```java
package minesweeper;

import javafx.application.Platform;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;

/**
 * Bomb cell
 * @author Thomas Nonis
 * @author thomas.nonis@studenti.unitn.it
 */
public class Bomb extends Cell{
    /**
     * The background color of the circle representing the bomb
     */
    private static final Color BOMB_BG = Color.RED;

    Bomb(){}

    /**
     * Triggers the bomb
     * @param mainRef The reference to the main class
     */
    public void trigger(Minesweeper mainRef){
        super.trigger(mainRef);

        this.getChildren().add( new Circle(SIZE / 2 * 0.75, BOMB_BG) );

        mainRef.cellsLeft--;

        if(!mainRef.peekMode){
            new Prompt(Prompt.Status.LOSS);
            //Based on the interpretation, choose one:
            Platform.exit();
            //mainRef.reset();
        }
    }
}

```

```java
package minesweeper;

import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;

/**
 * Abstract class for all cells
 * @author Thomas Nonis
 * @author thomas.nonis@studenti.unitn.it
 */
public abstract class Cell extends StackPane{
    /**
     * The size of the cell
     */
    protected static final double SIZE = 50.0;

    /**
     * The background color of an unclicked cell
     */
    private static final Color DEF_BG = Color.BLUE;

    /**
     * The background color of a clicked cell
     */
    private static final Color SHOW_BG = Color.YELLOW;

    /**
     * The rectangle that acts as the background
     */
    private Rectangle background;

    Cell(){
        background = new Rectangle(SIZE, SIZE, DEF_BG);

        this.getChildren().add(background);
    }

    /**
     * Triggers the cell
     * @param mainRef The reference to the main class
     */
    public void trigger(Minesweeper mainRef){
        background.setFill(SHOW_BG);
    }
}

```

```java
package minesweeper;

import javafx.scene.control.Label;

/**
 * Normal cell
 * @author Thomas Nonis
 * @author thomas.nonis@studenti.unitn.it
 */
public class Normal extends Cell{
    /**
     * Keeps track of the cell's status
     */
    private boolean isActive;

    /**
     * The label that represents the number of surrounding bombs
     */
    Label lbl = new Label();

    Normal(){
        isActive = true;
    }

    /**
     * Triggers the cell
     * @param mainRef The reference to the main Class
     */
    public void trigger(Minesweeper mainRef){
        if(isActive){
            super.trigger(mainRef);
            this.getChildren().add(lbl);
            mainRef.cellsLeft--;
            isActive = false;
        }
    }

    /**
     * Computes the number of surrounding bombs
     * @param cells The grid of cells, that keeps the locations
     * @param x The X coordinate of the current cell
     * @param y The Y coordinate of the current cell
     */
    public void setCloseBombs(Cell[][] cells, int x, int y){
        int n = 0;

        //LOL
        if(x > 0 && y > 0 && cells[x-1][y-1] instanceof Bomb) n++;    //top left
        if(x > 0 && cells[x-1][y] instanceof Bomb) n++;      //left
        if(x > 0 && y < cells.length-1 && cells[x-1][y+1] instanceof Bomb) n++;    //bottom left
        if(y > 0 && cells[x][y-1] instanceof Bomb) n++;      //top
        if(x < cells.length - 1 && y > 0 && cells[x+1][y-1] instanceof Bomb) n++;    //top right
        if(x < cells.length-1 && cells[x+1][y] instanceof Bomb) n++;      //right
        if(y < cells.length-1 && cells[x][y+1] instanceof Bomb) n++;      //bottom
        if(x < cells.length-1 && y < cells.length-1 && cells[x+1][y+1] instanceof Bomb) n++;
  //bottom right

        lbl.setText( Integer.toString(n) );
    }
}

```

```java
package minesweeper;

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

/**
 * Prompt class, used to display the WIN/LOSS popup
 * @author Thomas Nonis
 * @author thomas.nonis@studenti.unitn.it
 */
public class Prompt {
    public static enum Status{
        VICTORY("Hai vinto!"),
        LOSS("Hai perso :(");

        private String msg;
        Status(String msg){
            this.msg = msg;
        }

        public String getMsg(){
            return msg;
        }
    }

    Prompt(Status status){
        Button btn = new Button("Ok");
        Label txt = new Label(status.getMsg());
        VBox root = new VBox(txt, btn);
        root.setAlignment(Pos.CENTER);
        root.setSpacing(5.0);
        root.setPadding( new Insets(5.0, 15.0, 5.0, 15.0));
        Scene scene = new Scene(root);
        Stage prompt = new Stage();

        btn.setOnAction(e -> {
            prompt.close();
        });

        prompt.setScene(scene);
        prompt.sizeToScene();
        prompt.initModality(Modality.APPLICATION_MODAL);
        prompt.initStyle(StageStyle.UTILITY);
        prompt.showAndWait();
    }
}

```

```java
package minesweeper;

import java.util.ArrayList;
import java.util.Collections;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.Event;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;


/**
 * Main application of the minesweeper game
 * @author Thomas Nonis
 * @author thomas.nonis@studenti.unitn.it
 */
public class Minesweeper extends Application {
    /**
     * The size of the square grid
     */
    private static final int GRID_SIZE = 9;

    /**
     * The number of bombs in the grid
     */
    private static final int N_BOMBS = 10;

    /**
     * The number of non-bomb cells the user has yet to click
     */
    int cellsLeft;

    /**
     * Keeps track of the mode
     */
    boolean peekMode;

    //GUI elements
    Button testBtn, randomBtn, quitBtn, peekBtn;
    HBox controls;
    GridPane grid;
    VBox root;
    Scene scene;

    /**
     * Matrix used to keep track of the coordinates of the cells
     */
    Cell[][] cells = new Cell[GRID_SIZE][GRID_SIZE];

    public static void main(String[] args) {
        launch(args);
    }

    /**
     * Initializes the application
     */
    @Override
    public void init(){
        testBtn = new Button("Test");
        randomBtn = new Button("Random");
```

```
68          quitBtn = new Button("Abbandona");
69          peekBtn = new Button("Sbircia");
70
71          controls = new HBox(testBtn, randomBtn, quitBtn, peekBtn);
72          grid = new GridPane();
73          root = new VBox(grid, controls);
74
75          scene = new Scene(root);
76
77          grid.setVgap(1.0);
78          grid.setHgap(1.0);
79          grid.setMouseTransparent(true);
80          root.setAlignment(Pos.CENTER);
81          controls.setAlignment(Pos.CENTER);
82          controls.setPadding( new Insets(10.0, 0, 10.0, 0) );
83          controls.setSpacing(25.0);
84
85          reset();
86      }
87
88      /**
89       * Starts the application
90       * @param window The primary stage
91       */
92      @Override
93      public void start(Stage window) {
94          testBtn.setOnAction(e -> {
95              fillTest();
96              grid.setMouseTransparent(false);
97              testBtn.setDisable(true);
98              randomBtn.setDisable(true);
99              quitBtn.setDisable(false);
100             peekBtn.setDisable(false);
101         });
102
103         randomBtn.setOnAction(e -> {
104             fillRandom();
105             grid.setMouseTransparent(false);
106             testBtn.setDisable(true);
107             randomBtn.setDisable(true);
108             quitBtn.setDisable(false);
109             peekBtn.setDisable(false);
110         });
111
112
113         quitBtn.setOnAction(e ->  {
114             Platform.exit();
115         });
116
117         peekBtn.setOnAction(e -> {
118             peekMode = true;
119             peekBtn.setDisable(true);
120         });
121
122         window.setScene(scene);
123         window.sizeToScene();
124         window.setResizable(false);
125         window.setTitle("Minesweeper");
126         window.show();
127     }
128
129     /**
130      * Sets the game to its initial status
131      */
132     public void reset(){
133         peekMode = false;
134         testBtn.setDisable(false);
```

```
135         randomBtn.setDisable(false);
136         quitBtn.setDisable(true);
137         peekBtn.setDisable(true);
138         grid.setMouseTransparent(true);
139         cellsLeft = GRID_SIZE * GRID_SIZE;
140         fill();
141     }
142 
143     /**
144      * Fills the grid with normal cells
145      */
146     private void fill(){
147         for(int y = 0; y < GRID_SIZE; y++){
148             for(int x = 0; x < GRID_SIZE; x++){
149                 setCell(new Normal(), x, y);
150             }
151         }
152         //computeCells() should not be needed at this point, as no bombs are present
153     }
154 
155     /**
156      * Replaces the diagonal and the top right cells with bomb cells
157      */
158     private void fillTest(){
159         for(int i = 0; i < GRID_SIZE; i++){
160             setCell(new Bomb(), i, i);
161         }
162         setCell(new Bomb(), GRID_SIZE - 1, 0);
163         computeCells();
164     }
165 
166     /**
167      * Randomly fills the grid with bomb and normal cells
168      */
169     private void fillRandom(){
170         ArrayList<Cell> pool = new ArrayList<>();
171         for(int i = 0; i < N_BOMBS; i++){
172             pool.add(new Bomb());
173         }
174         for(int i = 0; i < GRID_SIZE * GRID_SIZE - N_BOMBS; i++){
175             pool.add(new Normal());
176         }
177 
178         Collections.shuffle(pool);
179 
180         for(int y = 0; y < GRID_SIZE; y++){
181             for(int x = 0; x < GRID_SIZE; x++){
182                 setCell(pool.remove(0), x, y);
183             }
184         }
185         computeCells();
186     }
187 
188     /**
189      * Sets the cell at the desired coordinates in the grid. If a cell is already present, it is
    replaced.
190      * @param cell The new cell to add
191      * @param x The x coordinate
192      * @param y The y coordinate
193      */
194     private void setCell(Cell cell, int x, int y){
195         grid.getChildren().remove( cells[x][y] ); //if no element was present it does nothing
196         grid.add(cell, x, y);
197         cells[x][y] = cell; //purposely inverted
198         cell.setOnMouseClicked(cellHandler);
199     }
200 
```

```
201     /**
202      * Computes normal cell values
203      */
204     private void computeCells(){
205         for(int y = 0; y < GRID_SIZE; y++){
206             for(int x = 0; x < GRID_SIZE; x++){
207                 if(cells[x][y] instanceof Normal){
208                     ((Normal) cells[x][y]).setCloseBombs(cells, x, y);
209                 }
210             }
211         }
212     }
213 
214     /**
215      * Main event handler for the game
216      */
217     EventHandler<Event> cellHandler = e -> {
218         ((Cell) e.getSource()).trigger(this);
219         if(cellsLeft <= N_BOMBS){
220             new Prompt(Prompt.Status.VICTORY);
221             reset();
222         }
223         if(peekMode){
224             peekMode = false;
225         }
226     };
227 
228 }
229 
```