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.
  ```

```
75    /*************************************************************************
76    * Main decision point, what panel did we click?
77    */
78    public void mouseClicked(MouseEvent e){
79       minesClicked++;                // Track how many times we have clicked
80       Object source = e.getSource(); // Get the pane object we clicked
81
82       // Go through each panel, find the one we clicked
83       for(int x = 0; x < NUM; ++x){
84          if(source == panel[x]){ // We clicked this panel
85
86             if(minesClicked == WIN){ // We clicked WIN amount of times
87
88                // We hit the mine and lost
89                if(mine == x){
90                   panel[x].setBackground(Color.RED);// Set the mine panel red
91                   panel[x].add(mineLabel);          // Add the word * MINE * to the label
92                   panel[x].validate();              // Re layout the component
93                   repaint();                        // Redraw the program to ensure we see everything
94
95                   // Allow the user to choose to play again, or exit the game
96                   int answer = JOptionPane.showConfirmDialog(null, "You hit the mine, sorry you lost.\nPlay again?",
97                   "Play again?", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
98                   // If the user clicks Yes
99                   if(answer == JOptionPane.YES_OPTION){
100                     totalGames++;                   // Track number of games played
101                     panel[x].remove(mineLabel);     // Remove the mine label
102                     panel[x].validate();            // Re layout the component
103                     playAgain();                    // Reset the gameboard
104                  }else
105                     System.exit(0); // Quit the program
106               }
107
108               // We missed the mine and won
109               else{
110                  panel[x].setBackground(Color.WHITE); // Set the panel white that we clicked
111                  repaint();  // Redraws the screen to make sure we see everything
112
113                  // Allow the user to choose to play again, or exit the game
114                  int answer = JOptionPane.showConfirmDialog(null, "You won!\nPlay again?", "Play again?",
115                  JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
116                  // If the user clicks Yes
117                  if(answer == JOptionPane.YES_OPTION){
118                     totalGames++;  // Increment the number of games played
119                     gamesWon++;    // Increment the number of games won
120                     playAgain();   // Reset the gamboard
121                  }
122                  else
123                     System.exit(0); // Quit the program
124               }
125            }
126
```

```
107                 // We missed the mine and won
108                 else{
109                     panel[x].setBackground(Color.WHITE); // Set the panel white that we clicked
110                     repaint();  // Redraws the screen to make sure we see everything
111 
112                     // Allow the user to choose to play again, or exit the game
113                     int answer = JOptionPane.showConfirmDialog(null, "You won!\nPlay again?", "Play again?",
114                     JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
115                     // If the user clicks Yes
116                     if(answer == JOptionPane.YES_OPTION){
117                         totalGames++;  // Increment the number of games played
118                         gamesWon++;    // Increment the number of games won
119                         playAgain();   // Reset the gamboard
120                     }
121                     else
122                         System.exit(0); // Quit the program
123                 }
124             }
125 
126             // We hit the mine and lost
127             else if(mine == x){
128                 panel[x].setBackground(Color.RED);  // Set the mine panel red
129                 panel[x].add(mineLabel);            // Add the  word * MINE * to the label
130                 panel[x].validate();                // Re layout the component
131                 repaint();                          // Redraw the program to ensure we see everything
132 
133                 // Ask the the user to choose to play again, or exit  the game
134                 int answer = JOptionPane.showConfirmDialog(null, "You hit the mine, sorry you lost.\nPlay again?",
135                 "Play again?", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
136                 // If the user clicks Yes
137                 if(answer == JOptionPane.YES_OPTION){
138                     totalGames++;                   // Track number of games played
139                     panel[x].remove(mineLabel);     // Remove the mine label
140                     panel[x].validate();            // Re layout the component
141                     playAgain();                    // Reset the gameboard
142                 }else
143                     System.exit(0); // Quit the program
144             }
145 
146             // We didn't win or hit the mine, turn that panel white
147             else{
148                 panel[x].setBackground(Color.WHITE);    // Turn the selected panel white
149                 repaint();                              // Redraws the screen to make sure we see everything
150                 panel[x].removeMouseListener(this);     // Remove the ability to click the panel
151             }
152         }
153     }
154 }
```

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)

```
 8 import java.text.NumberFormat;
 9
10 public class HitRate{
11
12    // Create a percent number format, converts number to string
13    private static final NumberFormat nf = NumberFormat.getPercentInstance();
14
15    public static String Calculate(int Wins, int Games){
16       // Create variable to hold HitRate result
17       double dblPercent;
18
19       // Convert integers to doubles to get the correct answer
20       dblPercent = ((double) Wins) / ((double) Games);
21
22       return nf.format(dblPercent);
23    }
24 }
```

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.