

**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**

Alisher B Aitbayev (201657891)

Kalamkas A Zhagyparova (201520921)

**Evaluated by**

Aresh Dadlani

**Spring 2019**

**ACKNOWLEDGEMENT**

In the development of our project, several individuals have contributed in its successful completion by providing us with invaluable advice, great support and motivation.

First and foremost, we would like to show our gratitude to our Course Instructor, Aresh Dadlani, Assistant Professor at Nazarbayev University, for not only guiding us throughout the course, paying a lot of attention and time, but also, giving us an opportunity to work on this engaging project and encouraging to learn.

Our thanks and appreciations also go to our groupmates and friends, who gave us valuable suggestions regarding the project, and who have contributed towards the generation and fulfillment of ideas as well as the improvement of the quality of the project. We would like to express our special gratitude to our friend, 2-year Computer Science student at Nazarbayev University, Daulet Issatayev, who helped us to figure out and fix an unintendedly created bug in the code.

In addition, we want to thank the developers of the well-known online video game “Dota-2”, for the creation of one of the unique heroes, namely “Techies”, humorous and inventive geniuses of pyrotechnics, who were taken as a basis of the overall design and theme of the game.

WITH SINCERE THANKS,

Alisher Aitbayev

Kalamkas Zhagyparova

**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. Alisher Aitbayev

Student 2. Kalamkas Zhagyparova

**ABSTRACT**

*A classic minesweeper game is a single player video game, the objective of which is to clear all cells of the board without detonating mines hidden in some cells. The graphics of the original game include light grey colors, yellow smiley, red timer and flag counter, which now seems to be less pleasant and old-fashioned. This report offers a discussion of the work done in the design and development of the Multi-Player Minesweeper game with the contemporary view using the Java Platform. The project consists of the results obtained after completion of each programming level provided in the manual. The game was developed by applying the Java Swing* *that allows to create graphical user interface ( GUI ) components (such as buttons).*

Table of Contents

[*1.* Introduction 6](#_Toc6783375)

[2. Level A 6](#_Toc6783376)

[*2.1* *Main algorithm* 7](#_Toc6783377)

[*2.2* *Additional features* 9](#_Toc6783378)

[3. Level B 11](#_Toc6783379)

[*3.1* *Main algorithm* 12](#_Toc6783380)

[*3.2* *Additional features* 14](#_Toc6783381)

[4. Level C 17](#_Toc6783382)

[*4.1* *Main algorithm* 17](#_Toc6783383)

[*4.2* *Additional features* 18](#_Toc6783384)

[5. Level D 21](#_Toc6783385)

[*5.1* *Main algorithm* 21](#_Toc6783386)

[*5.2* *Additional features* 21](#_Toc6783387)

[6. Conclusion 24](#_Toc6783388)

[Major Contributions 24](#_Toc6783389)

[References 25](#_Toc6783390)

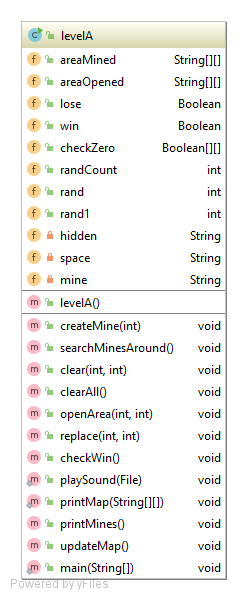
# Introduction

The aim of the project is to design and implement a simple two-player MineSweeper game in Java, which is a high-level object-oriented programming language. MineSweeper is a game that comes with Microsoft’s Windows Operating System, where the player has to detect the hidden land mines in the field. The field is a board game, which consists of initially covered squares that might contain either mine or the number of neighboring mines. Based on this numbers, the player should reveal all squares that are safe and do not contain mines. If the mine is revealed, the game is lost. There are several additional features like timer, smiley, flags and etc. implemented in the original MineSweeper game that were not left behind in our project too. So, in order to develop the game and achieve a pleasant, simple and effective program, four programming levels: A, B, C and D were implemented gradually. The level A required the most simple, non-OOP based program for a single player, which was improved and converted into graphical OOP based program in level B. Levels C and D implicated the advancement of multiplayer Minesweeper with arbitrary grid dimensions.

The graphical representation, overall layout and design was developed by taking “Techies”, heroes of online video game Dota-2, as a basis. Specifically, soundtracks, smiley, icons as well as images used for demonstration of bombs and flags were adopted in the theme of these goblins.

# Level A

## *Main algorithm*

Level A

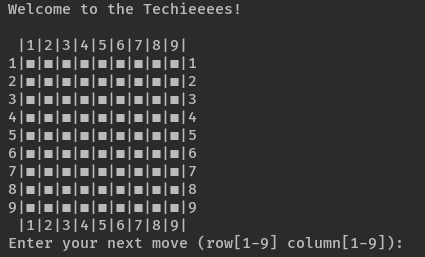
Algorithm

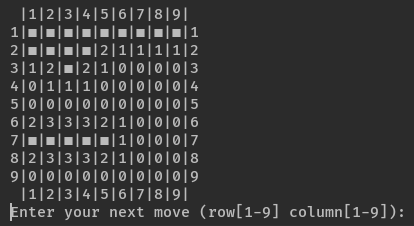
1. A constructor levelA() game is initialized with two-dimensional arrays areaMined[][] and areaOpened[][], which are empty
   1. areaMined[][] is responsible for mines
   2. areaOpened[][] is responsible for numbers and closed blocks
2. Inputs for the x and y coordinates are taken from the user
3. Method openArea() is called, which inserts space(“ ”) to areaOpened() in coordinates x and y or in other words, opens that block
   1. If block at coordinates areaMined[x][y] is opened already, it displays “This area is already opened!”
   2. If areaMined[x][y] is hidden(“■”), it inserts space(“ ”) to areaMined[x][y] and areaOpened[x][y]
   3. If areaMined[x][y] is mine("¤"), it makes Boolean lose=true
4. Method checkWin() is called, which counts the number of opened blocks and if it is zero, it makes Boolean win=true
5. Method clearAll() is called, which checks all blocks recursively
   1. If block is zero, method clear() for the given block is called
   2. Method clear() call method openArea() for all neighbor blocks
6. Method searchMines() around is called, which checks all the block
   1. If block is opened, it checks neighbors and write the number of mines found
7. Method updateArea() is called, which prints the board(areaOpened[][]) to the console
8. Program enters the while loop
9. The status of the game is checked
   1. If boolean win is true, game will display "Counter-Techies won!" and breaks the loop, which will lead to the closure of the game
   2. If Boolean lose is true, game will display "KABOOOOOM! You are bombed!" and breaks the loop, which will lead to the closure of the game
   3. If win and lose is false, it will repeat the steps starting from step 2

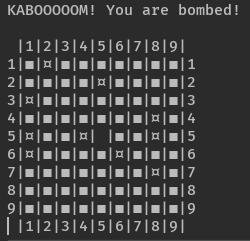
## *Additional features*

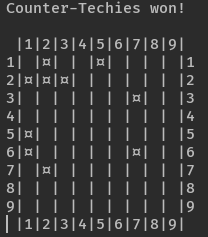
Additional features

* A numeration of rows and columns in order to ease the readability of the board and make it more attractive
* Method replace() which allows to not to lose on first move. The given method called only once after first move
* Playing sounds of the “techies”, which were taken from Dota 2 resources, at the beginning, after each move and when game is over



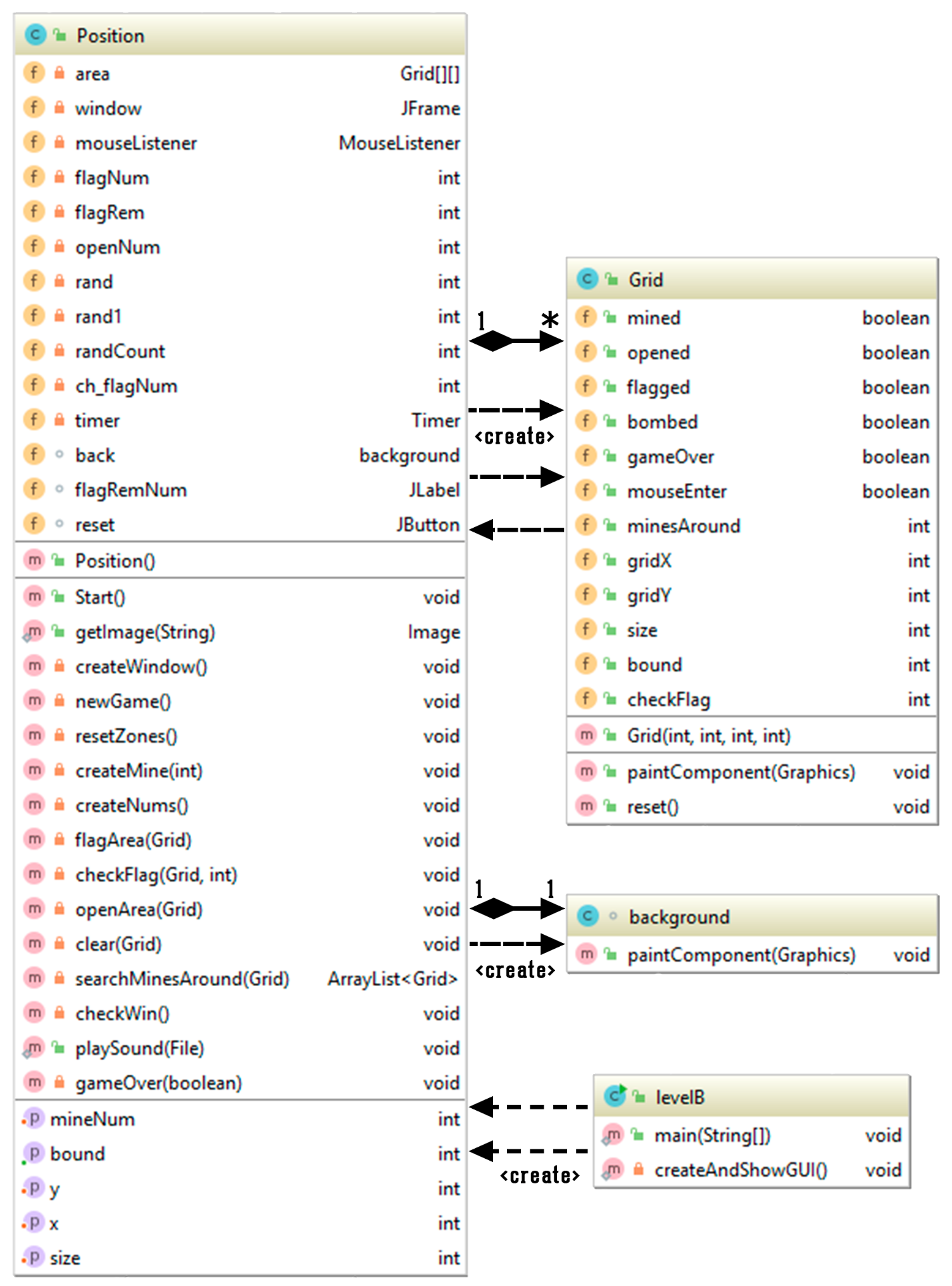






# Level B

## *Main algorithm*



In the given level, there is 3 classes: driver class levelB, Grid and Position.

1. A constructor Position() is initialized in levelB and other variables (x, y, mineNum, size)are set by the help of the setters.
   1. Constructor Position() initializes mouseListener. Different cases for mouse clicked, pressed, released, entered, excited and for right and left click are taken into consideration (will be further explained in Features section)
2. Method Start() is initialized, which initializes the methods createWindow() and newGame() from Position class.
3. In createWindow() method JFrame window is initialized and reset button, timer, background, remaining flag number are added (will be further explained in Features section). Moreover, a constructor Grid() is initialized and added.
4. As the the contructor initialized, the variables (bound, size, gridX, gridY) are also set. Moreover, class Grid has different Boolean variable, that will be set later on. Names of the variables are self-explanatory.
   1. If block is not opened and flagged
      1. mouse enters the block, picture “mouse” is drawn
      2. otherwise picture “closed” is drawn
   2. If block is flagged, image “flagged” is drawn.
      1. If game ends and block is not mined, “nonflagged” is drawn, which means that you mistakenly placed a flag
   3. If block is mined and opened
      1. If block is bombed, “mined” is drawn
      2. Otherwise, “mine” is drawn
      3. If flagged, “nomine” is drawn
   4. If block is opened and not mined, the number of mines around is drawn
5. Method newGame() initializes methods resetZones(), createMines() and createNums
   1. resetZones() resets all variables of the Grid and repaints them
   2. createMine() randomly creates mines on the Grid
   3. createNums checks the mines around every block and store the number in the Grid
      1. If the number is zero, method clear() is called, which clears the neighbors of the given block
6. If block is not flagged and the block was clicked by left mouse button, method openArea() is called
   1. If block is mined, the game will be end and player will lose
   2. If block is not flagged and not opened, the block is made to be opened and repainted. Moreover, if the minesAround is zero, method clear() is called
7. If block is clicked by right mouse button, flagArea() is called, which places a flag if it is not flagged and removes flag if flagged.
8. After each click the method checkWin() is called to check the win status. If number of opened block is equal to total block number – mine number, it will call gameOver(true)
   1. Method gameOver checks the win status and will show it. Moreover, it will display all the blocks of the window

## *Additional features*

* Playing sounds of the “techies”, which were taken from Dota 2 resources, at the beginning, after each move and when game is over
* Game has reset button, which allows to restart the game by calling newGame() method
  + If player loses, the special gif animation(bombed) is set on reset button icon
  + If player wins, icon(brutal) of the reset button is changed
* Timer is added, which starts after first left click, stops when game is over and resets as new game starts
* The flag counter is added, which shows the remaining flag numbers. Resets as new game starts
* Different scenarios for mouse interaction
  + If reset button is pressed, its icon(happy) is changed
  + If block is pressed, the icon(happy) of the reset button is changed
  + If mouse enters the area of the block, the color of the given block is changed and the color returned when the mouse is exited
* Extra condition to win
  + If all the mines are flagged and the number of flags is equal to number of mines, player will win
* Player cannot lose on the first move
* Graphical representation(flags, mines, icons, gif animations) of the “techies” were taken from Dota 2 resources
* Game show the bombed mine with red background



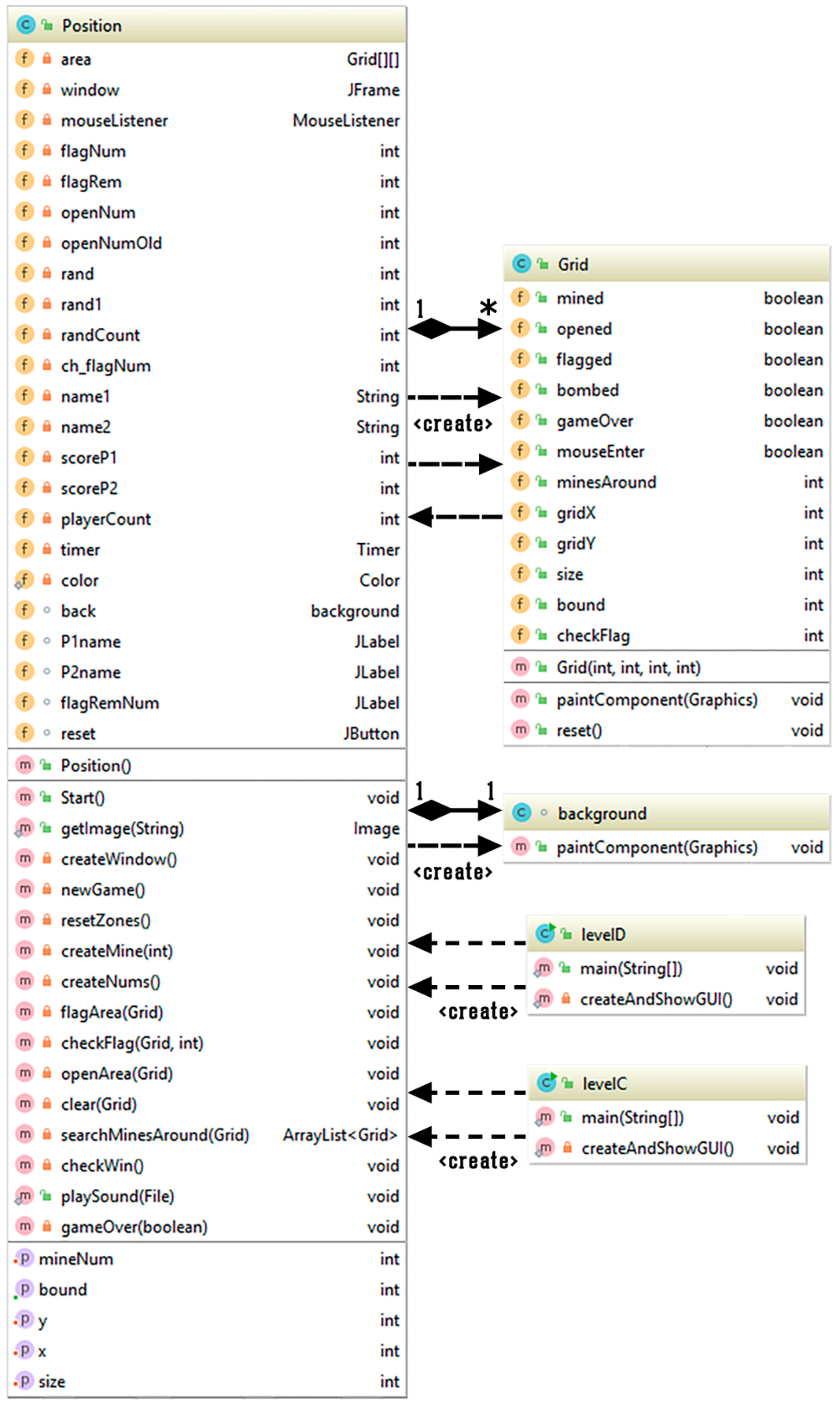






# Level C

## *Main algorithm*



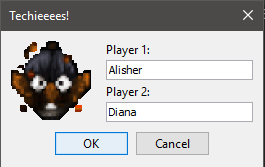
The code from LevelB was improved in LevelC and LevelD, and some new methods were added.

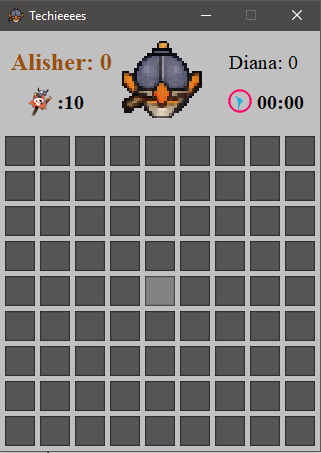
1. JOptionPane is added, which takes the names of the players as an input
2. The counter that choose the players turn
3. The score for each player is added
   1. Score equals the number of opened blocks by the player.
4. Name and score of the players are displayed on the window
5. The game is ended when
   1. One of the players opens mined block and that player will lose
   2. All the blocks are opened
      1. If one of the players will have higher score in comparison with another
      2. If they have the same score and both players will win

## *Additional features*

Additional features

* Both players cannot lose on their first move
* The name of the player on the window will change (color, size and Bold) on his/her turn
* If player will not enter the name, the game will not start











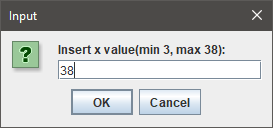
# Level D

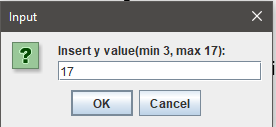
## *Main algorithm*

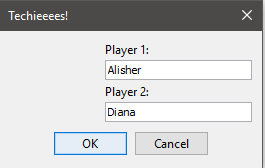
1. JOptionPane is added, which takes the grid size

## *Additional features*

* If player will not enter appropriate grid size, the game will not start

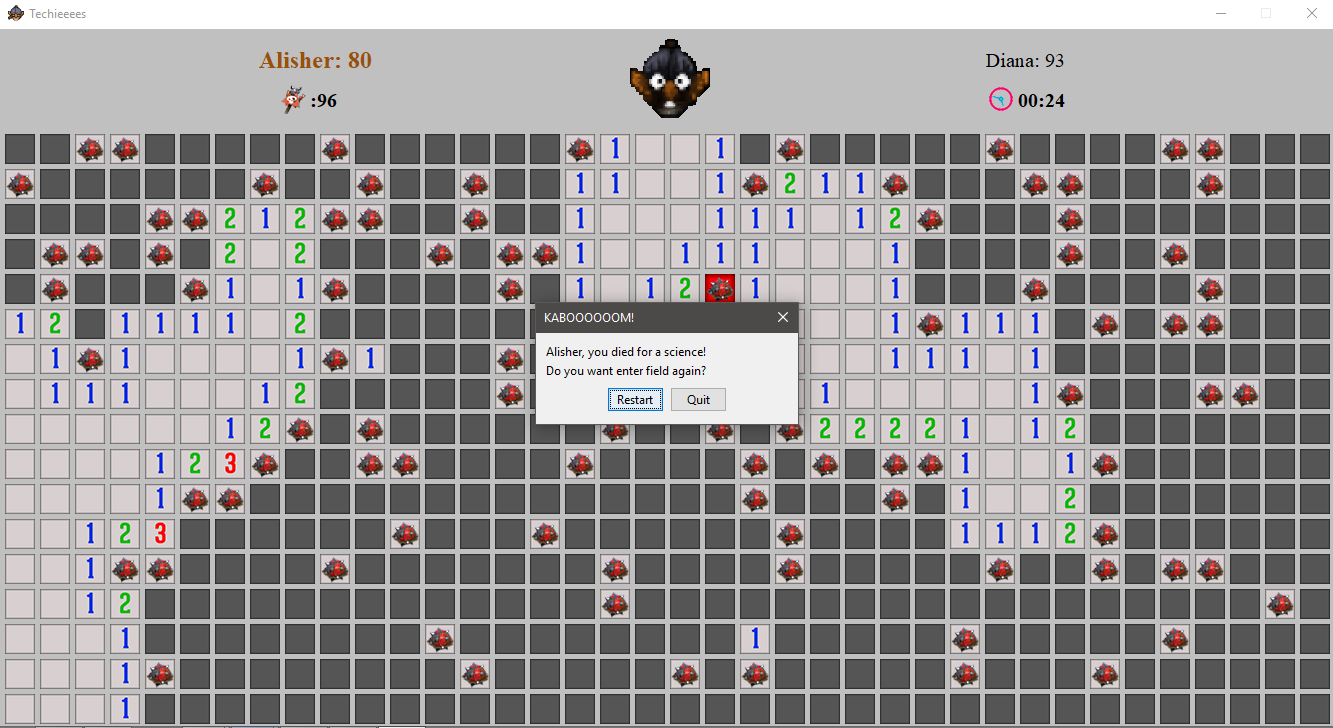












# Conclusion

The project succeeded in designing and implementing the Minesweeper game in Java, by progressively completing four programming levels. Theoretically, the project helped to enhance knowledge about the application of OOP principles, utilization of Java Swing library in graphical representation of the game, inserting and playing sounds in Java as well as development of UML class diagram. Practically, it improved our programming and reinforced our organizing and planning skills. Throughout the development of the project, our teamwork skills were enhanced by organizing meetings, dividing the responsibilities and coming into agreement with different ideas. Although, the created game meets the requirements of a classical Minesweeper, further improvements can be made by adding more entertaining and complex features like addition of animations, implementing list of high scores and developing several levels of difficulties. Finally, all of the objectives of the project were successfully achieved and done appropriate to the conditions that were asked.

# Major Contributions

**Alisher Aitbayev:** idea about taking “Techies” as a main theme of project and its implementation through insertion of topic-related images (levels B, C and D) and sounds (all levels); development of the major logic applied in levels B, C, D; major graphics of level D; implementation of reset button in levels B, C, D.

**Kalamkas Zhagyparova:**

development of the major logic in levels A, C; idea about the application of gifs and its implementation in levels B, C and D; major graphics of levels B, C; implementation of timer in levels B, C and D.

# References

|  |  |
| --- | --- |
| [1] | Package javax.swing (n.d). In Oracle Docs. Retrieved March 22 – April 19, 2019, from:  https://docs.oracle.com/javase/8/docs/api/index.html?javax/swing/package-summary.html |
| [2] | How to add text to JLabel (2015). In Stack Overflow. Retrieved April 16, 2019, from:  <https://stackoverflow.com/questions/27624270/how-to-add-text-to-jlabel> |
| [3] | JPanel - Java GUI Building Tutorial #2. MrDeathJockey [Online Video]. Retrieved from https://www.youtube.com/watch?v=pFYOZNc5Jt8 |