**Application Programming**

*Assignment 2*

***Nikoloz Muladze***

I made a Minesweeper game for this assignment.

a) All classes:

public class: CE203\_1904937\_Ass2

private sub-classes of a public class: MouseDetector

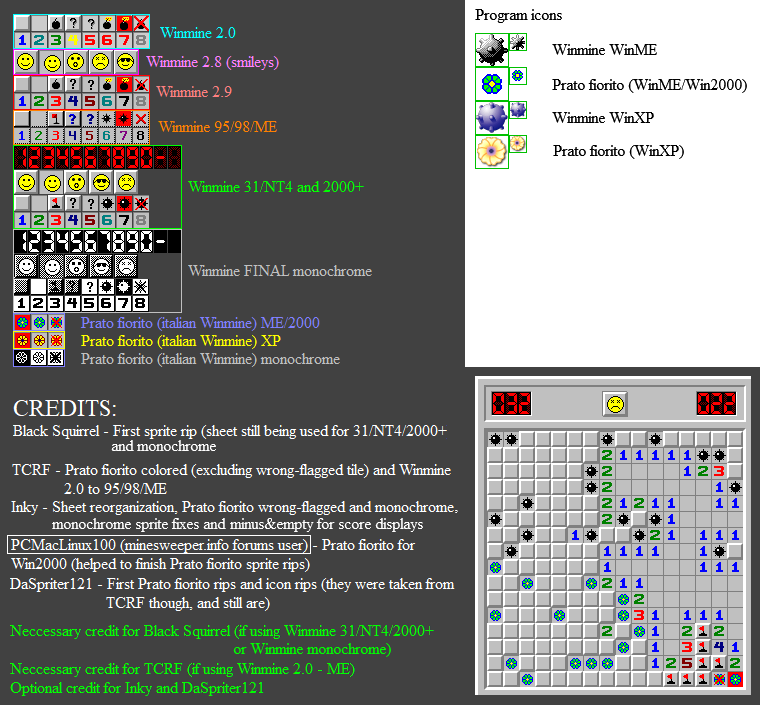
Other classes inside this .java file: Keyboard, Main, Scoreboard.

b) All the sprites/images used in the game:



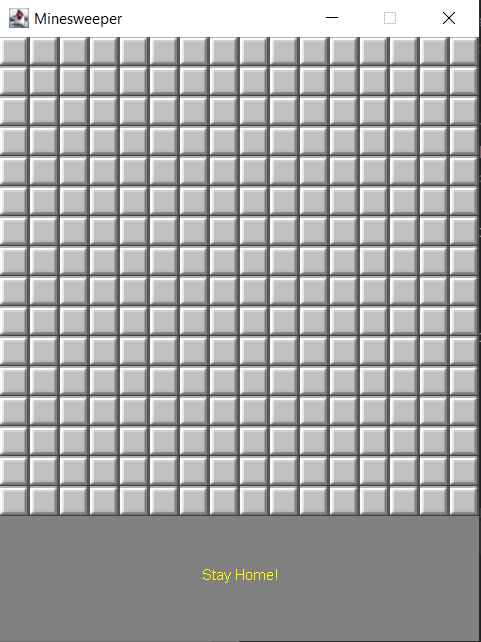


Reference sheet: [PC / Computer - Minesweeper - Everything - The Spriters Resource (spriters-resource.com)](https://www.spriters-resource.com/pc_computer/minesweeper/sheet/19849/)

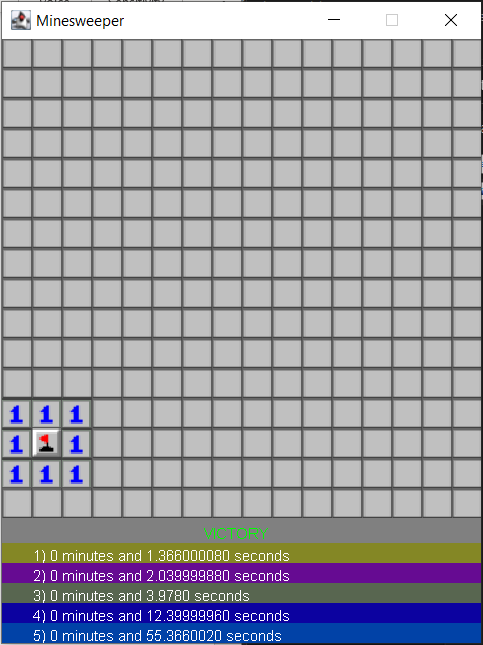
 I used Winmine 31/NT4 and 2000+

c) For the sake of simplicity, I changed N\_MINES variable to 1 which generates only 1 mine, on submission N\_MINES will be set to 45. And TITLE of JFrame as required by the assignment details to my ***Registration number: 1904937***

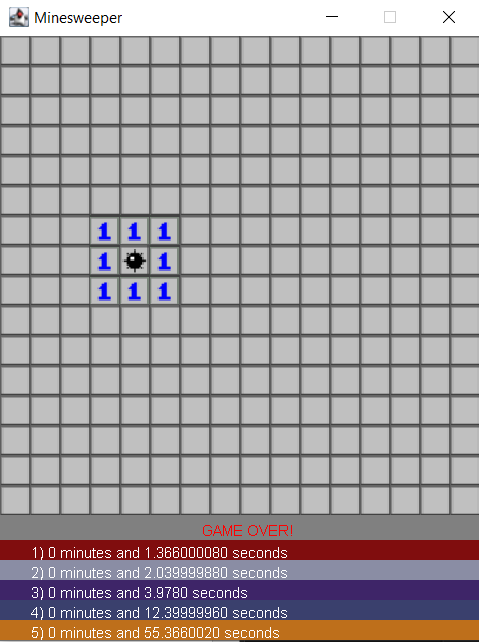
1) Looks of the program on a startup execution:



2) On Win:



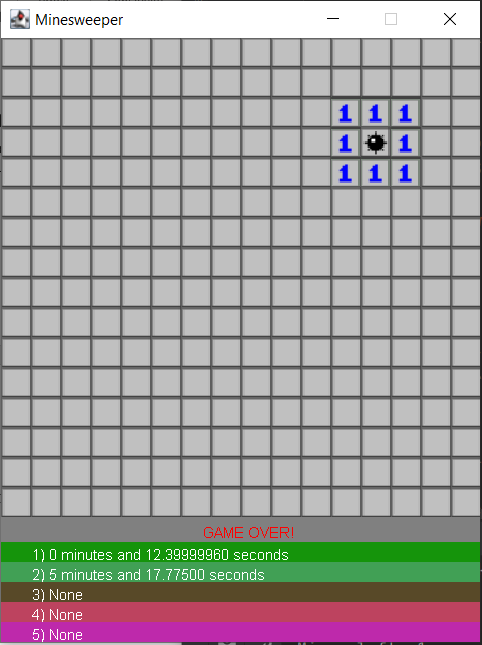
3) On Lose:



d) Functionality of the game:

Game works just like the normal minesweeper game, without the timer being displayed but it shows the time in a scoreboard once you win. The time is sorted in ascending order and displays only top 5 scores, but all the previous scores are stored in a text file.

Time is recorded in minutes, I used BigDecimal to display minutes as an integer part and subtract double number by an integer part to get a decimal part only. Then multiply by 60 and that’s how seconds are being calculated.

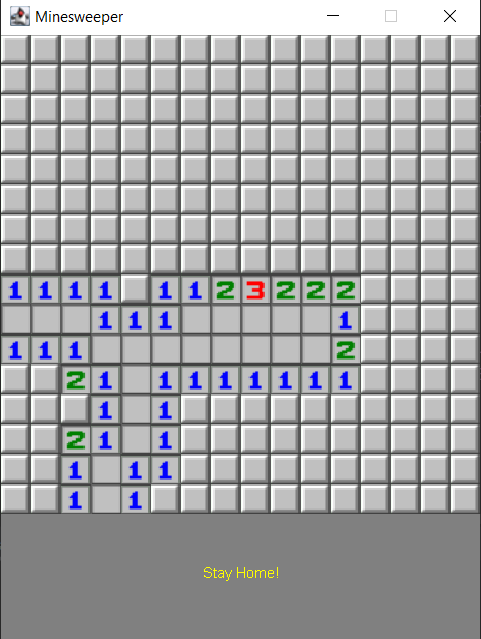
Scoreboard class is for writing a file and reading a file, 2 separate methods. In case of any compile errors the scoreboard.txt must be deleted. If the readFile() method reading convention is not met as expected, there will be a huge compile error (Convention is “\n” new line). If there is no scoreboard.txt it is created once you win. And if there are not enough wins, less than 5, “None” is displayed instead of minutes and seconds.

1) There also is a class Keyboard, I couldn’t think of any functionality. So once you click W or S it outputs a message in a terminal, I made it because assignment asked for making for Keyboard and Mouse detecting classes



2) How the mouse Works:

a) Left Click reveals the cell like so:



b) Right Click sets the Flag/Mark (Also terminal says how many marks are left to use):



c) If the cell contains a bomb after Left Clicking, it reveals all the bombs:

d) If you set a Flag and that cell does not contain a bomb, it renders a crossed mine, meaning there was no bomb in that cell:



**EVENT-DRIVEN MODEL**

As the name says, event-driven model is the way of making application designs with an event-driven system.

Event-driven program after initialization simply waits for events to occur and responds to them.

For example, Keyboard or Mouse events.

addMouseListener(new MouseDetector());

As explained in the notes:

1) Initialisation

2) Wait for events…

3) If event occurs perform associated action

4) Repeat steps 2-3 until exit action…

5) Exit

I would like to add interesting fact about KeyListener. It only works if it is added after the program have been rendered. After the following line:

setVisible(true);

**Java JDBC**

JDBC stands for Java Database Connectivity.

JDBC is a JAVA API for connecting and executing the query with the database.

To connect to the database in Java using JDBC. There are 5 steps to follow:

1) Register the Driver class

2) Create connection

3) Create statement

4) Execute queries

5) Close connection

1) **Register the Driver class**

We are going to use forName() method to register the driver class as follows:

public static void forName(String className) throws ClassNotFoundException

example: Class.forName(“com.mysql.jdbc.Driver”);

2)  **Create the connection object**

Here we are going to use getConnection()

1) public static Connection getConnection(String url) throws SQLException

2) public static Connection getConnection(String url, String name, String password) throws SQLException

Example: Connection con = DriverManager.getConnection(“jdbc:mysql://localhost:8080”, “root”, “password”);

3)  **Create the Statement object**

To create statement the method used is called createStatement():

public Statement createStatement()throws SQLException

Example: Statement stmt = con.createStatement();

**4) Execute the queries**

Next we are going to use executeQuery():

public ResultSet executeQuery(String sql)throws SQLException

Example:

ResultSet rs = stmt.executeQuery(“select \* from dataTable”);

while(rs.next()) {

rs.getInt(1) //etc

}

**5) Close the connection**

public void close()throws SQLException

Example: con.close();

The following steps are taken from

<https://www.javatpoint.com>

now let’s say our scoreboard is in MYSQL database:

1) we need to create a Database:

CREATE DATABASE scoreboard;

USE scoreboard;

CREATE TABLE scoresTable(scores VARCHAR(20))

2) Now we need to connect, this is how it would look like:

import java.sql.\*;  
class MysqlCon{  
 public static void main(String args[]){  
 try{  
 Class.*forName*("com.mysql.jdbc.Driver");  
 Connection con=DriverManager.*getConnection*(  
 "jdbc:mysql://localhost:8080","root","password");  
 Statement stmt=con.createStatement();  
 ResultSet rs=stmt.executeQuery("select \* from scoreboard");  
 while(rs.next()) {  
 System.*out*.println(rs.getString(1));

}  
 con.close();  
 }catch(Exception e){ e.printStackTrace();}  
 }  
}

Password can be anything, it is determined after installation of MYSQL.

.getString(1) gets values from the table by column index.

I haven’t done any logic of how it would be sorted, stored in an ArrayList, but it is a basic way of connecting to a JDBC as required.