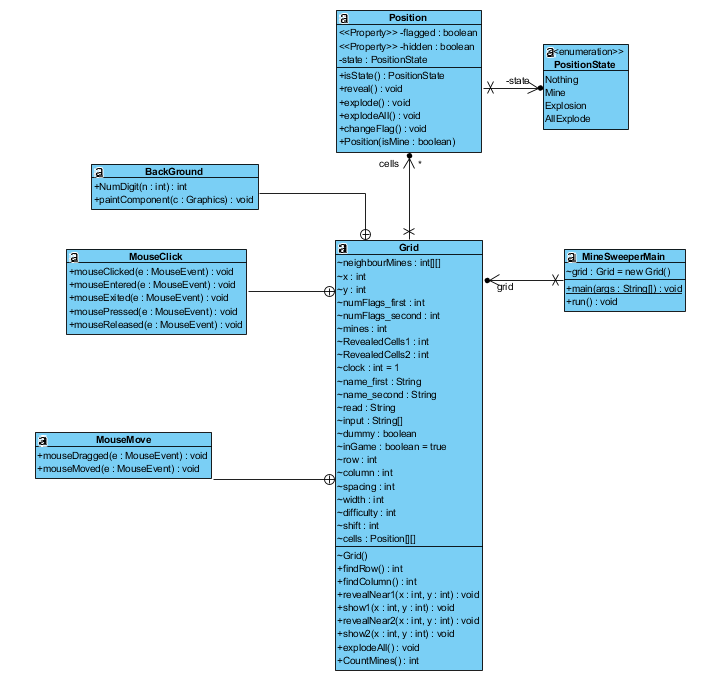


Figure 2.4 The level D UML diagram

## Abbreviations and Acronyms

UML -  Unified Modeling Language

# Major Contributions

The task division was based on the principle of equal load and challenging the students. Therefore, Asset Baisalov worked with the Level A and Level C, at the same time Darkhan Baizhan worked with level B and level D. The main principle was based on naming one of the student as the lead of the specific level. It means that of Asset Baisalov was lead at the level A, he was responsible for level architecture and implementation algorithm, i.e. the ways how to create methods and others. Also, there was used the strong sides of each student. Independently, who is lead at the current level, Darkhan Baizhan worked with try catch block, and Asset Baisalov majorly worked with the data arrays and how to sort them. Darkhan Baizhan worked with the graphic part and algorithm how to transform the coordinate into the row and column. At the same time Asset Baisalov mostly worked with information flow at the multiplayer part and with challenges about right timings. Therefore we think that the work division was fair due to division not only algorithms, but also responsibility for levels, because each level will affect on other one. In case of Darkhan, the fact that level D would not affect other levels neglected by fact that level B was the core of OOP part of the project. In case of Asset, the easy implementation process oriented program neglected by the fact of finding right algorithm to change the player and displaying right info,

**Conclusion**

During the project execution the basic knowledge of the Java was improved, also the using the IO devices as mouse and creating the additional windows was studied. The main problems of the students, who performed the project were based on the lack of the experience in work with Java Swing.

The main challenge was implementing the graphics, which would be usable for taking coordinates displaying the additional information as the name of the player, number of the flags and scores of players. There was found the combined way of solution, which includes the material from learning sources and the own ways of the students by using the background in OOP programming and Java.

Also, using the java libraries allowed the work with the input device, which known as mouse. The special algorithm developed by one of the students allowed to transform the coordinates from the desk into the numbers, which means the number of column and the number of row.

The project call to the creative way of thinking, and it revealed such ability at students.

**References**

[1] <https://www.javatpoint.com/java-swing> Java Swing:: tutorial

[2] <https://www.geeksforgeeks.org/enum-in-java/> ENUM in Java, Intro