```
package main;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.util.ArrayList;
import java.util.Random;

import org.junit.Test;

import gui.GameBoard;
import gui.GameFrame;
import gui.PanelButtons;
import gui.PanelLabel;
import gui.PanelOptions;
import object.Cell;

public class Controller {

	private static GameFrame gameFrame;
	private static Container container;
	public static Setting settings;
	public static Cell[][] gameArray;

	public static void main(String[] args) {

		gameFrame = new GameFrame();
		container = gameFrame.getContentPane();
		settings = new Setting();
		newGame();
	}

	/**
	 * set up new game
	 */
	public static void newGame(){
		Controller.gameArray = new Cell

	[Controller.settings.getAcross()][Controller.settings.getDown()];
		newPage();
		bombPlacement();
	}

	/**
	 * create new game board
	 */
	public static void newPage() {
		container.removeAll();
		gameFrame.getContentPane().setLayout(new BorderLayout());
		gameFrame.getContentPane().add(new PanelButtons(),
BorderLayout.NORTH);
```

```
            gameFrame.getContentPane().add(new PanelLabel(" "),
BorderLayout.EAST);
            gameFrame.getContentPane().add(new PanelLabel(" "),
BorderLayout.WEST);
            gameFrame.getContentPane().add(new GameBoard(),
BorderLayout.CENTER);
            gameFrame.setVisible(true);
            container = gameFrame.getContentPane();
      }

      /**
       * Randomly place bombs
       */
      public static void bombPlacement() {
            int bombsPlaced = 0;
            int randX, randY = 0;
            boolean ok = false;
            do {
                  randX = randomInt(0,(settings.getAcross()-1));
                  randY = randomInt(0,(settings.getDown()-1));
                  if (!gameArray[randX][randY].isBomb()){
                        gameArray[randX][randY].setBomb(true);

incrementNeighbourCells(gameArray[randX][randY]);
                        bombsPlaced ++;
                  }
                  if (bombsPlaced == settings.getBombCount()) {
                        ok = true;
                  }
            } while(!ok);
      }

      /**
       * called from cell action listener. Handle bomb detection.
       * @param buttonString
       */
      public static void buttonCellPressed(Cell cell){
            //Test          //showCells();
            if (!cell.isOpened()){
                  if (settings.isFlag()) {
                        manageFlag(cell);
                  } else {

                        if (!cell.isFlagged()) {
                              if (cell.isBomb()) {
                                    endGame();
                                    System.out.println("bomb");
                              } else if (cell.getNeighbourMine() > 0) {

      cell.setText(String.valueOf(cell.getNeighbourMine()));
                                    cell.setOpened(true);
                                    cell.setEnabled(false);
```

```
                        } else {
                              cell.setOpened(true);
                              cell.setEnabled(false);
                              cellSearch(cell);
                        }
                  }// flagged cell do nothing
            }
            checkWinner();
      } // open cell do nothing
}

private static void checkWinner() {
      int winnerCount = 0;
      int falseCount = 0;
      for (int i = 0; i < settings.across; i++) {
            for (int j = 0; j < settings.down; j++) {
                  if (gameArray[i][j].isFlagged() &&
(!gameArray[i][j].isBomb())) {
                        falseCount ++;
                  }
                  if (gameArray[i][j].isBomb() &&
gameArray[i][j].isFlagged()){
                        winnerCount ++;
                  }
            }
      }
      if ((winnerCount == settings.getBombCount()) && (falseCount
== 0)) {
            for (int i = 0; i < settings.across; i++) {
                  for (int j = 0; j < settings.down; j++) {
                        gameArray[i][j].setText("W");
                        gameArray[i][j].setBackground(new
Color(40,30,100));

                  }
            }
      }
}

/**
 * Take in a max and a min boundary and produce a random number
 * @param min
 * @param max
 * @return
 */
public static int randomInt(int min, int max) {
      Random rnd = new Random();
      int randomNum = rnd.nextInt((max-min)+1)+min;
      return randomNum;
}

/**
```

```
       *
       * @param bombCell
       */
      public static void incrementNeighbourCells(Cell bombCell){
            ArrayList<Integer> vectorX = getAcrossVectors
                        (bombCell.getAcross(),bombCell.getDown());
            ArrayList<Integer> vectorY = getDownVectors
                        (bombCell.getAcross(),bombCell.getDown());
            if (vectorX.size() == vectorY.size()) {
                  for (int i = 0; i < vectorX.size(); i++){

      gameArray[vectorX.get(i)][vectorY.get(i)].setNeighbourMine(

      (gameArray[vectorX.get(i)][vectorY.get(i)].getNeighbourMine() +
1));
                  }
            } else {
                  System.out.println("x and y search vectors are
different size");
            }
      }

      /**
       * Used to get an ArrayList of X coordinates for neighbouring
cells
       * @param vectorX
       * @param vectorY
       * @return ArrayList
       */
      public static ArrayList<Integer> getAcrossVectors(int vectorX,int
vectorY) {
            ArrayList<Integer> xCoordinate = new ArrayList<>();
            xCoordinate.add(vectorX);
            if ((vectorX-1) >= 0){
                  //left
                  xCoordinate.add(vectorX -1);
            }
            if (((vectorX-1) >=0) && ((vectorY - 1) >=0)){
                  //top left
                  xCoordinate.add(vectorX -1);
            }
            if ((vectorY-1 >=0)){
                  //top
                  xCoordinate.add(vectorX);
            }
            if ((vectorX+1) <= (settings.across-1) && (vectorY-1) >=0){
                  //top right
                  xCoordinate.add(vectorX +1);
            }
            if ((vectorX +1) <= (settings.across-1)){
                  //right
                  xCoordinate.add(vectorX +1);
```

```
            }
            if ((vectorX +1) <= (settings.across -1) && (vectorY +1) <=
(settings.down-1)) {
                  //bottom right
                  xCoordinate.add(vectorX +1);
            }
            if ((vectorY + 1) <= (settings.down - 1)) {
                  //bottom
                  xCoordinate.add(vectorX);
            }
            if ((vectorX -1) >= 0 && (vectorY +1) <= (settings.down -
1)) {
                  //bottom left
                  xCoordinate.add(vectorX -1);
            }
            return xCoordinate;
      }

      /**
       * Used to return an array list of Y coordinates of neighbouring
cells
       * @param vectorX
       * @param vectorY
       * @return Array List
       */
      public static ArrayList<Integer> getDownVectors(int vectorX, int
vectorY){
            ArrayList<Integer> yCoordinate = new ArrayList<>();
            yCoordinate.add(vectorY);
            if ((vectorX-1) >= 0){
                  //left
                  yCoordinate.add(vectorY);
            }
            if (((vectorX-1) >=0) && ((vectorY - 1) >=0)){
                  //top left
                  yCoordinate.add(vectorY -1);
            }
            if ((vectorY-1 >=0)){
                  //top
                  yCoordinate.add(vectorY -1);
            }
            if ((vectorX+1) <= (settings.across-1) && (vectorY-1) >=0){
                  //top right
                  yCoordinate.add(vectorY -1);
            }
            if ((vectorX +1) <= (settings.across-1)){
                  //right
                  yCoordinate.add(vectorY);
            }
            if ((vectorX +1) <= (settings.across -1) && (vectorY +1) <=
(settings.down-1)) {
                  //bottom right
```

```
                yCoordinate.add(vectorY +1);
            }
            if ((vectorY + 1) <= (settings.down - 1)) {
                //bottom
                yCoordinate.add(vectorY +1);
            }
            if ((vectorX -1) >= 0 && (vectorY +1) <= (settings.down -
1)) {
                //bottom left
                yCoordinate.add(vectorY +1);
            }
            return yCoordinate;
      }

      /*
       * use x and y vectors and get cells to run search on,
       */
      private static void cellSearch(Cell searchCell) {
            ArrayList<Integer> vectorX = getAcrossVectors
                        (searchCell.getAcross(),searchCell.getDown());
            ArrayList<Integer> vectorY = getDownVectors
                        (searchCell.getAcross(),searchCell.getDown());
            if (vectorX.size() == vectorY.size()) {
                for (int i = 0; i < vectorX.size(); i++){

      buttonCellPressed(gameArray[vectorX.get(i)][vectorY.get(i)]);
                }
            } else {
                System.out.println("x and y search vectors are
different size");
            }

      }
      /**
       * used to show all cell neighbour and bomb values
       */
      @Test
      private void showCells(){
            for (int i = 0;i < settings.getAcross(); i++){
                for (int j = 0; j < settings.getDown(); j++) {

      gameArray[i][j].setText(String.valueOf(gameArray[i][j].getNeighbo
urMine()));
                        if (gameArray[i][j].isBomb()){
                              gameArray[i][j].setText("B");
                        }
                }
            }
      }

      /**
       * stop cells being able to be used. to end the game.
```

```
	 */
	private static void endGame() {
		for (int i = 0;i < settings.getAcross(); i++){
			for (int j = 0; j < settings.getDown(); j++) {
				gameArray[i][j].setEnabled(false);
				if(gameArray[i][j].isBomb()){
					gameArray[i][j].setBackground(new
Color(150,35,75));
					gameArray[i][j].setText("B");

				}
			}
		}
	}

	/**
	 * Allow user to change the settings
	 */
	public static void showOptions() {
		new PanelOptions();
	}

	/**
	 * flag and un-flag cells
	 */
	public static void manageFlag(Cell cell) {
		cell.setFlagged(!cell.isFlagged());
		if (cell.isFlagged()){
			cell.setText("F");
		} else {
			cell.setText(" ");
		}
	}
}
```