```
/***************************************************************
 * GUI front end to the game of Pig
 *
 * NOTE: The bad conventions in this class are not my fault
 *      -Silas Agnew
 *
 * @author Scott Grissom, Silas Agnew
 * @version October 24, 2017
 **************************************************************/
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class PigGUI extends JFrame implements ActionListener {

    /** visual representation of the dice */
    GVdie d1, d2;

    /** buttons and labels */
    JButton roll, hold, compButton;
    JLabel round, player, computer;
    PigGame game;

    /** menu items */
    JMenuBar menus;
    JMenu fileMenu;
    JMenuItem quitItem;
    JMenuItem playItem;
    JMenuItem restartItem;

/****************************************************************
Create all elements and display within the GUI
***************************************************************/
 public static void main(String args[]) {
     PigGUI gui = new PigGUI();
     gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     gui.setTitle("Game of Pig");
     gui.pack();
     gui.setVisible(true);
 }

/****************************************************************
GUI constructor
***************************************************************/
public PigGUI(){
    // Create the game object as well as the GUI Frame
    game = new PigGame();
    setBackground(Color.CYAN);

    // Use a GridBagLayout
    setLayout(new GridBagLayout());
    GridBagConstraints panelPosition = new GridBagConstraints();

    // Create the buttons
    roll = new JButton("Roll");
    hold = new JButton("Hold");
    compButton = new JButton("Computer");
    compButton.setEnabled(false);

    // Register the listeners for the three buttons
    roll.addActionListener(this);
    hold.addActionListener(this);
    compButton.addActionListener(this);

    // Place both dice in the middle row
    d1 = game.getDie(1);
    panelPosition.gridx = 0;
    panelPosition.gridy = 1;
    add(d1, panelPosition);

    d2 = game.getDie(2);
    panelPosition.gridx = 2;
    panelPosition.gridy = 1;
    add(d2, panelPosition);

    // Create the labels
    round = new JLabel ("Round: 0");
    player = new JLabel ("Player: 0");
    computer = new JLabel ("Computer: 0");
    player.setForeground(Color.red);

    // Place labels along the top
    panelPosition.gridx = 0;
    panelPosition.gridy = 0;
    add(player, panelPosition);
    panelPosition.gridx = 1;
    panelPosition.gridy = 0;
    add(round,panelPosition);
    panelPosition.gridx = 2;
    panelPosition.gridy = 0;
    add(computer, panelPosition);

    // Place buttons along the bottom
    panelPosition.gridx = 0;
    panelPosition.gridy = 2;
    add(roll,panelPosition);
    panelPosition.gridx = 1;
    panelPosition.gridy = 2;
    add(hold, panelPosition);

    // Place computer button below second die
    panelPosition.gridx = 2;
    panelPosition.gridy = 2;
    add(compButton, panelPosition);

    // Set up file menus
    setupMenus();
}

/*****************************************************************
Respond to the user action

```

```
@param e - the JComponent just selected
*****************************************************************/
public void actionPerformed(ActionEvent e){

    // what did the user just select?
    JComponent buttonPressed = (JComponent) e.getSource();

    // quit the game
    if (buttonPressed == quitItem) {
        System.exit(1);
    }

    // start a new game
    if (buttonPressed == restartItem) {
        game.restart();
    }

    // start a new game
    if (buttonPressed == playItem) {
        playAutoGame();
    }

    // check if player rolls
    if (buttonPressed == roll) {
        game.playerRolls();
    }

    // check if player holds
    if (buttonPressed == hold) {
        game.playerHolds();
    }

    // check if computer's turn
    if (buttonPressed == compButton) {
        game.computerTurn();
    }

    // update text colors and disable buttons as needed
    if (game.isPlayerTurn()) {
        compButton.setEnabled(false);
        roll.setEnabled(true);
        hold.setEnabled(true);
        player.setForeground(Color.red);
        computer.setForeground(Color.black);
    } else {
        compButton.setEnabled(true);
        roll.setEnabled(false);
        hold.setEnabled(false);
        player.setForeground(Color.black);
        computer.setForeground(Color.red);
    }

    // update the three score labels
    round.setText(game.getCurrentRoundScore() + "");
    player.setText(game.getPlayerScore() + "");
    computer.setText(game.getComputerScore() + "");

    // display winning message if player or computer wins
    if (game.playerWon()) {
        JOptionPane.showMessageDialog(this, "Player Won!");
    } else if (game.computerWon()) {
        JOptionPane.showMessageDialog(this, "CPU Won!");
    }
}

/*****************************************************************
 * Play one auto game
*****************************************************************/
private void playAutoGame(){
   game.restart();
   game.autoGame();
}

/*****************************************************************
Set up the menu items
*****************************************************************/
private void setupMenus(){
    fileMenu = new JMenu("File");
    quitItem = new JMenuItem("Quit");
    playItem = new JMenuItem("Auto Play");
    restartItem = new JMenuItem("Restart");
    quitItem.addActionListener(this);
    restartItem.addActionListener(this);
    playItem.addActionListener(this);
    fileMenu.add(restartItem);
    fileMenu.add(playItem);
    fileMenu.add(quitItem);
    menus = new JMenuBar();
    menus.add(fileMenu);
    setJMenuBar(menus);
}
}
```