MineField 2.0

Time required: 120 minutes

Comment each line of code.

Open Minefield 1.0. This project will finish the program.

Requirements

- Add the following import statements to put the icon in the program control box.

```
import java.awt.image.BufferedImage;  // Needed for the Jframe icon image
import javax.imageio.ImageIO;  // Get the image from a file
import java.io.IOException;   // handles exception if the image can't be loaded
```

- Place minefield.png in the program folder.

- Use the following code in the public static void main to place the image in the program icon.

```
// Read the image that will be used as the application icon.
// Using "/" in front of the image file name will locate the
// image at the root folder of our application. If you don't
// use a "/" then the image file should be on the same folder
// with your class file.
 BufferedImage image = null;
  try {
      image = ImageIO.read(frame.getClass().getResource("/minefield.png"));
  } catch (IOException e) {
      e.printStackTrace();
}
frame.setIconImage(image);
frame.setVisible(true);
```

- The class should implement MouseListener and ActionListener

- Place the following code at the top of the program to create a random number pool for the mine.

    - **private Random randomMine = new Random();**
- Track the following
    - Games Won, Total Games, Mines Clicked.
- Create a variable to hold the panel number of the mine.
- Create a mine label.
- Add ActionListeners to each of the menu items.
- When you are adding the individual panels to the gameboard, add a mouse listener to each panel.
  ```
  panel[x].addMouseListener(this);
  ```

-
  ```
  public void actionPerformed(ActionEvent event){
          // Get a reference to the object that was clicked
          Object source = event.getSource();
  ```
- Inside this method, determine which radio button is chosen to set the WIN variable, the number of mines before you win. Call the newGame method. Also add the action for the exit, New Game and about menu.
- The New Game method resets the global variables, then calls the playAgain function.
- The PlayAgain method updates the status bar, generates a new mine, and resets the gameBoard. Remove the MouseListener, Add the mouseListener and reset the color for each panel. Call validate() and repaint() to update everything.

Add the following code for the mouseClicked event.

```java
/**********************************************************************
* Main decision point, what panel did we click?
*/
public void mouseClicked(MouseEvent e){
   minesClicked++;                // Track how many times we have clicked
   Object source = e.getSource(); // Get the pane object we clicked

   // Go through each panel, find the one we clicked
   for(int x = 0; x < NUM; ++x){
      if(source == panel[x]){ // We clicked this panel

         if(minesClicked == WIN){ // We clicked WIN amount of times

            // We hit the mine and lost
            if(mine == x){
               panel[x].setBackground(Color.RED);// Set the mine panel red
               panel[x].add(mineLabel);          // Add the word * MINE * to the label
               panel[x].validate();              // Re layout the component
               repaint();                        // Redraw the program to ensure we see everything

               // Allow the user to choose to play again, or exit the game
               int answer = JOptionPane.showConfirmDialog(null, "You hit the mine, sorry you lost.\nPlay again?",
               "Play again?", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
               // If the user clicks Yes
               if(answer == JOptionPane.YES_OPTION){
                  totalGames++;                  // Track number of games played
                  panel[x].remove(mineLabel);    // Remove the mine label
                  panel[x].validate();           // Re layout the component
                  playAgain();                   // Reset the gameboard
               }else
                  System.exit(0); // Quit the program
            }

            // We missed the mine and won
            else{
               panel[x].setBackground(Color.WHITE); // Set the panel white that we clicked
               repaint();  // Redraws the screen to make sure we see everything

               // Allow the user to choose to play again, or exit the game
               int answer = JOptionPane.showConfirmDialog(null, "You won!\nPlay again?", "Play again?",
               JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
               // If the user clicks Yes
               if(answer == JOptionPane.YES_OPTION){
                  totalGames++;  // Increment the number of games played
                  gamesWon++;    // Increment the number of games won
                  playAgain();   // Reset the gamboard
               }
               else
                  System.exit(0); // Quit the program
            }
         }
```

```java
               // We missed the mine and won
               else{
                  panel[x].setBackground(Color.WHITE); // Set the panel white that we clicked
                  repaint();  // Redraws the screen to make sure we see everything

                  // Allow the user to choose to play again, or exit the game
                  int answer = JOptionPane.showConfirmDialog(null, "You won!\nPlay again?", "Play again?",
                  JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
                  // If the user clicks Yes
                  if(answer == JOptionPane.YES_OPTION){
                     totalGames++;  // Increment the number of games played
                     gamesWon++;    // Increment the number of games won
                     playAgain();   // Reset the gamboard
                  }
                  else
                     System.exit(0); // Quit the program
               }
            }

            // We hit the mine and lost
            else if(mine == x){
               panel[x].setBackground(Color.RED);  // Set the mine panel red
               panel[x].add(mineLabel);            // Add the  word * MINE * to the label
               panel[x].validate();                // Re layout the component
               repaint();                          // Redraw the program to ensure we see everything

               // Ask the the user to choose to play again, or exit  the game
               int answer = JOptionPane.showConfirmDialog(null, "You hit the mine, sorry you lost.\nPlay again?",
               "Play again?", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
               // If the user clicks Yes
               if(answer == JOptionPane.YES_OPTION){
                  totalGames++;                  // Track number of games played
                  panel[x].remove(mineLabel);    // Remove the mine label
                  panel[x].validate();           // Re layout the component
                  playAgain();                   // Reset the gameboard
               }else
                  System.exit(0); // Quit the program
            }

            // We didn't win or hit the mine, turn that panel white
            else{
               panel[x].setBackground(Color.WHITE);   // Turn the selected panel white
               repaint();                             // Redraws the screen to make sure we see everything
               panel[x].removeMouseListener(this);    // Remove the ability to click the panel
            }
         }
      }
   }
```

HitRate.java

HitRate.java should be saved in the same folder as the program. To use this code in the program: HitRate.Calculate(gamesWon, totalGames)

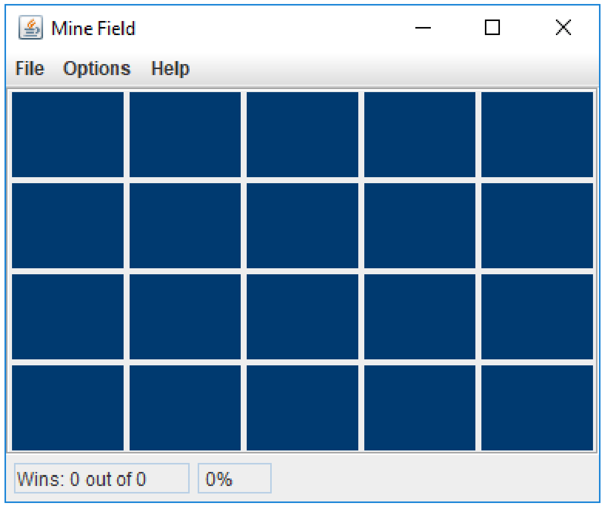
```java
import java.text.NumberFormat;

public class HitRate{

   // Create a percent number format, converts number to string
   private static final NumberFormat nf = NumberFormat.getPercentInstance();

   public static String Calculate(int Wins, int Games){
      // Create variable to hold HitRate result
      double dblPercent;

      // Convert integers to doubles to get the correct answer
      dblPercent = ((double) Wins) / ((double) Games);

      return nf.format(dblPercent);
   }
}
```

Submission

1. Test your finished project. Make corrections as necessary.

2. Create a Jar file from the project.

3. Zip up the project folder and the Jar. Submit it to Blackboard.



Wins: 0 out of 0 0%