OOP ASSIGNMENT 3

Minesweeper

Sahil Garg	8
Shounak Dey	32
Amrit Goyal	36
Kushagra Mittal	46
Sidhant Tibrewal	47

CSE SECTION D – SEM 3

OVERVIEW

PROBLEM STATEMENT

Our project aims to recreate the classic game of minesweeper. We have used java swing to make the GUI. We have also tried to incorporate proper data abstraction and encapsulation.

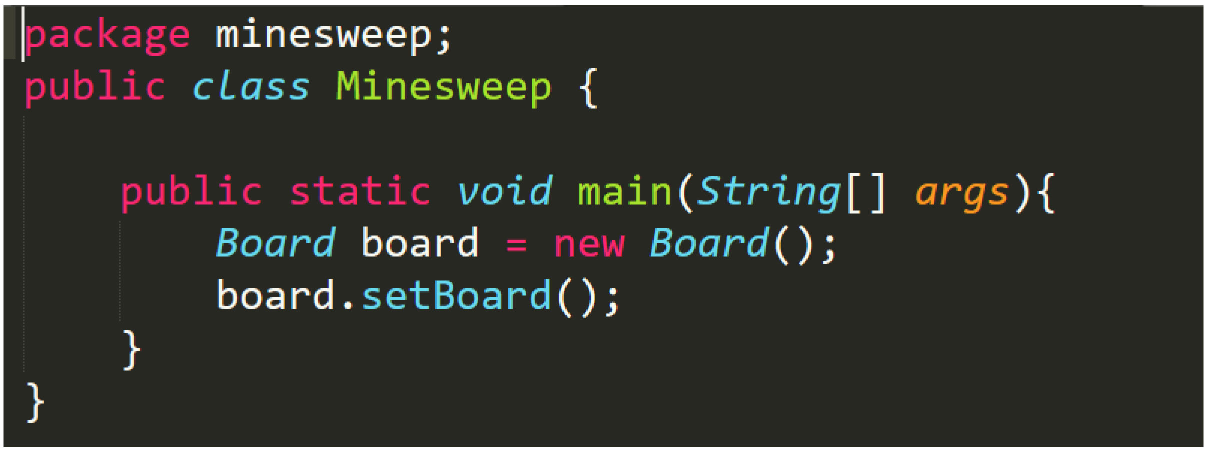
The game works like this:

There is an 8X8 grid in which 10 places are bombs. Whenever you click a button, if it is not a bomb, it displays the number of bombs around it. Also, the places with no bombs around it are exposed in a chain reaction. The goal is to select all the buttons without the bomb and leave the ones with the bomb untouched.

ALGORITHM/EXPLANATION

We have three classes.

First is the main class called Minesweep which calls the second class Board to create the GUI.



```java
package minesweep;
public class Minesweep {

    public static void main(String[] args){
        Board board = new Board();
        board.setBoard();
    }
}
```

The Board class contains a matrix of Cell class, dimensions of the board and some other variables.

```java
public class Board {
    private Cell[][] cells;
    private int cellID = 0;
    private int side = 8;
    private int limit = side-2;
```

The Cell class is basically a JButton class with additional variables like value, id, etc. and some additional methods.

```java
public class Cell implements ActionListener{
    private JButton button;
    private Board board;
    private int value;
    private int id;
    private boolean notChecked;
```

<u>OOP CONCEPTS APPLIED</u>

Interfaces and inheritance is used for enabling the functions of the JPanel, JButton and the JFrame classes, as well as to set the listeners for user interaction.

Data Abstraction is used to hide the members of the class from external classes that might affect the functioning of the game.

Packages are created and used to provide a structure for the entire application and allow easy distinguishing of app modules.

Arraylists are used to generate random numbers for locations of mines in the game, thus implementing the concepts of Generics in Java.

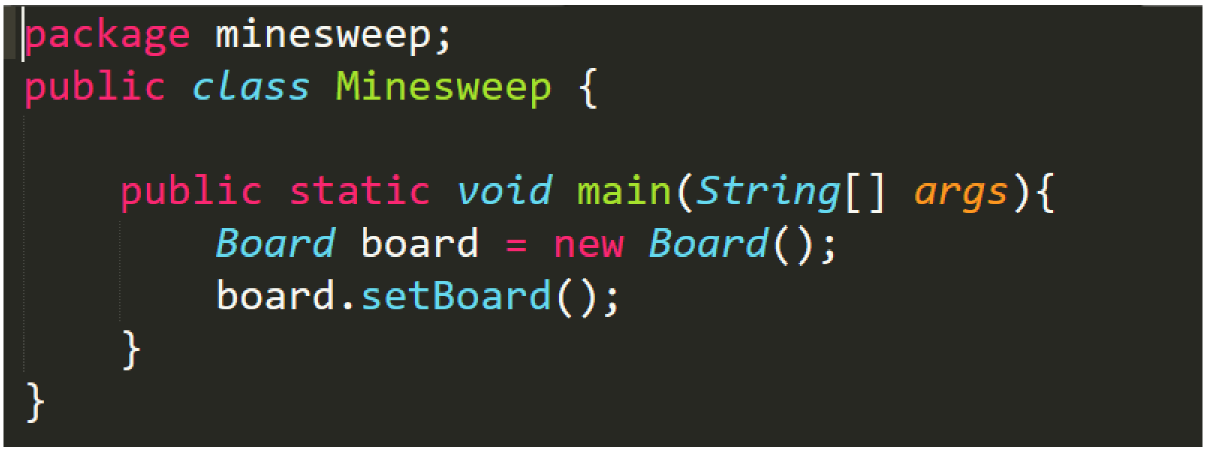
Swings is used to run the application and provide a smooth GUI experience in the game.

Multi - Threading is used for starting an event dispatching thread and running the GUI interface in a new thread.

MODULES/USER DEFINED FUNCTION

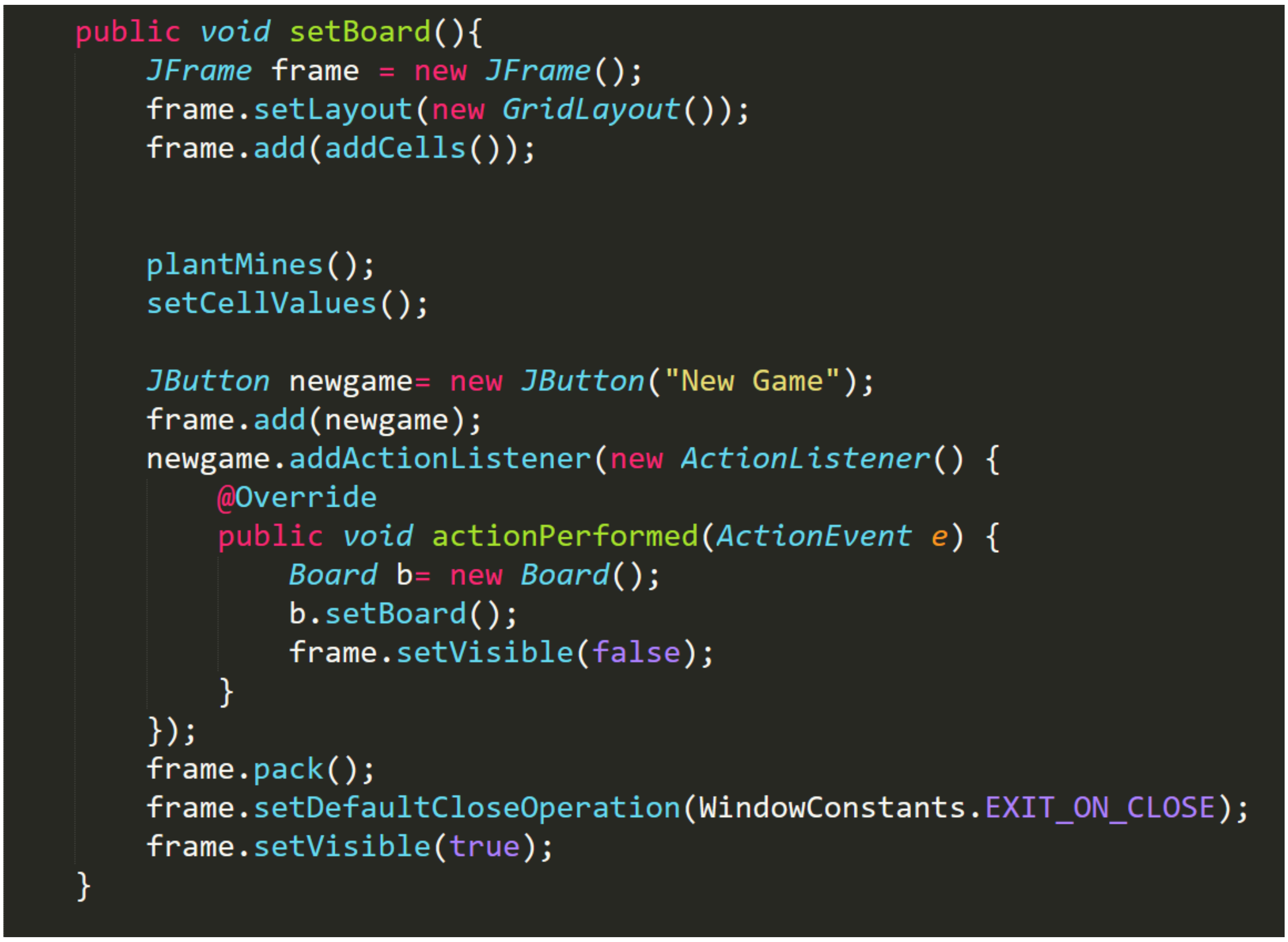
Following is how the code functions with all the methods:

Minesweep class creates a new board object and calls its setBoard function.



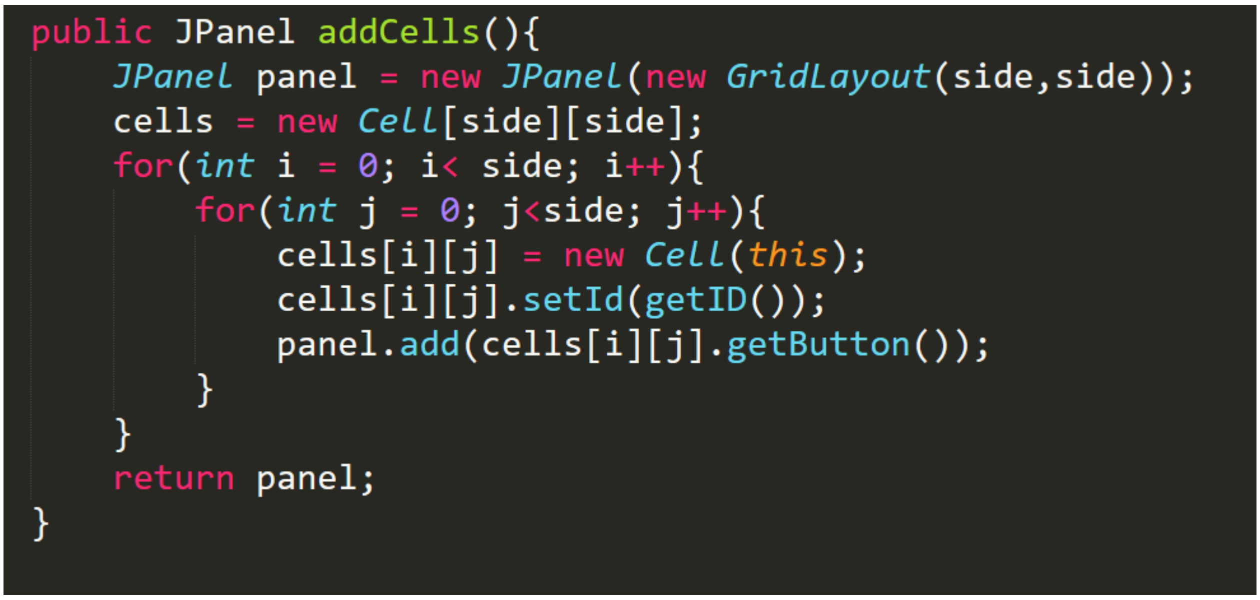
```
package minesweep;
public class Minesweep {

    public static void main(String[] args){
        Board board = new Board();
        board.setBoard();
    }
}
```

The setBoard function of Board class creates the JFrame.



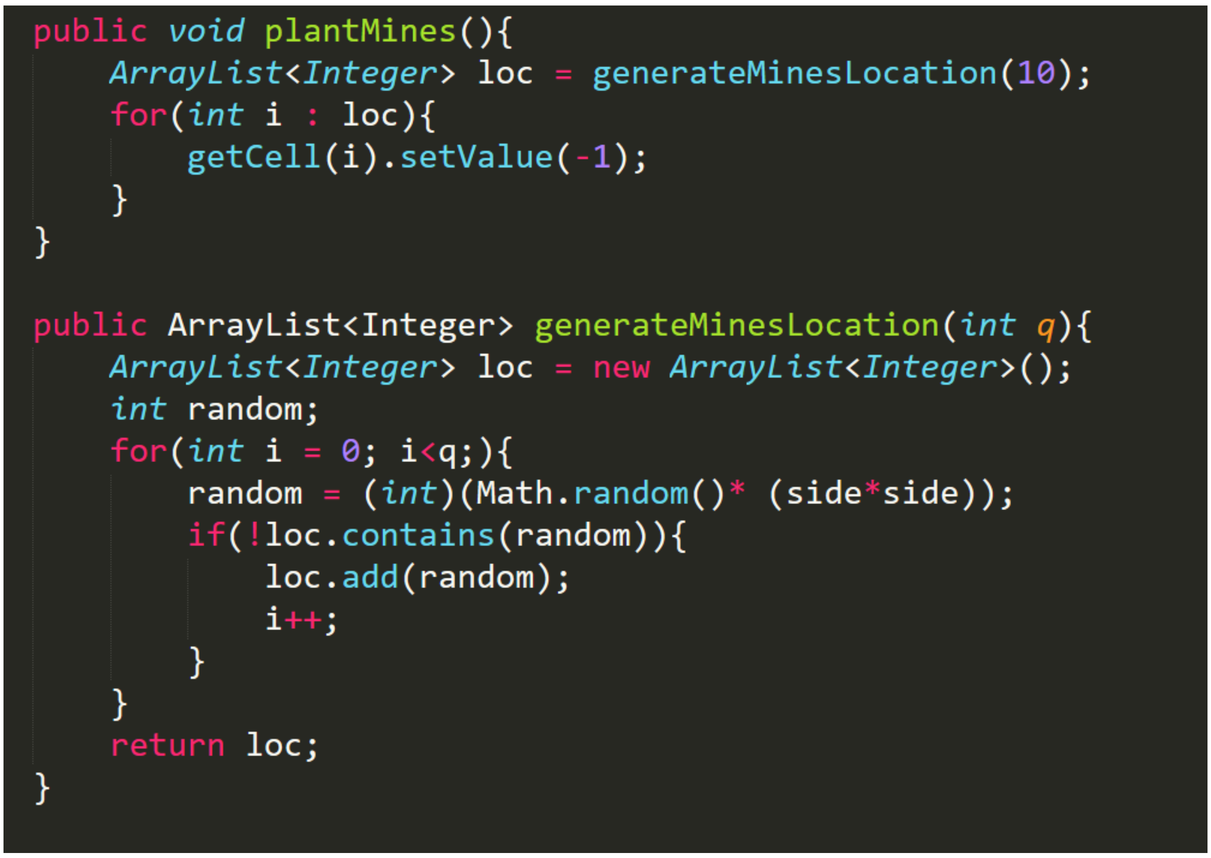
```java
public void setBoard(){
    JFrame frame = new JFrame();
    frame.setLayout(new GridLayout());
    frame.add(addCells());


    plantMines();
    setCellValues();

    JButton newgame= new JButton("New Game");
    frame.add(newgame);
    newgame.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Board b= new Board();
            b.setBoard();
            frame.setVisible(false);
        }
    });
    frame.pack();
    frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    frame.setVisible(true);
}
```

It calls the addCells() function which creates new Cell objects, assign them a unique id and add them to a panel.



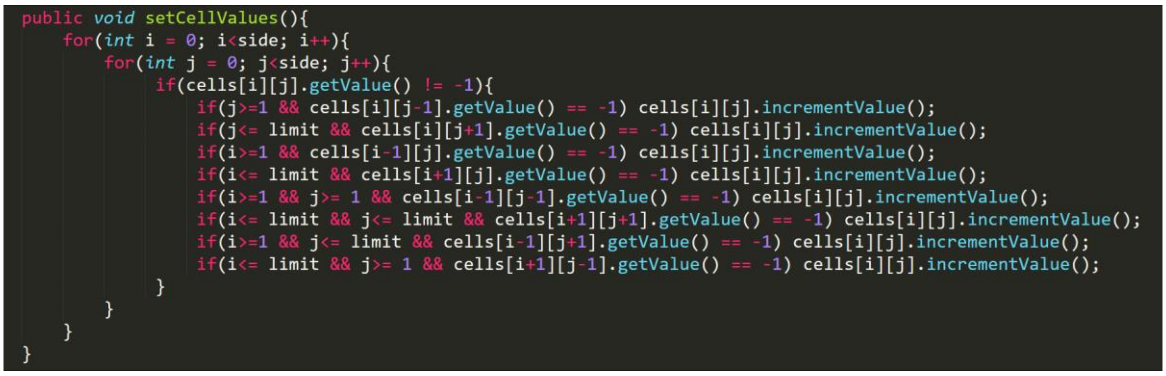
```java
public JPanel addCells(){
    JPanel panel = new JPanel(new GridLayout(side,side));
    cells = new Cell[side][side];
    for(int i = 0; i< side; i++){
        for(int j = 0; j<side; j++){
            cells[i][j] = new Cell(this);
            cells[i][j].setId(getID());
            panel.add(cells[i][j].getButton());
        }
    }
    return panel;
}
```

It then calls the plantMines() function which creates a random list of 10 numbers from 1 to 64 an sets the respective cells to the value of (-1) which means it is a bomb.



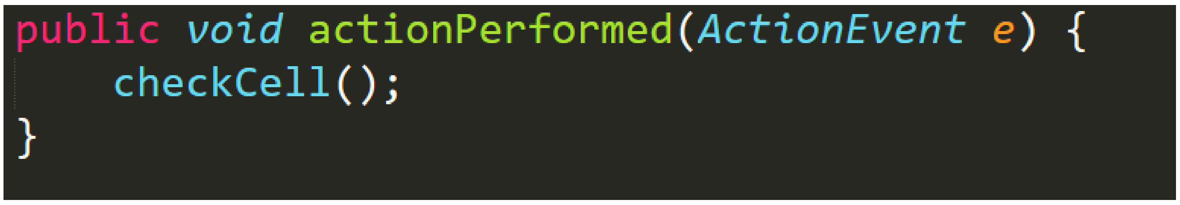
```java
public void plantMines(){
    ArrayList<Integer> loc = generateMinesLocation(10);
    for(int i : loc){
        getCell(i).setValue(-1);
    }
}

public ArrayList<Integer> generateMinesLocation(int q){
    ArrayList<Integer> loc = new ArrayList<Integer>();
    int random;
    for(int i = 0; i<q;){
        random = (int)(Math.random()* (side*side));
        if(!loc.contains(random)){
            loc.add(random);
            i++;
        }
    }
    return loc;
}
```

Next it calls setCellValues() which scans cells surrounding each cell, and if a bomb is found it increments the value of the cell.

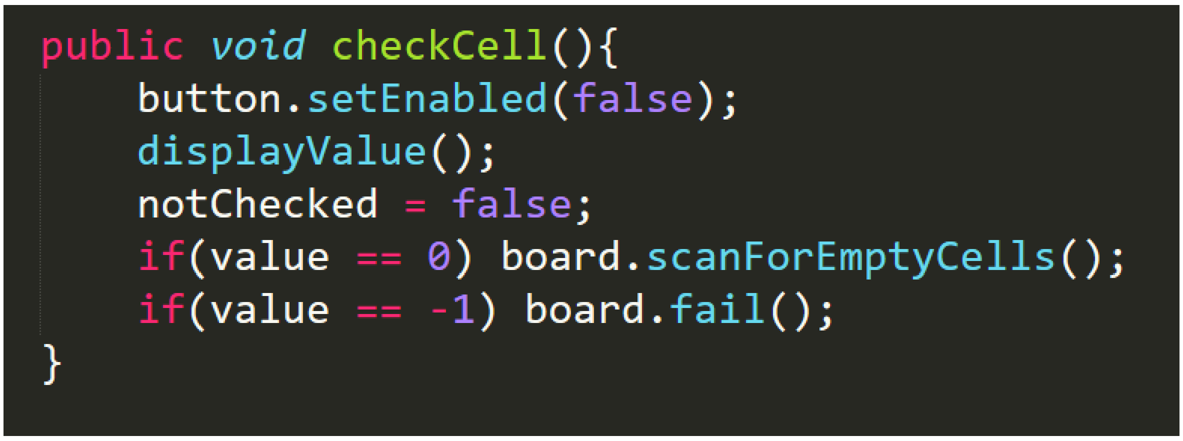


```java
public void setCellValues(){
    for(int i = 0; i<side; i++){
        for(int j = 0; j<side; j++){
            if(cells[i][j].getValue() != -1){
                if(j>=1 && cells[i][j-1].getValue() == -1) cells[i][j].incrementValue();
                if(j<= limit && cells[i][j+1].getValue() == -1) cells[i][j].incrementValue();
                if(i>=1 && cells[i-1][j].getValue() == -1) cells[i][j].incrementValue();
                if(i<= limit && cells[i+1][j].getValue() == -1) cells[i][j].incrementValue();
                if(i>=1 && j>= 1 && cells[i-1][j-1].getValue() == -1) cells[i][j].incrementValue();
                if(i<= limit && j<= limit && cells[i+1][j+1].getValue() == -1) cells[i][j].incrementValue();
                if(i>=1 && j<= limit && cells[i-1][j+1].getValue() == -1) cells[i][j].incrementValue();
                if(i<= limit && j>= 1 && cells[i+1][j-1].getValue() == -1) cells[i][j].incrementValue();
            }
        }
    }
}
```

When a cell is pressed it calls the checkCell() function which does the following:

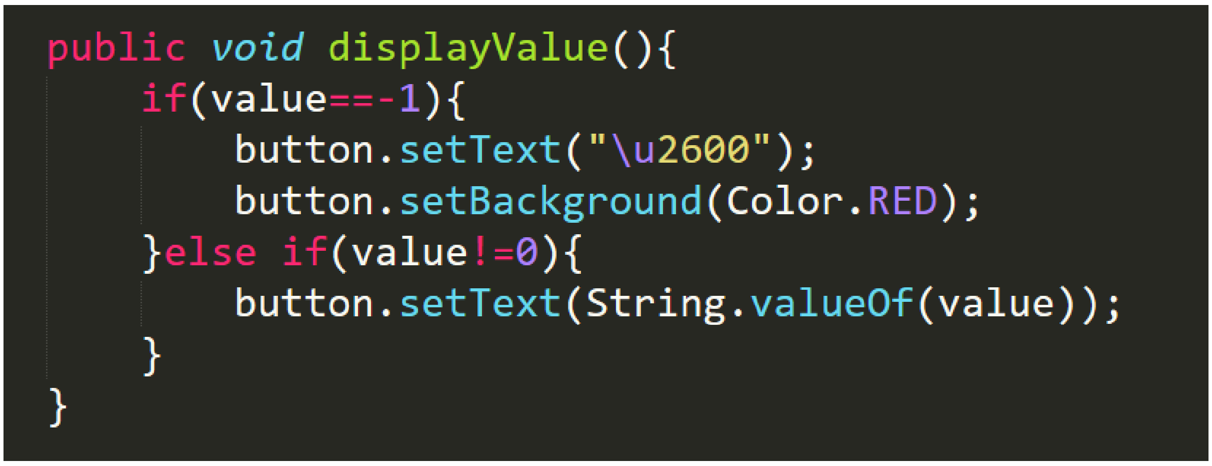


```java
public void actionPerformed(ActionEvent e) {
    checkCell();
}
```



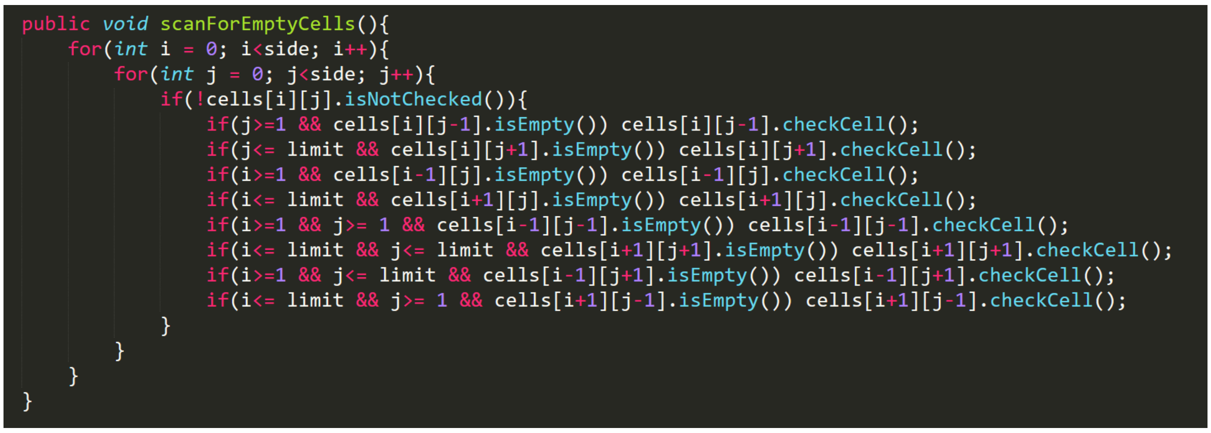
```java
public void checkCell(){
    button.setEnabled(false);
    displayValue();
    notChecked = false;
    if(value == 0) board.scanForEmptyCells();
    if(value == -1) board.fail();
}
```

It displays the value of the cell using displayValue() in which, if the value is -1, it sets the text to /u2600 which is Unicode value of sun and the colour to red, otherwise it sets the text to the value.



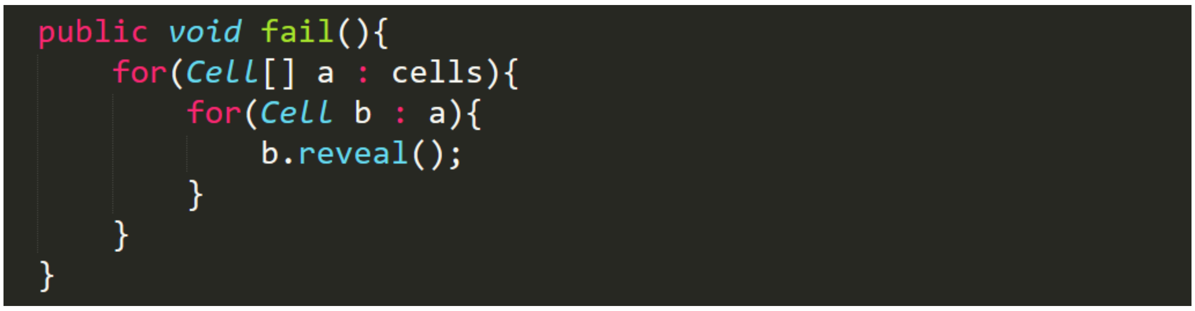
```java
public void displayValue(){
    if(value==-1){
        button.setText("\u2600");
        button.setBackground(Color.RED);
    }else if(value!=0){
        button.setText(String.valueOf(value));
    }
}
```

If the value of cell is 0 checkCell() scans and displays all the cells with value 0 till a cell with value not 0 is encountered using scanForEmptyCells().

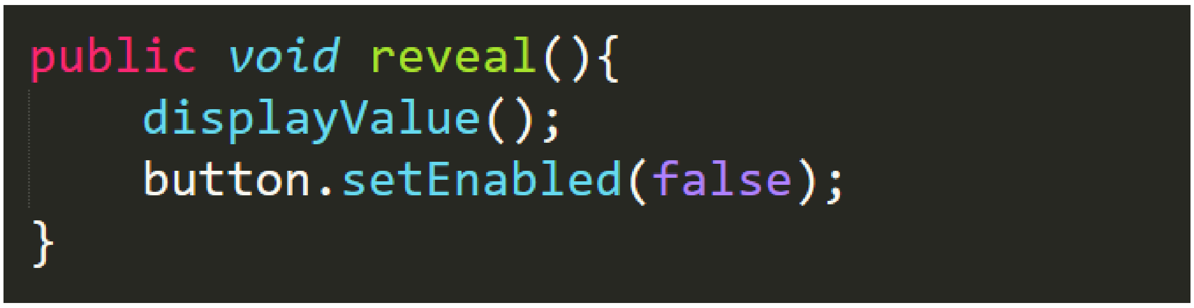


```java
public void scanForEmptyCells(){
    for(int i = 0; i<side; i++){
        for(int j = 0; j<side; j++){
            if(!cells[i][j].isNotChecked()){
                if(j>=1 && cells[i][j-1].isEmpty()) cells[i][j-1].checkCell();
                if(j<= limit && cells[i][j+1].isEmpty()) cells[i][j+1].checkCell();
                if(i>=1 && cells[i-1][j].isEmpty()) cells[i-1][j].checkCell();
                if(i<= limit && cells[i+1][j].isEmpty()) cells[i+1][j].checkCell();
                if(i>=1 && j>= 1 && cells[i-1][j-1].isEmpty()) cells[i-1][j-1].checkCell();
                if(i<= limit && j<= limit && cells[i+1][j+1].isEmpty()) cells[i+1][j+1].checkCell();
                if(i>=1 && j<= limit && cells[i-1][j+1].isEmpty()) cells[i-1][j+1].checkCell();
                if(i<= limit && j>= 1 && cells[i+1][j-1].isEmpty()) cells[i+1][j-1].checkCell();
            }
        }
    }
}
```

If the value is -1, it calls the fail() function which displays all the cells using reveal() function.

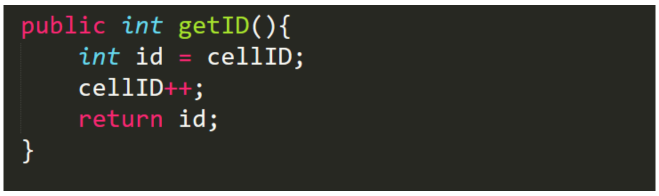


```java
public void fail(){
    for(Cell[] a : cells){
        for(Cell b : a){
            b.reveal();
        }
    }
}
```

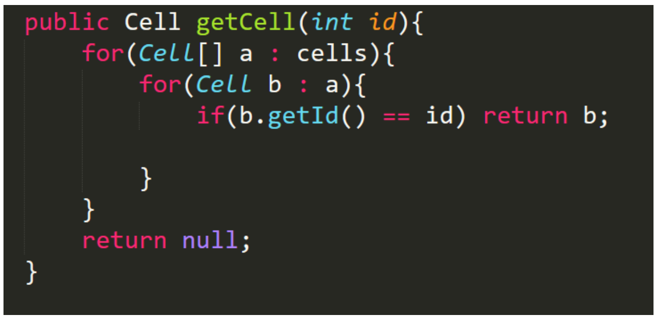


```java
public void reveal(){
    displayValue();
    button.setEnabled(false);
}
```

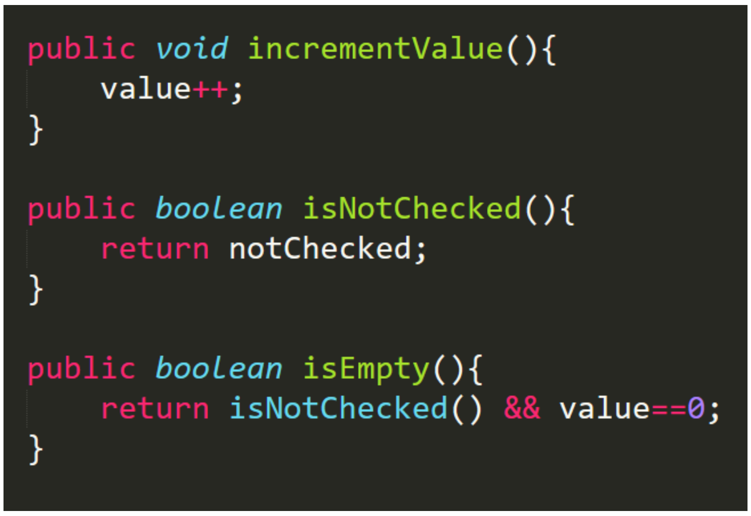
Other methods that are used in above sequences are:



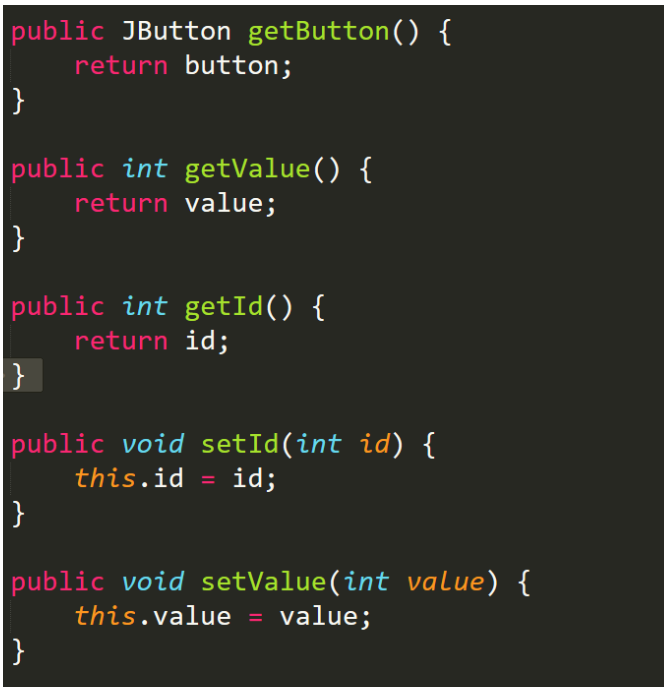
```java
public int getID(){
    int id = cellID;
    cellID++;
    return id;
}
```



```java
public Cell getCell(int id){
    for(Cell[] a : cells){
        for(Cell b : a){
            if(b.getId() == id) return b;

        }
    }
    return null;
}
```



```java
public void incrementValue(){
    value++;
}

public boolean isNotChecked(){
    return notChecked;
}

public boolean isEmpty(){
    return isNotChecked() && value==0;
}
```



```java
public JButton getButton() {
    return button;
}

public int getValue() {
    return value;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public void setValue(int value) {
    this.value = value;
}
```

SOURCE CODE

Minesweep class

```
package minesweep;
public class Minesweep {

    public static void main(String[] args){
        Board board = new Board();
        board.setBoard();
    }
}
```

Board class

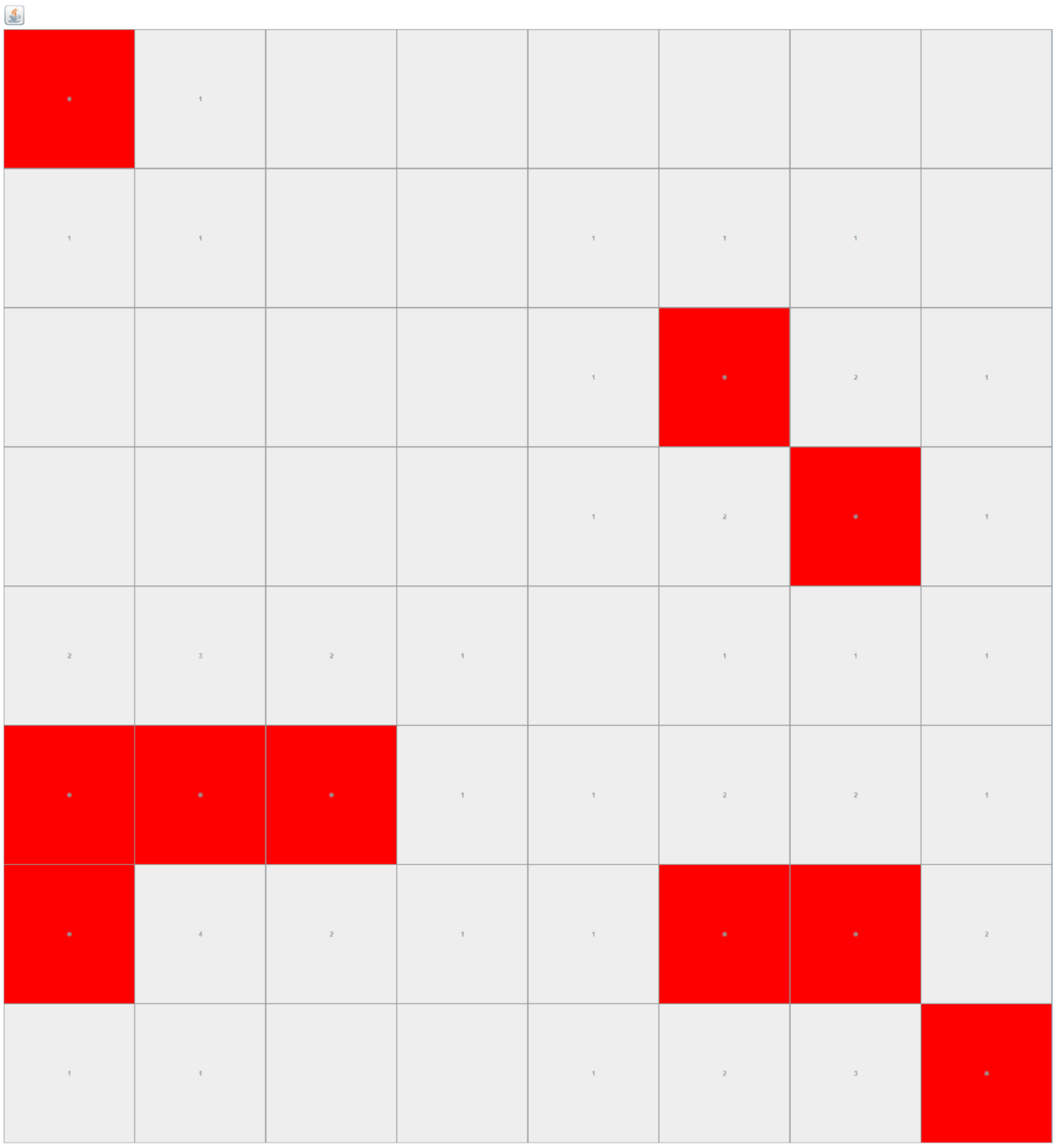
```
package minesweep;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class Board {
    private Cell[][] cells;
    private int cellID = 0;
    private int side = 8;
    private int limit = side-2;

    public void setBoard(){
        JFrame frame = new JFrame();
        frame.setLayout(new GridLayout());
        frame.add(addCells());
        plantMines();
        setCellValues();

        JButton newgame= new JButton("New Game");
        frame.add(newgame);
        newgame.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Board b= new Board();
                b.setBoard();
                frame.setVisible(false);
            }
        });
        frame.pack();
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

    public JPanel addCells(){
        JPanel panel = new JPanel(new GridLayout(side,side));
        cells = new Cell[side][side];
        for(int i = 0; i< side; i++){
            for(int j = 0; j<side; j++){
                cells[i][j] = new Cell(this);
                cells[i][j].setId(getID());
                panel.add(cells[i][j].getButton());
            }
        }
        return panel;
    }

    public void plantMines(){
        ArrayList<Integer> loc = generateMinesLocation(10);
        for(int i : loc){
            getCell(i).setValue(-1);
        }
    }

    public ArrayList<Integer> generateMinesLocation(int q){
        ArrayList<Integer> loc = new ArrayList<Integer>();
        int random;
        for(int i = 0; i<q;){
            random = (int)(Math.random()* (side*side));
            if(!loc.contains(random)){
                loc.add(random);
                i++;
            }
        }
        return loc;
    }

    public void setCellValues(){
        for(int i = 0; i<side; i++){
            for(int j = 0; j<side; j++){
                if(cells[i][j].getValue() != -1){
                    if(j>=1 && cells[i][j-1].getValue() == -1) cells[i][j].incrementValue();
                    if(j<= limit && cells[i][j+1].getValue() == -1) cells[i][j].incrementValue();
                    if(i>=1 && cells[i-1][j].getValue() == -1) cells[i][j].incrementValue();
                    if(i<= limit && cells[i+1][j].getValue() == -1) cells[i][j].incrementValue();
                    if(i>=1 && j>= 1 && cells[i-1][j-1].getValue() == -1) cells[i][j].incrementValue();
                    if(i<= limit && j<= limit && cells[i+1][j+1].getValue() == -1)
    cells[i][j].incrementValue();
                    if(i>=1 && j<= limit && cells[i-1][j+1].getValue() == -1)
    cells[i][j].incrementValue();
                    if(i<= limit && j>= 1 && cells[i+1][j-1].getValue() == -1)
    cells[i][j].incrementValue();
                }
            }
        }
    }


    public void scanForEmptyCells(){
        for(int i = 0; i<side; i++){
            for(int j = 0; j<side; j++){
                if(!cells[i][j].isNotChecked()){
                   if(j>=1 && cells[i][j-1].isEmpty()) cells[i][j-1].checkCell();
                   if(j<= limit && cells[i][j+1].isEmpty()) cells[i][j+1].checkCell();
                   if(i>=1 && cells[i-1][j].isEmpty()) cells[i-1][j].checkCell();
                   if(i<= limit && cells[i+1][j].isEmpty()) cells[i+1][j].checkCell();
                   if(i>=1 && j>= 1 && cells[i-1][j-1].isEmpty()) cells[i-1][j-1].checkCell();
                   if(i<= limit && j<= limit && cells[i+1][j+1].isEmpty()) cells[i+1][j+1].checkCell();
                   if(i>=1 && j<= limit && cells[i-1][j+1].isEmpty()) cells[i-1][j+1].checkCell();
                   if(i<= limit && j>= 1 && cells[i+1][j-1].isEmpty()) cells[i+1][j-1].checkCell();
                }
            }
        }
```

```
    }

    public int getID(){
        int id = cellID;
        cellID++;
        return id;
    }

    public Cell getCell(int id){
        for(Cell[] a : cells){
            for(Cell b : a){
                if(b.getId() == id) return b;

            }
        }
        return null;
    }

    public void fail(){
        for(Cell[] a : cells){
            for(Cell b : a){
                b.reveal();
            }
        }
    }
}
```

Cell class

```
package minesweep;
import javax.swing.*;
import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


public class Cell implements ActionListener{

   private JButton button;

   private Board board;

   private int value;

   private int id;

   private boolean notChecked;


   public Cell(Board board){

      button = new JButton();

      button.addActionListener(this);

      button.setPreferredSize(new Dimension(20,20));

      button.setMargin(new Insets(0,0,0,0));

      this.board = board;

      notChecked = true;

   }


   public JButton getButton() {

      return button;

   }


   public int getValue() {

      return value;

   }


   public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public void setValue(int value) {
    this.value = value;
}

public void displayValue(){
    if(value==-1){
        button.setText("\u2600");
        button.setBackground(Color.RED);
    }else if(value!=0){
        button.setText(String.valueOf(value));
    }
}

public void checkCell(){
    button.setEnabled(false);
    displayValue();
    notChecked = false;
    if(value == 0) board.scanForEmptyCells();
    if(value == -1) board.fail();
}

public void incrementValue(){
        value++;
    }

    public boolean isNotChecked(){
        return notChecked;
    }

    public boolean isEmpty(){
        return isNotChecked() && value==0;
    }

    public void reveal(){
        displayValue();
        button.setEnabled(false);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        checkCell();
    }

}
```

GAME SCREENSHOT



*	1						
1	1			1	1	1	
				1	*	2	1
				1	2	*	1
2	3	2	1		1	1	1
*	*	*	1	1	2	2	1
*	4	2	1	1	*	*	2
1	1			1	2	3	*

POSSIBLE FUTURE ADDITION

In the real game you can set the bomb as flagged. This can be incorporated in future. Also a message displaying “YOU HAVE WON” can be added using JDialogBox.