# Project Report

**Group members:**

- Charlene Khun
- Minh Nhat Duong
- Arnav Amin
- Benjamin Man
- Arpon Purkayastha

**Class:** CS-170-01

**Assignment:** CS 170 Fall Project

**Submission Date:** 10:00pm, Wed (12/8), but asked for extension until Fri (12/10)

**Project title:** AlphabetLearner

## Individual Member Contributions

**Minh:**

- Created a flowchart showing step-by-step of how the project is supposed to work.
- Created an initial design to demonstrate what the program should look like.
- Figured out how to make a program window to display at the center of the screen. Initially, we needed to drag the window to the center. This line of code was added to fix the issue: "*frame.setLocationRelativeTo(null);*" It needed to be inserted before we told the application window to open.
- Collaborated with **Charlene** to test out and design the MainFrame, HowToPlay and Game files. Also, provided some ideas, such as the "thinking emoji image" in the HowToPlay file.
- Initialized the text file so that all player's scores are 0.
- Looked up some pictures and music, Sent it to **Charlene** so she could import it into the code.
- Wrote and recorded all 3 Group Meeting Logs. Also wrote the cover page of the report.

**Arnav:**

- Implemented 5 image / word questions. Found and cited the images.
- Provided insight on how certain parts of the program should function to produce the results that we were looking for.
- Took the code from **Charlene** and implemented a live counter in the game, which allows the user to keep track of how many points they scored over the course of the game.
- Wrote out all the comments for all files that were created. These files included: MainFrame.java, HowToPlay.java, Game.java, and LeaderBoard.java.

- Wrote program description for the HowToPlay.java file and functions for the lab report.
- Coordinated with **Arpon** and **Benjamin** to finish up the leaderboard and counter. Also communicated with them to split up work on the program descriptions for the lab report.
- Coordinated with team members to efficiently divide up the project so that all people have equal workload.

**Arpon:**

- Built out the UI(UserInterface) of Leaderboard.java using WindowBuilder
- Developed the logic of Leaderboard.java and made assisting files such as scores.txt to support application
- Coordinated with **Charlene** to integrate Leaderboard with the context of the game
- Fixed logic bugs in Leaderboard.java and constructed logic to persist game data through opens and closes of the application
- Found suitable audio files for successful and unsuccessful user answers to play during runtime
- Cooperated in creating the structure of the overall game
- Wrote out program descriptions for Leaderboard.java and MainFrame.java (report).

**Charlene:**

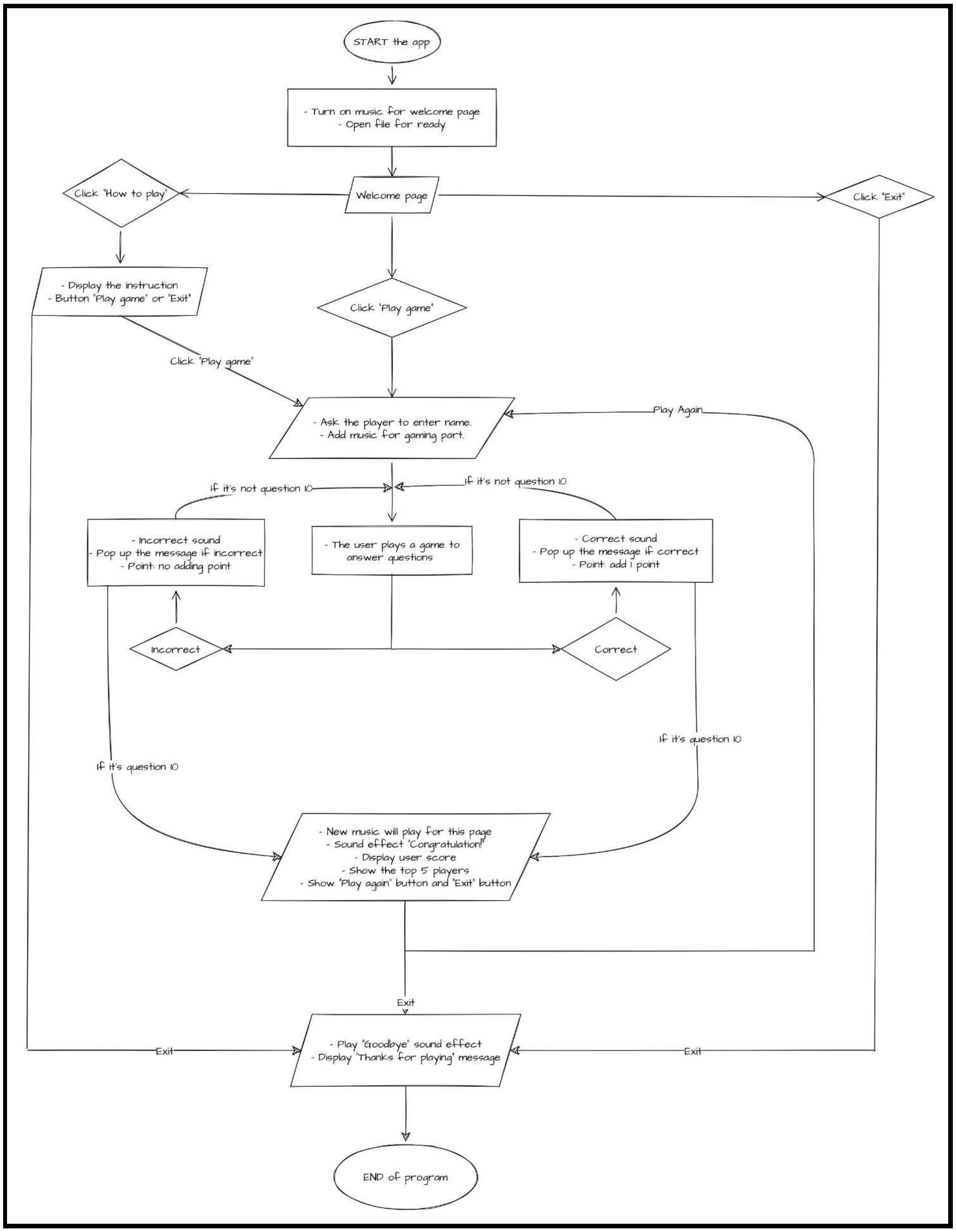
- Took the lead in forming the group and creating a Discord group chat. Organized all group meetings.
- Looked up how to install WindowBuilder, how to link a Jframe to another Jframe, and how to insert music/ insert sound effects. Shared the information with the rest of the group.

- Figured out how to change the dimensions of JOptionDialogBoxes and customize the font / font size
- Worked on the initial drafts of the project. Collaborated with **Minh** to complete the MainFrame, HowToPlay, and Game files.
- Expanded **Arpon's** initial leaderboard code by creating an algorithm to sort the leaderboard. Attempted to write the sorted leaderboard to a file. However, **he** made the final adjustments.
- Helped **Minh** pick out good images to add to the project.
- Figured out how to randomize the questions instead of just displaying 10 questions in order.
- Made finishing touches on the report

**Benjamin**

- Implemented 5 image / word questions. Found and cited the images.
- Helped devise coding strategy to implement leaderboard
- Debugged issues with text displaying
- Implemented text name validation to ensure name is not blank and does not include spaces
- Wrote the project description for Game.java (report).
- Took screenshots for game
- Built an early prototype of title layout. This served as inspiration and an example when building the final product of the Welcome page.

## Flowchart


START the app
- Turn on music for welcome page
- Open file for ready
Welcome page
Click "How to play"
Click "Exit"
- Display the instruction
- Button "Play game" or "Exit"
Click "Play game"
Click "Play game"
- Ask the player to enter name.
- Add music for gaming part.
Play Again
If it's not question 10
If it's not question 10
- Incorrect sound
- Pop up the message if incorrect
- Point: no adding point
- The user plays a game to answer questions
- Correct sound
- Pop up the message if correct
- Point: add 1 point
Incorrect
Correct
If it's question 10
If it's question 10
- New music will play for this page
- Sound effect "Congratulation!"
- Display user score
- Show the top 5 players
- Show "Play again" button and "Exit" button
Exit
Exit
Exit
- Play "Goodbye" sound effect
- Display "Thanks for playing" message
END of program

**NOTE:** **A PDF version of the flowchart is in the project zip file.**

## Initial Design

**Figure 1 - Welcome Page**

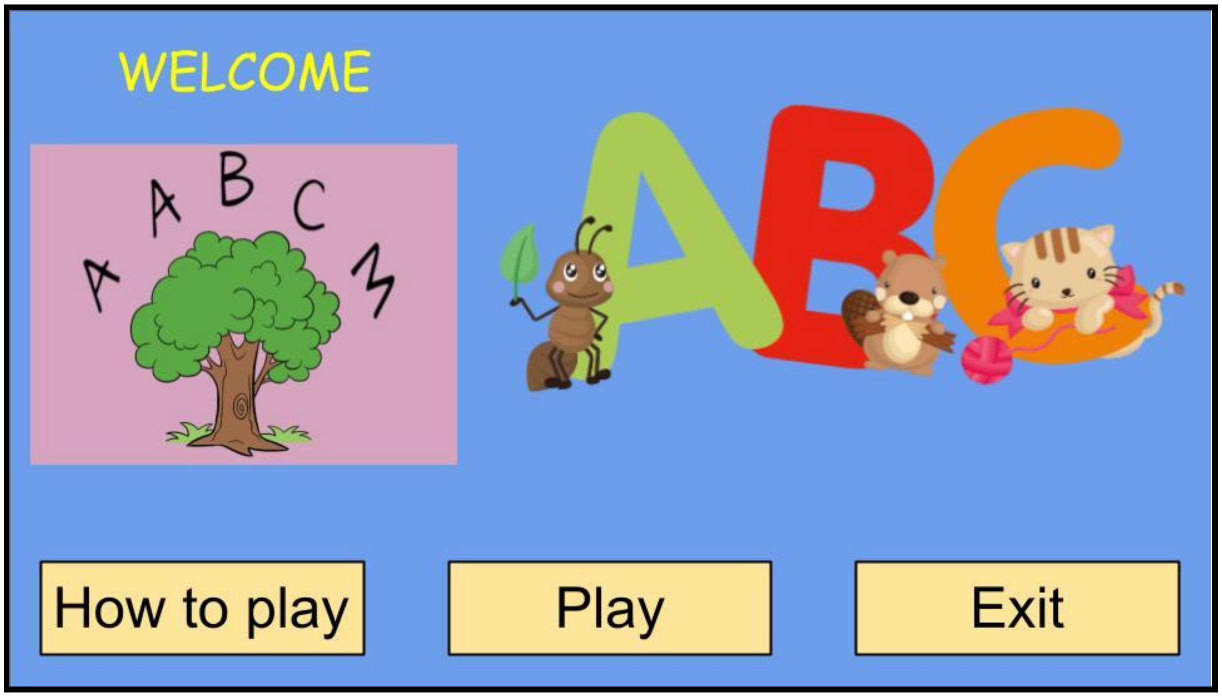


**Figure 2 - Game page**

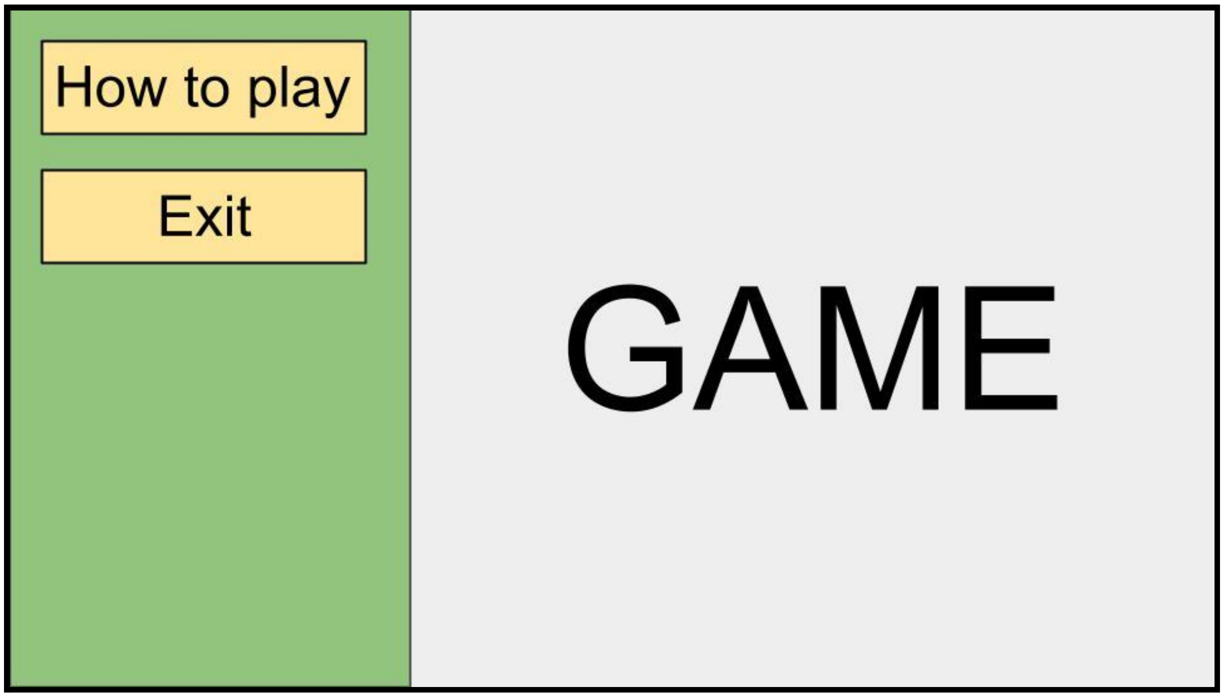


**Figure 3 - Display Score/ Leaderboard Page**

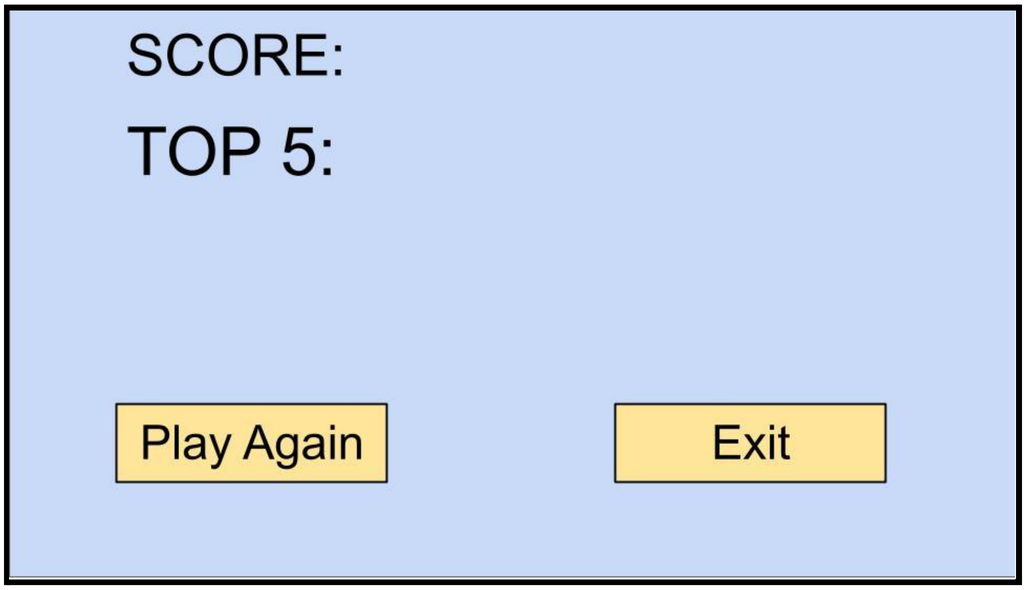
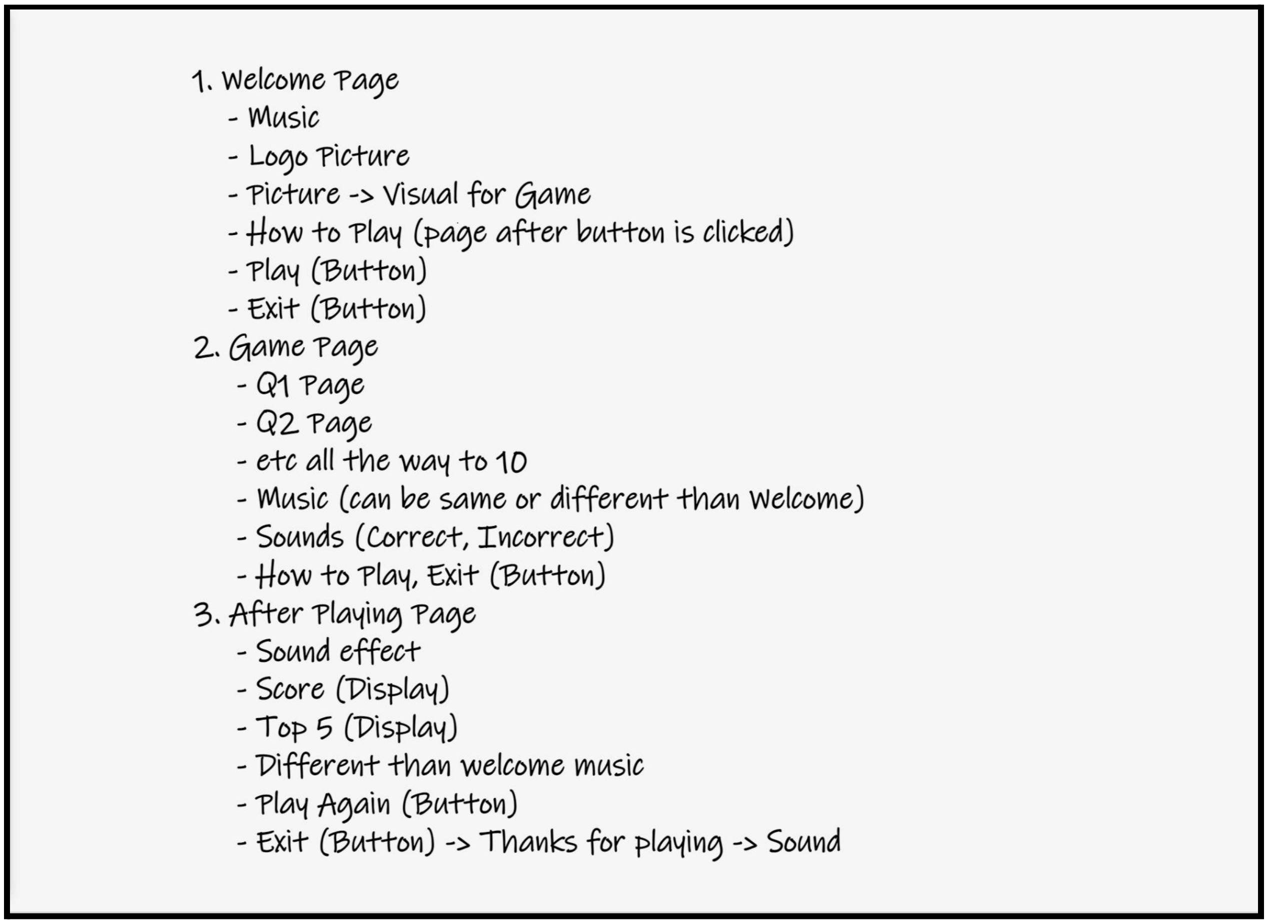


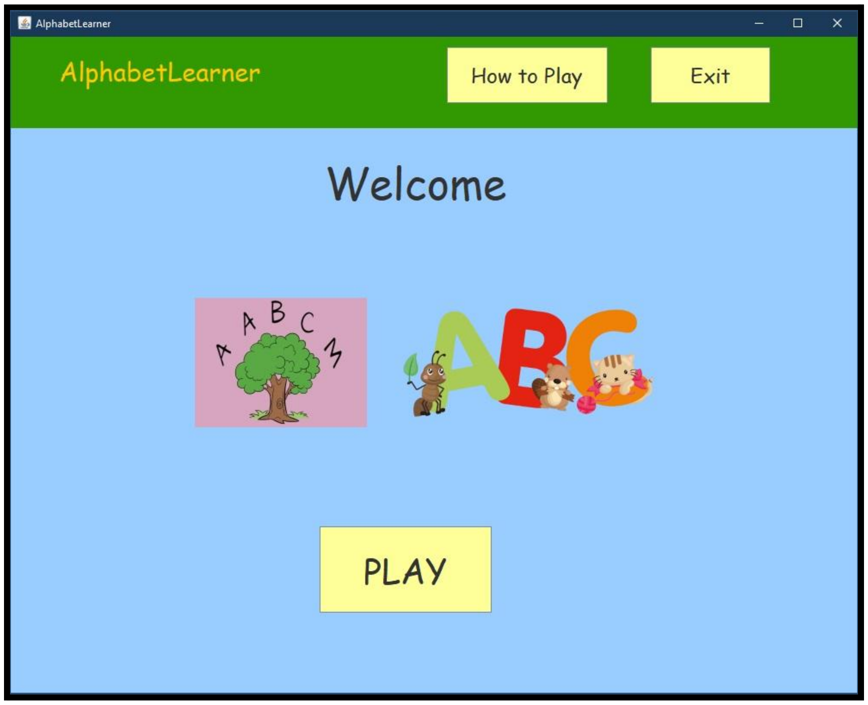
**Figure 4- Project Brainstorm**



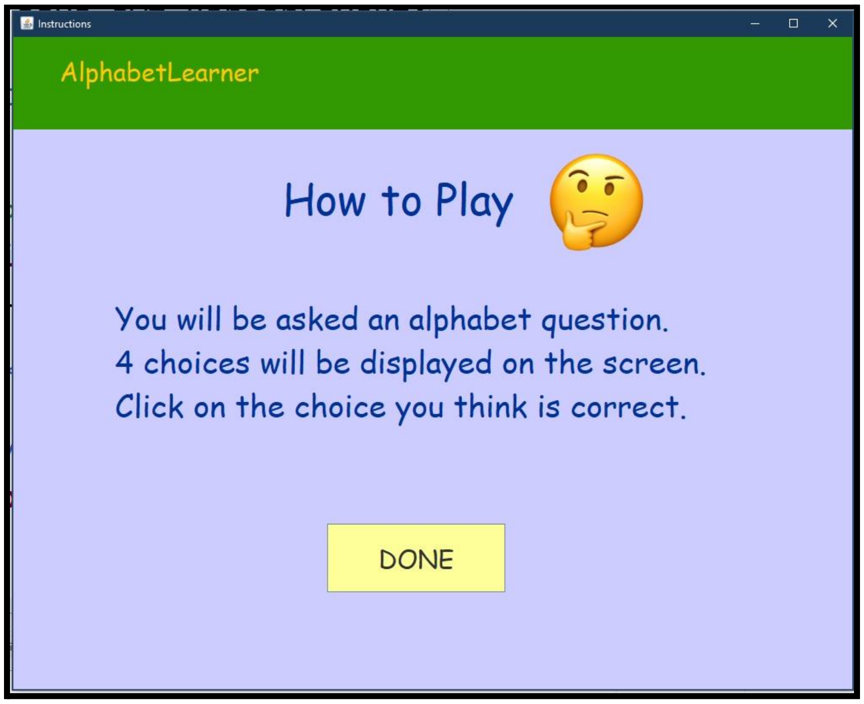
1. Welcome Page
   - Music
   - Logo Picture
   - Picture -> Visual for Game
   - How to Play (page after button is clicked)
   - Play (Button)
   - Exit (Button)
2. Game Page
   - Q1 Page
   - Q2 Page
   - etc all the way to 10
   - Music (can be same or different than Welcome)
   - Sounds (Correct, Incorrect)
   - How to Play, Exit (Button)
3. After Playing Page
   - Sound effect
   - Score (Display)
   - Top 5 (Display)
   - Different than welcome music
   - Play Again (Button)
   - Exit (Button) -> Thanks for playing -> Sound

## Final Product

### Welcome Page

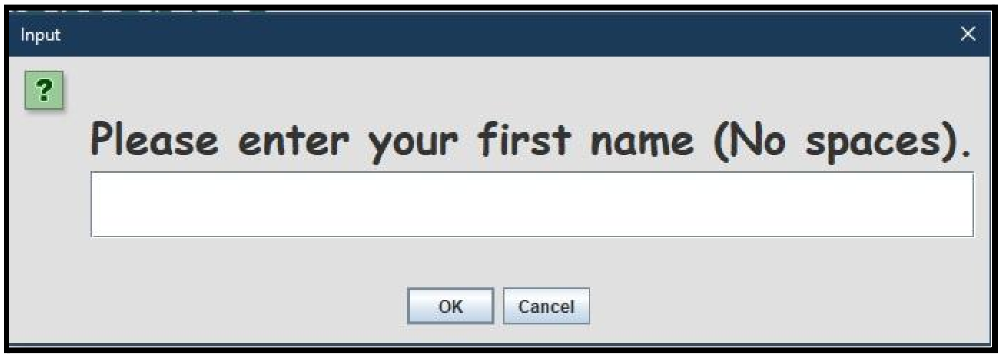


### How To Play Page



You will be asked an alphabet question.
4 choices will be displayed on the screen.
Click on the choice you think is correct.

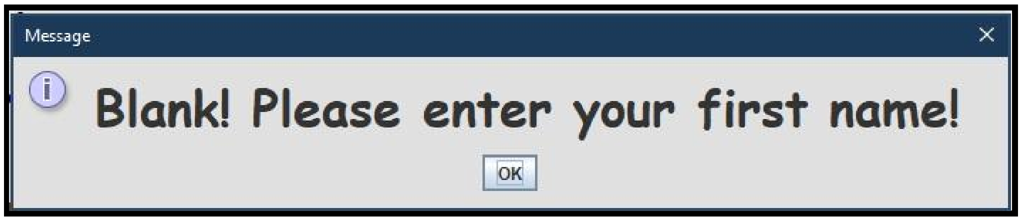
**User input (JOptionPane)**


Input

Please enter your first name (No spaces).

- What happens when user enters nothing

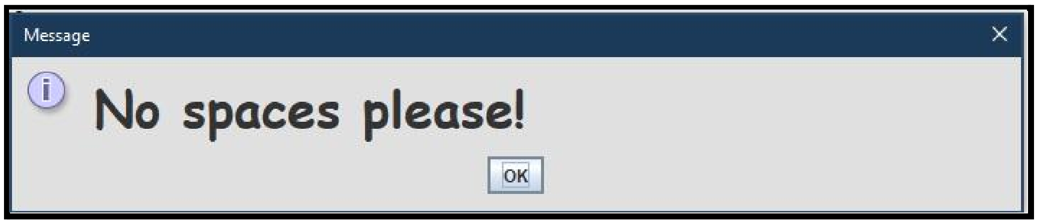
**Figure 1A**


Message

Blank! Please enter your first name!

- What happens when user enters a space

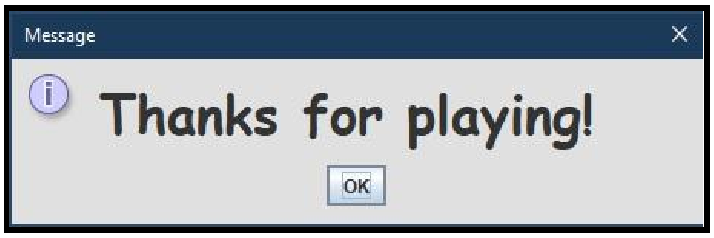
**Figure 1B**


Message

No spaces please!

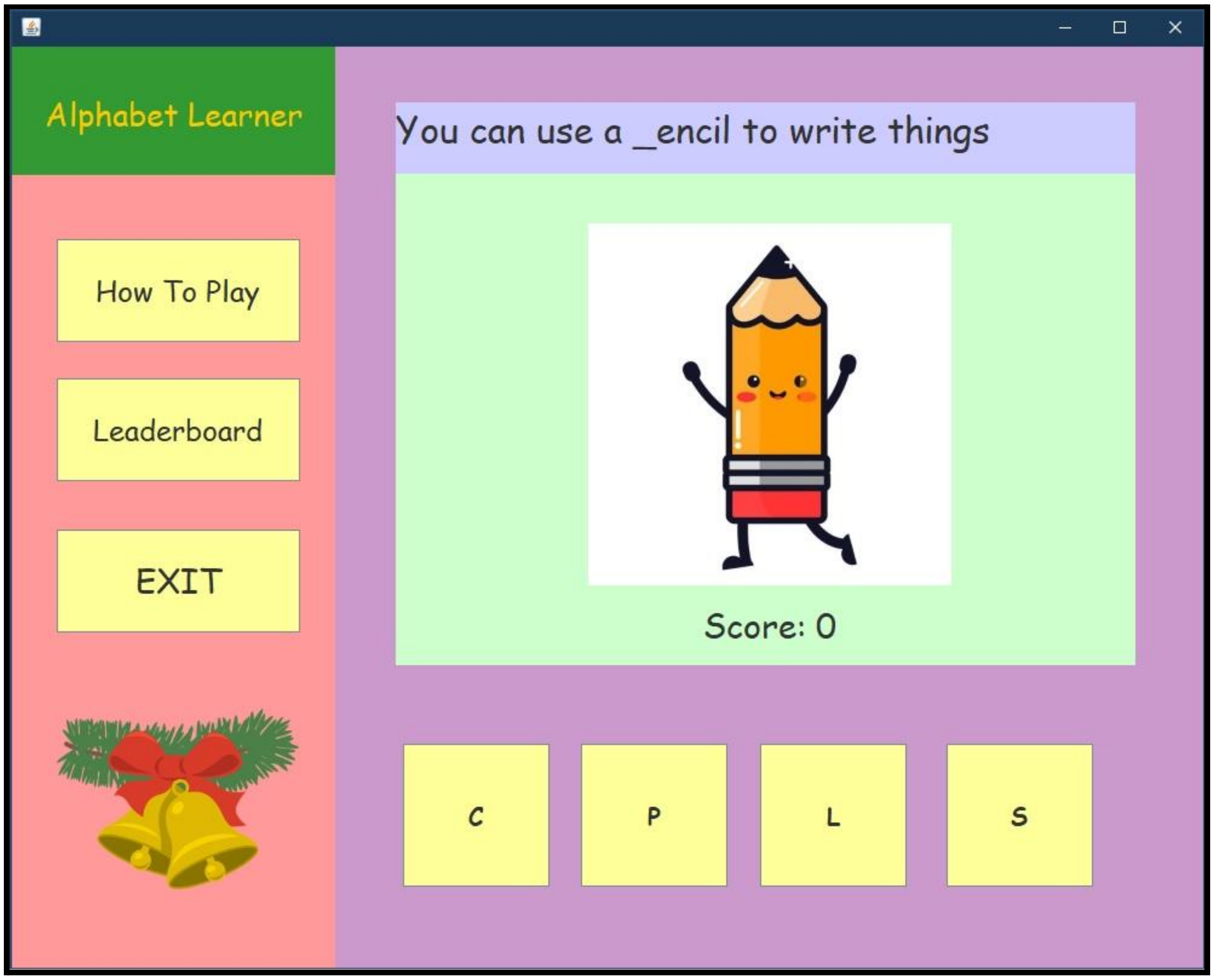
**Thank you (JOptionPane)**

- Any time the user clicks on the “EXIT” BUTTON


Message

Thanks for playing!

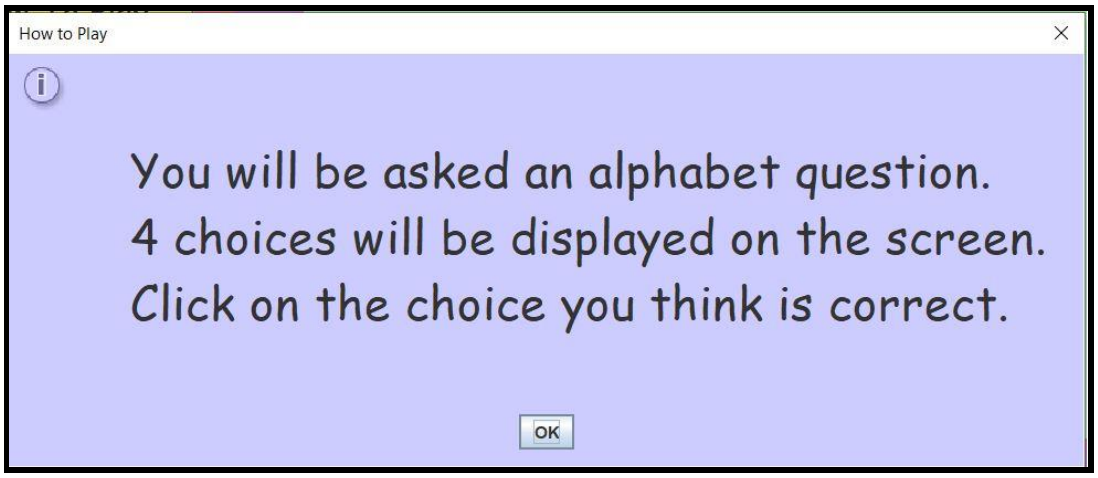
**Game Page**



You can use a _encil to write things

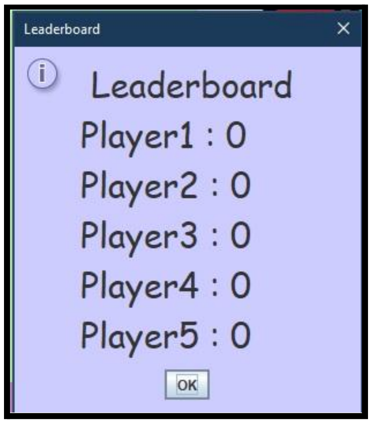
Score: 0

**How To Play JOptionPane Dialog Box**



You will be asked an alphabet question.
4 choices will be displayed on the screen.
Click on the choice you think is correct.

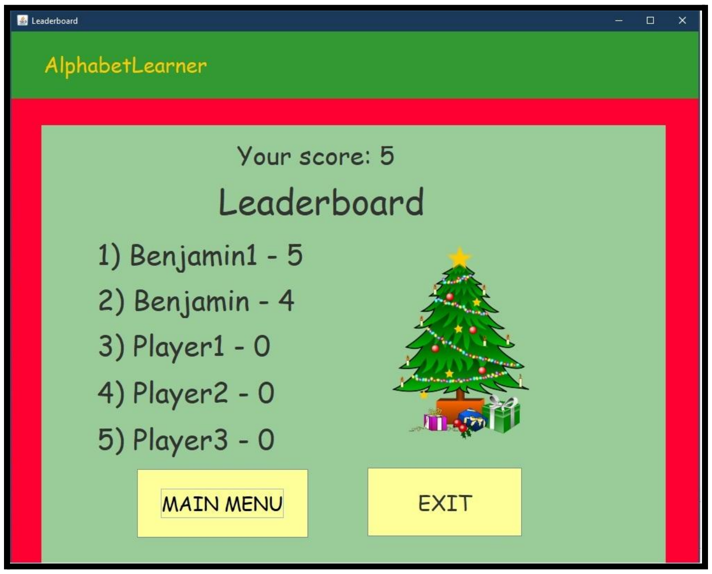
**LeaderBoard JOptionPane Dialog**


Leaderboard

Leaderboard
Player1 : 0
Player2 : 0
Player3 : 0
Player4 : 0
Player5 : 0

**LeaderBoard Frame**

// After playing the game twice


Leaderboard

AlphabetLearner

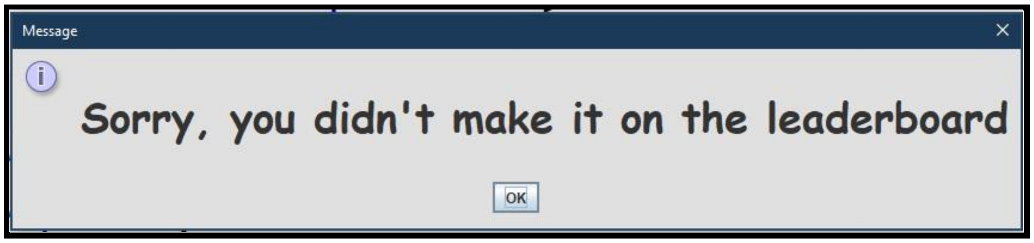
Your score: 5

Leaderboard

1) Benjamin1 - 5
2) Benjamin - 4
3) Player1 - 0
4) Player2 - 0
5) Player3 - 0

**LeaderBoard JOption Pane**

- If user does not make it on to the leaderboard

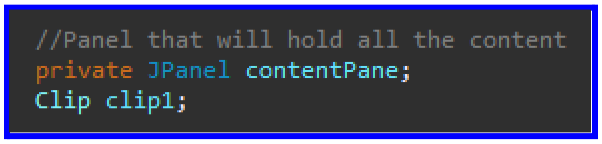

Message

Sorry, you didn't make it on the leaderboard

## Program Description

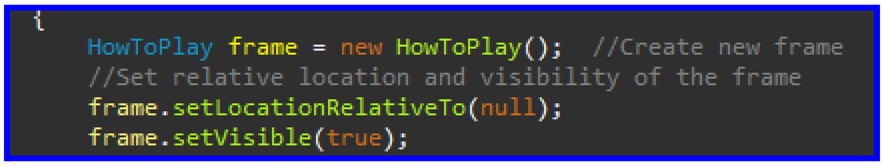
### HowToPlay.java

The file begins with declarations of the Pane and a Clip variable which will have the game sound:



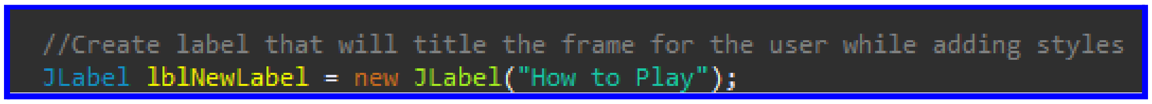
```
//Panel that will hold all the content
private JPanel contentPane;
Clip clip1;
```

Once these are declared, we start with the main method. Here, we can run the application. We create the HowToPlay frame and set its relative location. We want the relative location to be null so that window will be in the center of the screen. We also set visibility to true because we want the frame to be displayed.



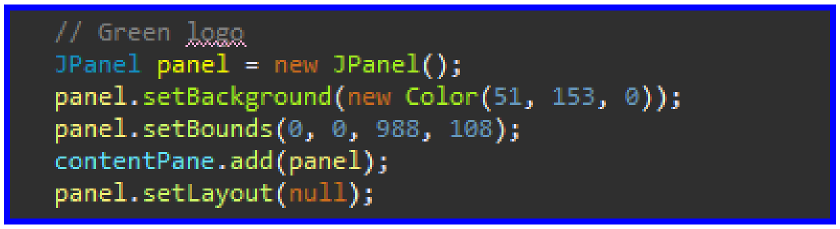
```
{
    HowToPlay frame = new HowToPlay();  //Create new frame
    //Set relative location and visibility of the frame
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
```

Then, we must define the function public HowToPlay to actually add functions to the frame. We set the title of the window to “Instructions”, set the bounds of the page, and give the frame some color in the background and foreground. Then, we can create a label that displays the title of the page.



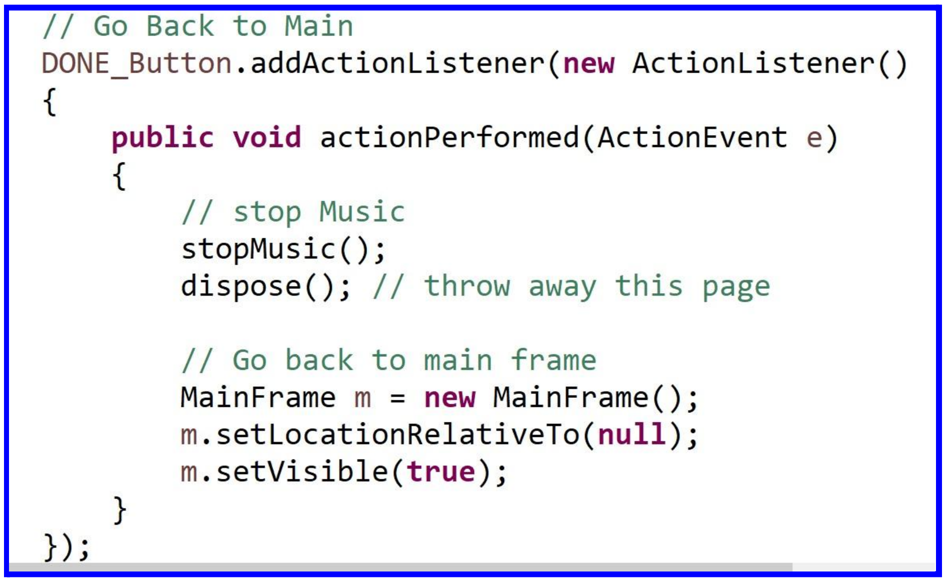
```
//Create label that will title the frame for the user while adding styles
JLabel lblNewLabel = new JLabel("How to Play");
```

After adding some color and formatting the label, we can move on to the green panel at the of the page. This panel adds a bit of color and separation for us to include the name of the game to the page.

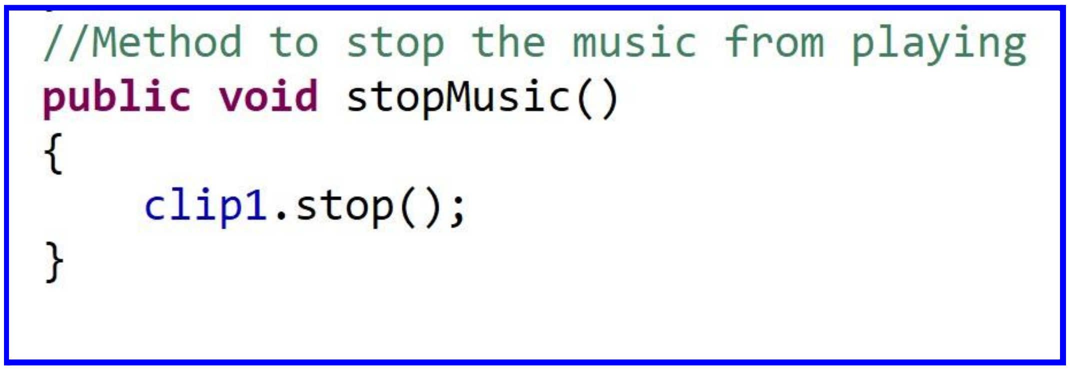


```
// Green logo
JPanel panel = new JPanel();
panel.setBackground(new Color(51, 153, 0));
panel.setBounds(0, 0, 988, 108);
contentPane.add(panel);
panel.setLayout(null);
```

All that's left in this is to actually add the instructions. This is done using a text area and manually typing in the instructions. As another visual flair, we also added an image to accompany the instructions. Finally, we can add final touches by adding in the game music and "DONE" button that will return the user back to the main frame. The music is then stopped using a function called stopMusic when the page is exited.



```
// Go Back to Main
DONE_Button.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        // stop Music
        stopMusic();
        dispose(); // throw away this page

        // Go back to main frame
        MainFrame m = new MainFrame();
        m.setLocationRelativeTo(null);
        m.setVisible(true);
    }
});
```

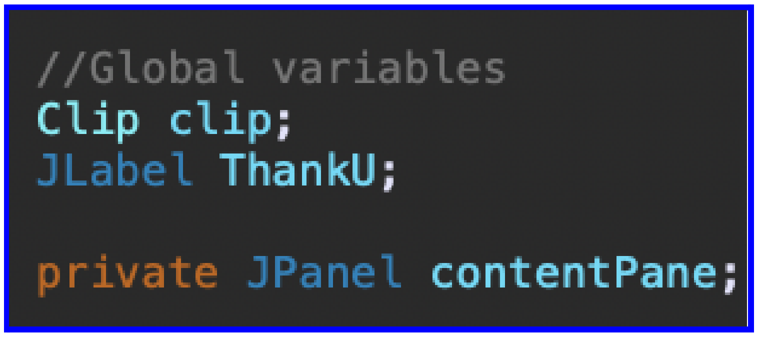


```
//Method to stop the music from playing
public void stopMusic()
{
    clip1.stop();
}
```

**MainFrame.java**

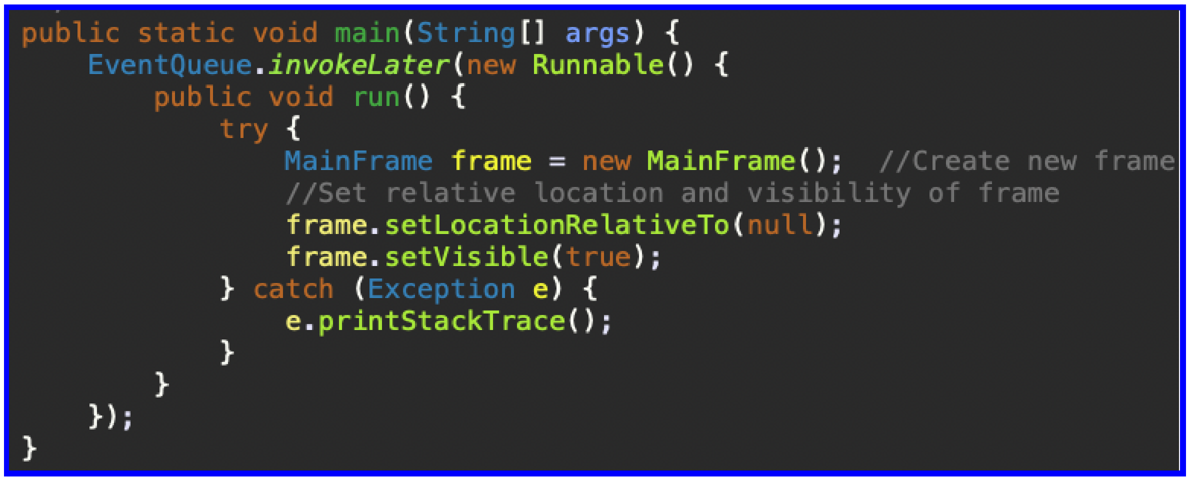
MainFrame is the entry point of the game. The user will be directed here upon running the application.

There are a few global variables: The music clip to play in the background, and the JPanel to display content.



```java
//Global variables
Clip clip;
JLabel ThankU;

private JPanel contentPane;
```

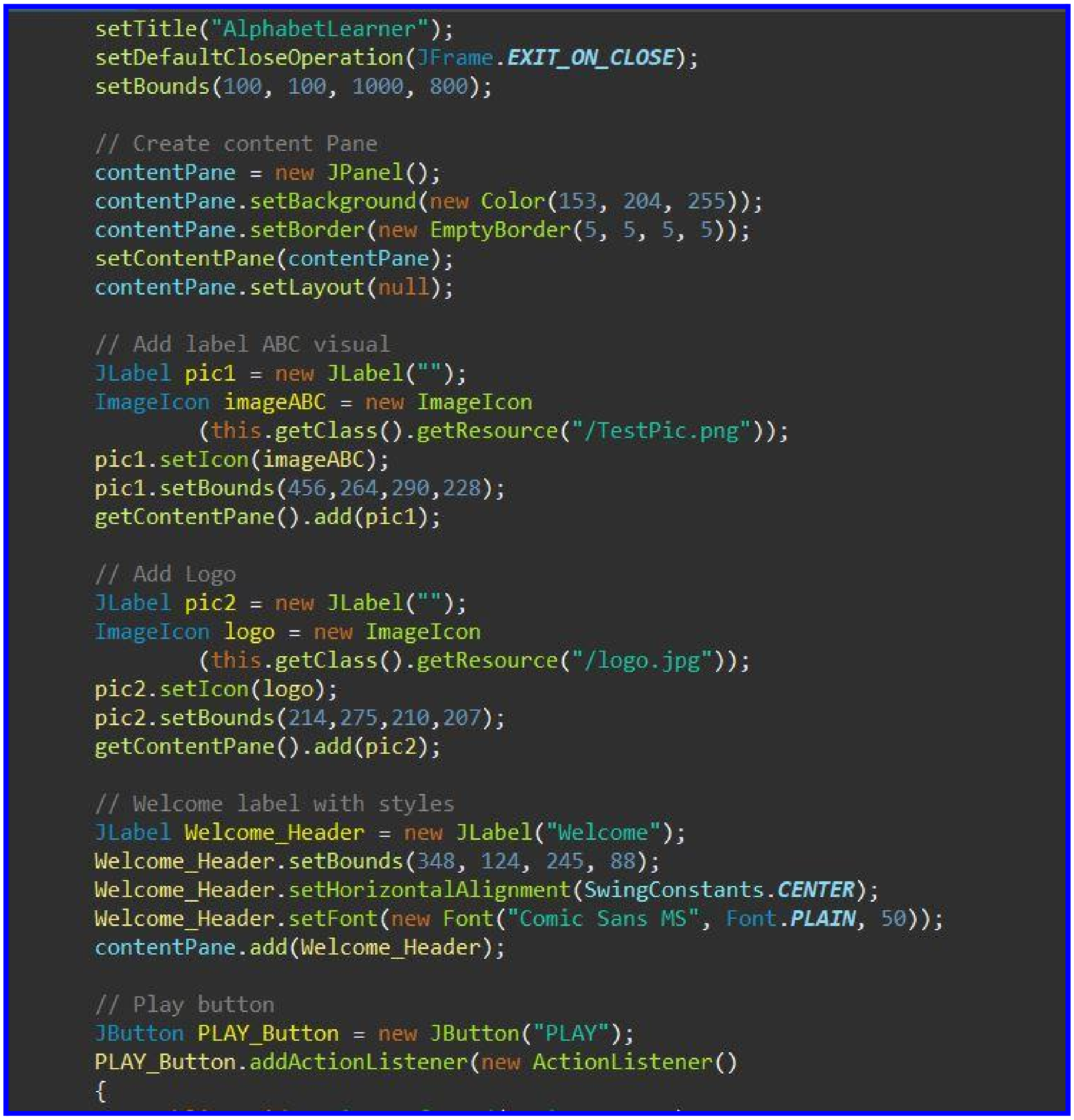
The main method follows. In this method, the frame is initialized and made visible to the user. Similar to the howToPlay page, the relative location is set to null so that the frame will be displayed in the center of the screen.



```java
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                MainFrame frame = new MainFrame();  //Create new frame
                //Set relative location and visibility of frame
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

Next, the constructor. All the visual elements are initialized, including the logo, labels, and buttons. See Figure 2A, Figure 2B, Figure 2C.

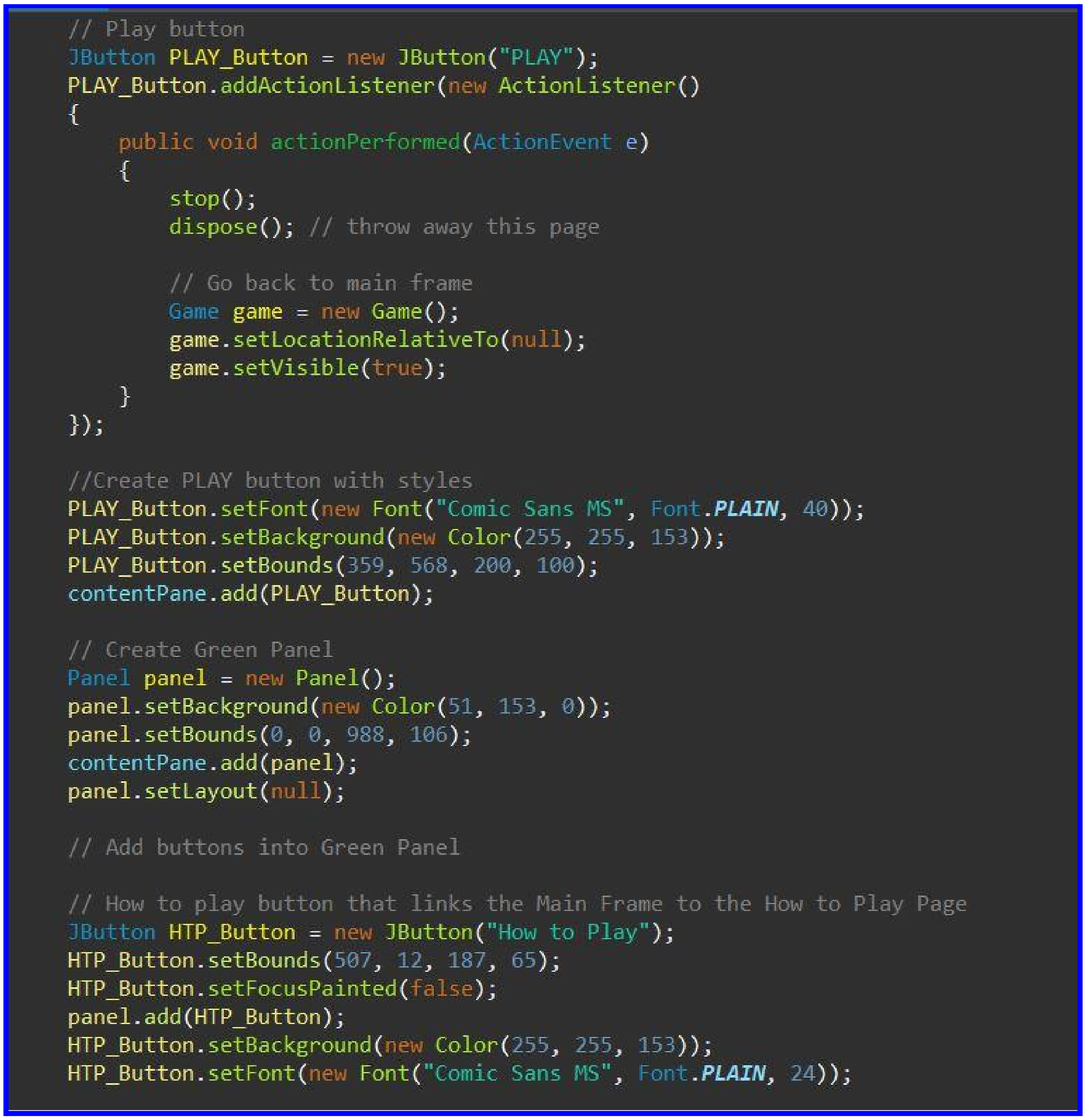
**Figure 2A**



```
setTitle("AlphabetLearner");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(100, 100, 1000, 800);

// Create content Pane
contentPane = new JPanel();
contentPane.setBackground(new Color(153, 204, 255));
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);
contentPane.setLayout(null);

// Add label ABC visual
JLabel pic1 = new JLabel("");
ImageIcon imageABC = new ImageIcon
        (this.getClass().getResource("/TestPic.png"));
pic1.setIcon(imageABC);
pic1.setBounds(456,264,290,228);
getContentPane().add(pic1);

// Add Logo
JLabel pic2 = new JLabel("");
ImageIcon logo = new ImageIcon
        (this.getClass().getResource("/logo.jpg"));
pic2.setIcon(logo);
pic2.setBounds(214,275,210,207);
getContentPane().add(pic2);

// Welcome label with styles
JLabel Welcome_Header = new JLabel("Welcome");
Welcome_Header.setBounds(348, 124, 245, 88);
Welcome_Header.setHorizontalAlignment(SwingConstants.CENTER);
Welcome_Header.setFont(new Font("Comic Sans MS", Font.PLAIN, 50));
contentPane.add(Welcome_Header);

// Play button
JButton PLAY_Button = new JButton("PLAY");
PLAY_Button.addActionListener(new ActionListener()
{
```

**Figure 2B**

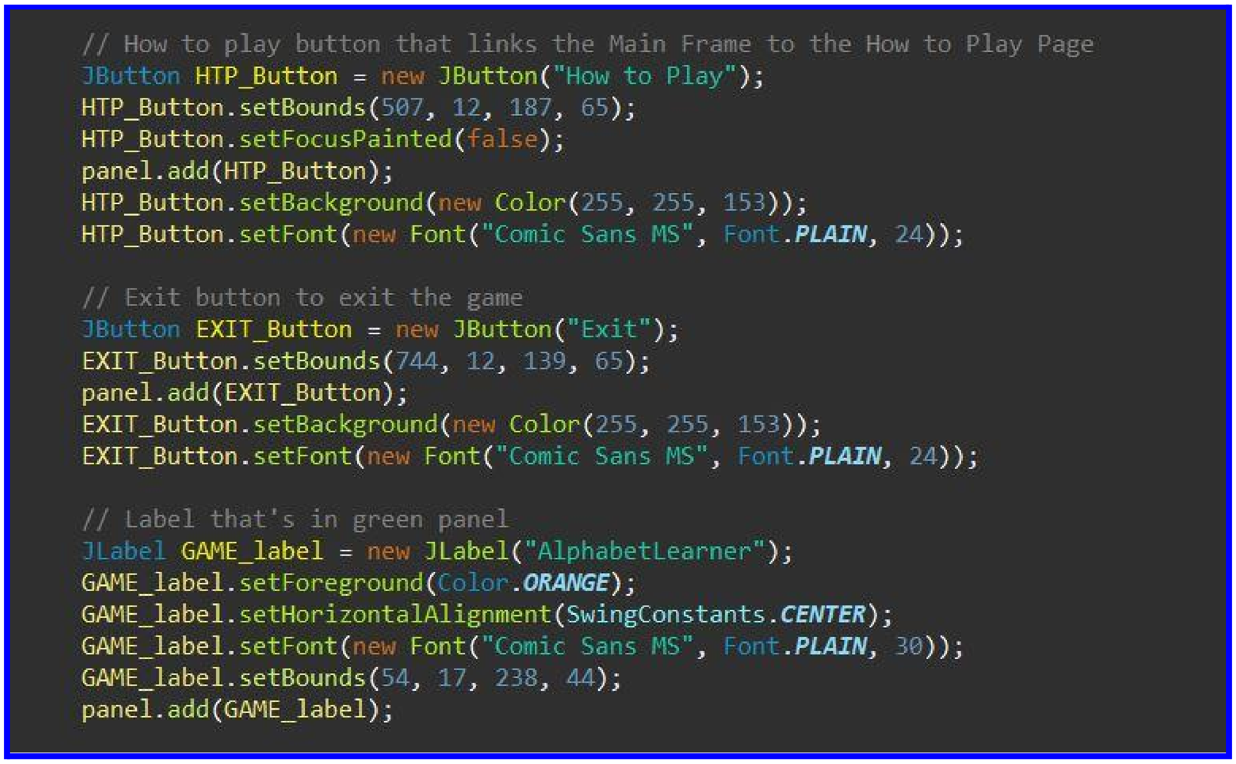
// An event listener is added for the play button. If the play button is clicked, the main frame music stops and the frame is disposed of. A new frame called game (Game.java)  is opened and made visible to the user.



```
    // Play button
    JButton PLAY_Button = new JButton("PLAY");
    PLAY_Button.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            stop();
            dispose(); // throw away this page

            // Go back to main frame
            Game game = new Game();
            game.setLocationRelativeTo(null);
            game.setVisible(true);
        }
    });

    //Create PLAY button with styles
    PLAY_Button.setFont(new Font("Comic Sans MS", Font.PLAIN, 40));
    PLAY_Button.setBackground(new Color(255, 255, 153));
    PLAY_Button.setBounds(359, 568, 200, 100);
    contentPane.add(PLAY_Button);

    // Create Green Panel
    Panel panel = new Panel();
    panel.setBackground(new Color(51, 153, 0));
    panel.setBounds(0, 0, 988, 106);
    contentPane.add(panel);
    panel.setLayout(null);

    // Add buttons into Green Panel

    // How to play button that links the Main Frame to the How to Play Page
    JButton HTP_Button = new JButton("How to Play");
    HTP_Button.setBounds(507, 12, 187, 65);
    HTP_Button.setFocusPainted(false);
    panel.add(HTP_Button);
    HTP_Button.setBackground(new Color(255, 255, 153));
    HTP_Button.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));
```

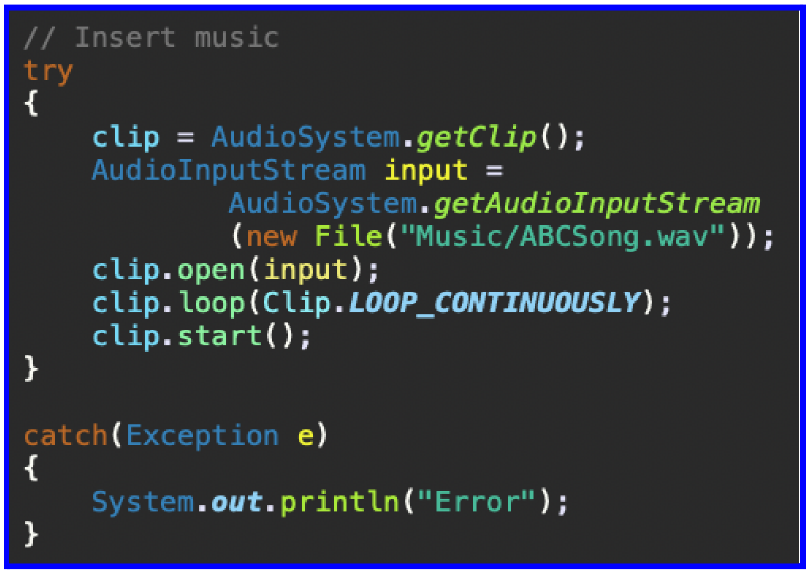
**Figure 2C:**

// The design of 2 buttons and 1 label.



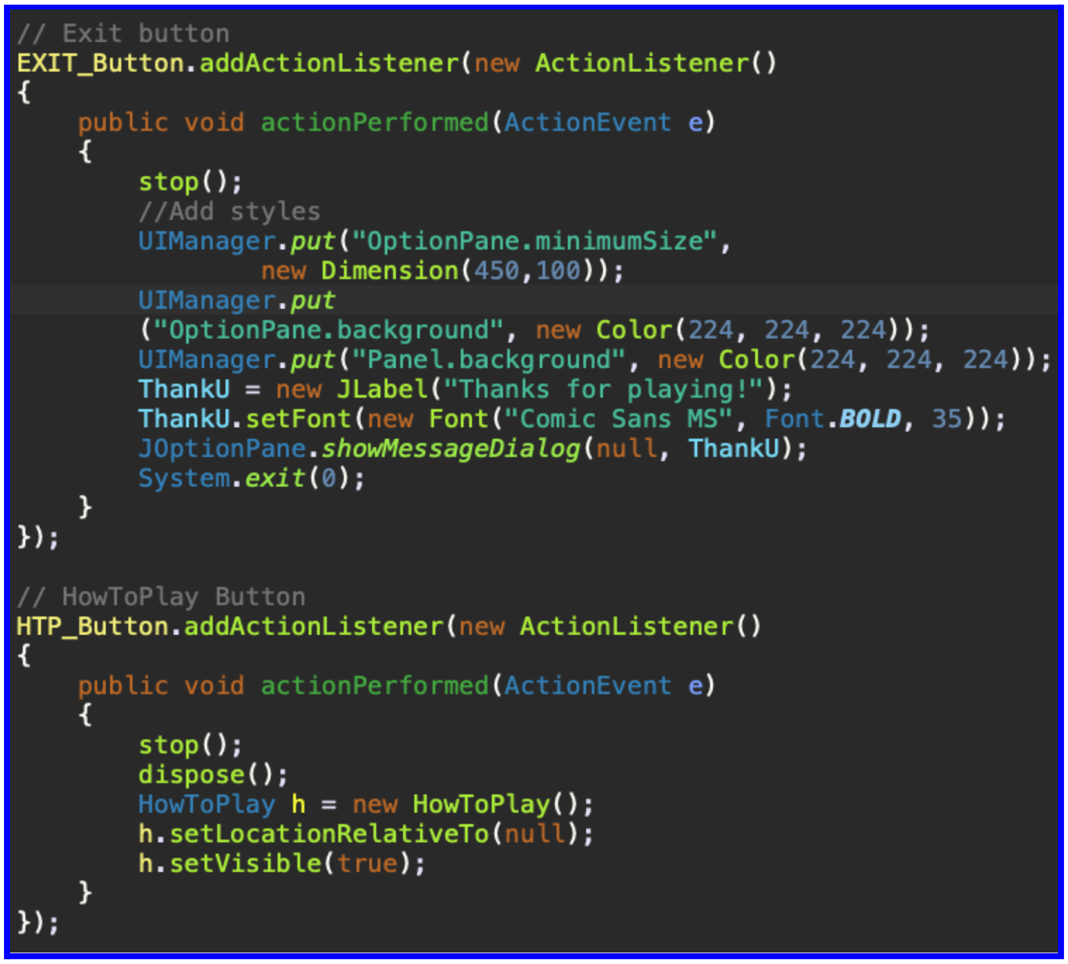
```
// How to play button that links the Main Frame to the How to Play Page
JButton HTP_Button = new JButton("How to Play");
HTP_Button.setBounds(507, 12, 187, 65);
HTP_Button.setFocusPainted(false);
panel.add(HTP_Button);
HTP_Button.setBackground(new Color(255, 255, 153));
HTP_Button.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));

// Exit button to exit the game
JButton EXIT_Button = new JButton("Exit");
EXIT_Button.setBounds(744, 12, 139, 65);
panel.add(EXIT_Button);
EXIT_Button.setBackground(new Color(255, 255, 153));
EXIT_Button.setFont(new Font("Comic Sans MS", Font.PLAIN, 24));

// Label that's in green panel
JLabel GAME_label = new JLabel("AlphabetLearner");
GAME_label.setForeground(Color.ORANGE);
GAME_label.setHorizontalAlignment(SwingConstants.CENTER);
GAME_label.setFont(new Font("Comic Sans MS", Font.PLAIN, 30));
GAME_label.setBounds(54, 17, 238, 44);
panel.add(GAME_label);
```

The music clip is referenced and played. It is looped to ensure that it doesn't stop if the end of the music clip is reached.



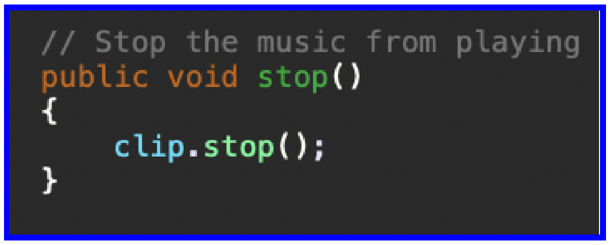
```
// Insert music
try
{
    clip = AudioSystem.getClip();
    AudioInputStream input =
            AudioSystem.getAudioInputStream
            (new File("Music/ABCSong.wav"));
    clip.open(input);
    clip.loop(Clip.LOOP_CONTINUOUSLY);
    clip.start();
}

catch(Exception e)
{
    System.out.println("Error");
}
```

Event listeners are added for the exit button and HowToPlay button. The exit button terminates the program while the HowToPlay button opens the HowToPlay.java file.



```
// Exit button
EXIT_Button.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        stop();
        //Add styles
        UIManager.put("OptionPane.minimumSize",
                new Dimension(450,100));
        UIManager.put
        ("OptionPane.background", new Color(224, 224, 224));
        UIManager.put("Panel.background", new Color(224, 224, 224));
        ThankU = new JLabel("Thanks for playing!");
        ThankU.setFont(new Font("Comic Sans MS", Font.BOLD, 35));
        JOptionPane.showMessageDialog(null, ThankU);
        System.exit(0);
    }
});

// HowToPlay Button
HTP_Button.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        stop();
        dispose();
        HowToPlay h = new HowToPlay();
        h.setLocationRelativeTo(null);
        h.setVisible(true);
    }
});
```

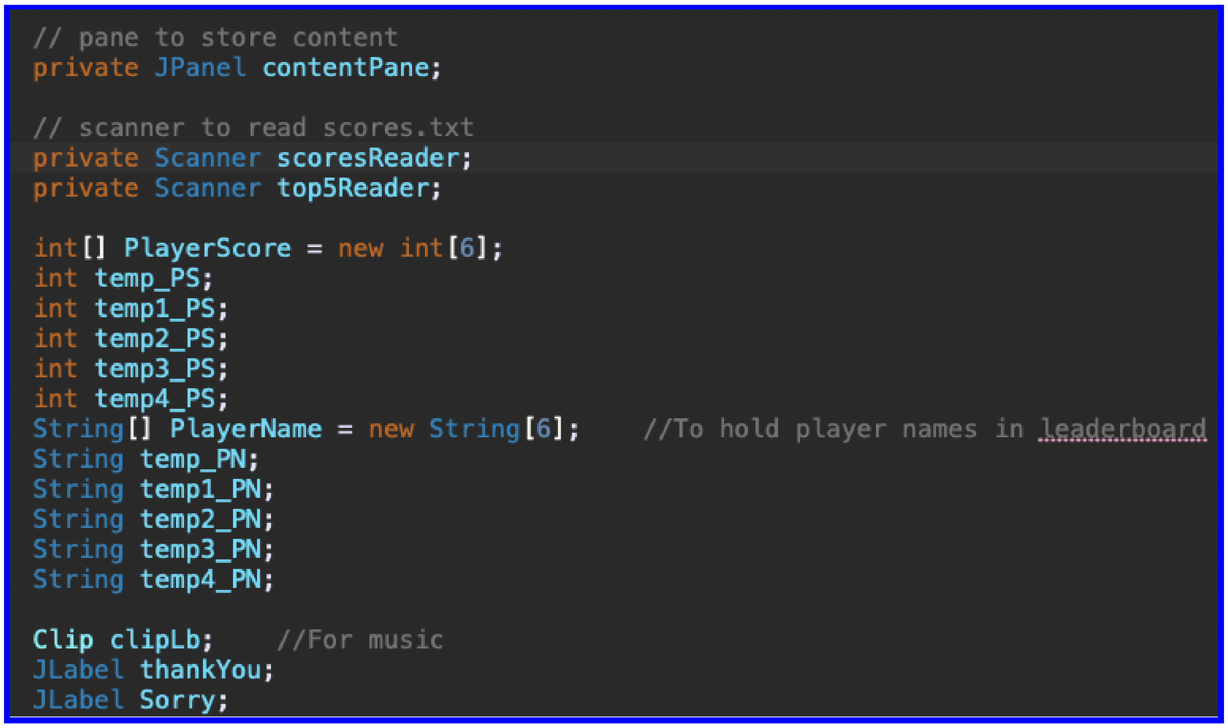
A stop method is initialized to stop the music once a page is changed.



```
// Stop the music from playing
public void stop()
{
    clip.stop();
}
```

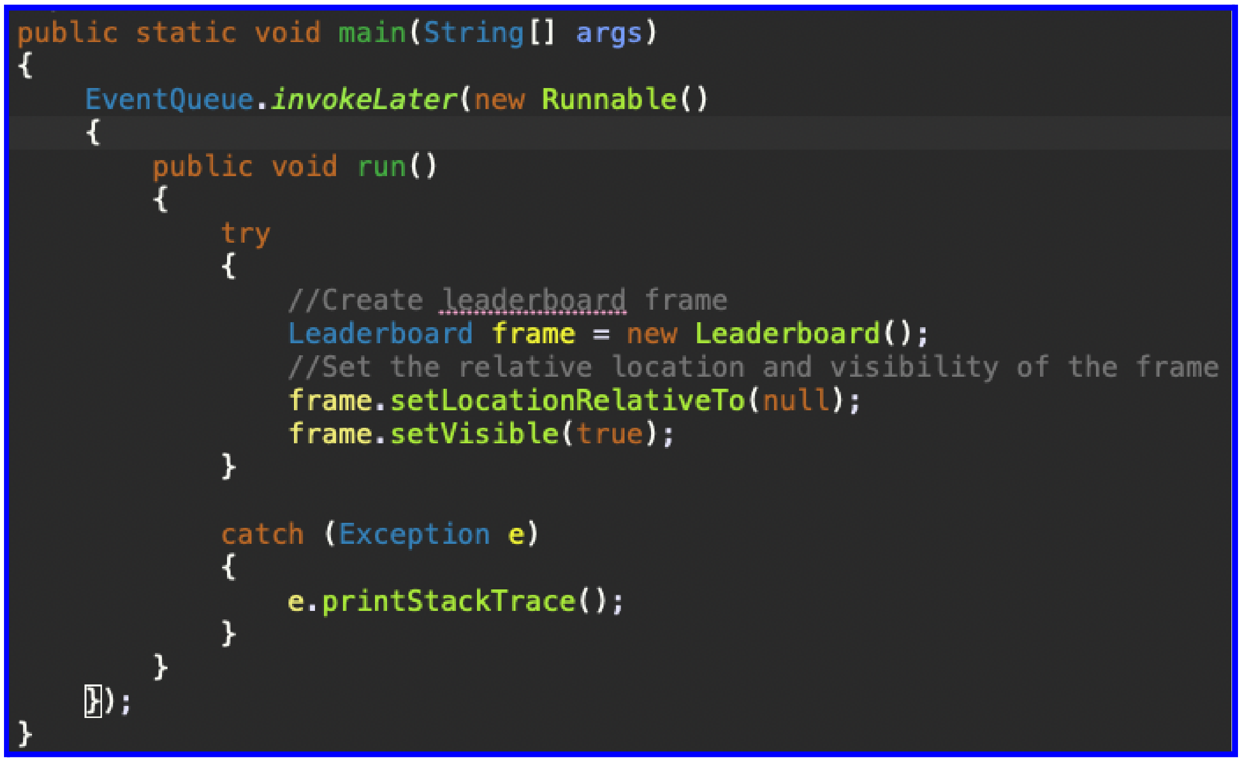
**<u>Leaderboard.java</u>**

The class begins with a declaration of the JPanel and contentPane where all content is being displayed. It is followed by two scanners which are used to read the assisting score file, “scores.txt” (which holds the top 5 scores). In addition, the class contains an array of 6 integers, PlayerScore, which is later used to find how the new score will fit into the leaderboard. Music files are stored as well to play music during the game. Global temp (temporary) variables are used to swap both playerName and corresponding playerScore when sorting the leaderboard. A label named “Sorry” is to display a Sorry message in JOptionPane if the user did not make it into the leaderboard.



```
// pane to store content
private JPanel contentPane;

// scanner to read scores.txt
private Scanner scoresReader;
private Scanner top5Reader;

int[] PlayerScore = new int[6];
int temp_PS;
int temp1_PS;
int temp2_PS;
int temp3_PS;
int temp4_PS;
String[] PlayerName = new String[6];     //To hold player names in leaderboard
String temp_PN;
String temp1_PN;
String temp2_PN;
String temp3_PN;
String temp4_PN;

Clip clipLb;     //For music
JLabel thankYou;
JLabel Sorry;
```

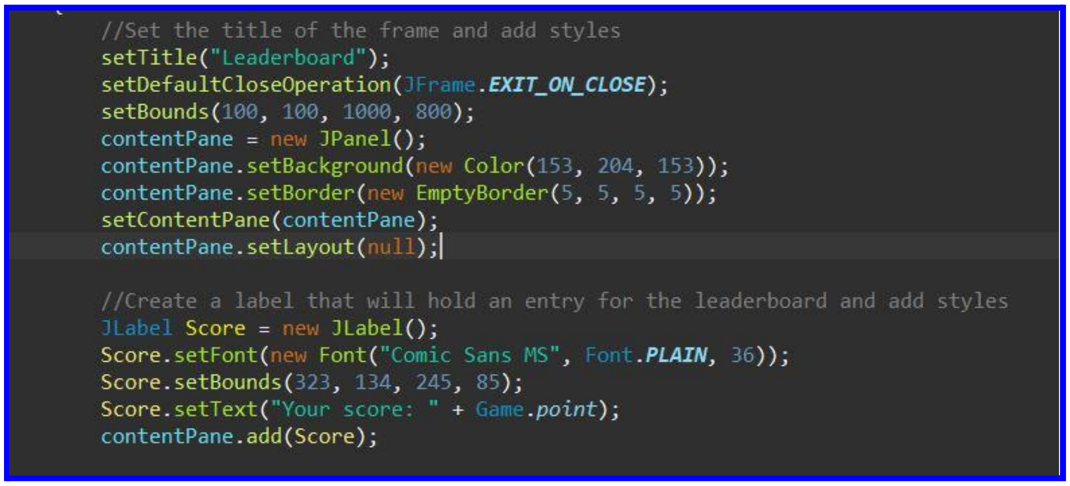
The main method follows. Within the main method, we trigger the invokeLater method from the EventQueue class and create an instance of the Leaderboard.



```
public static void main(String[] args)
{
    EventQueue.invokeLater(new Runnable()
    {
        public void run()
        {
            try
            {
                //Create leaderboard frame
                Leaderboard frame = new Leaderboard();
                //Set the relative location and visibility of the frame
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);
            }

            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
    });
}
```

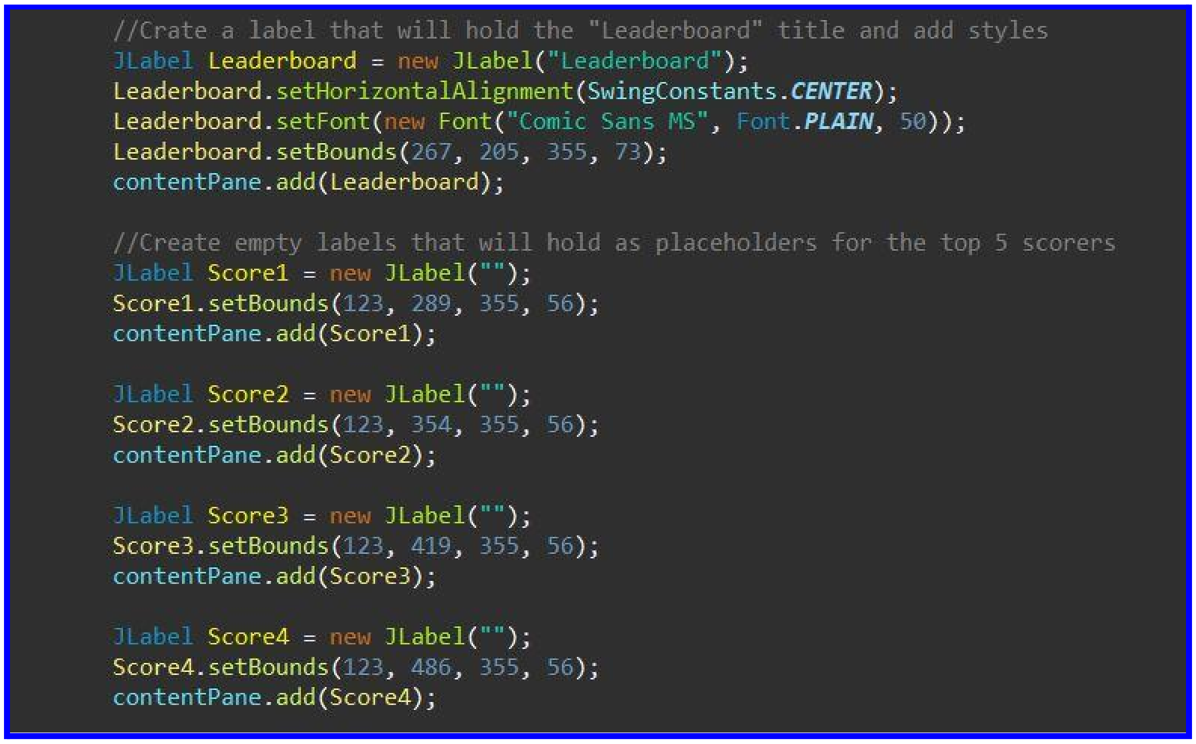
We then reach the constructor of the Leaderboard class. It contains several labels, buttons, and panels to display information to the user. See Figure 3A1, 3A2, 3B1, 3B2, and 3C.

**Figure 3A1**



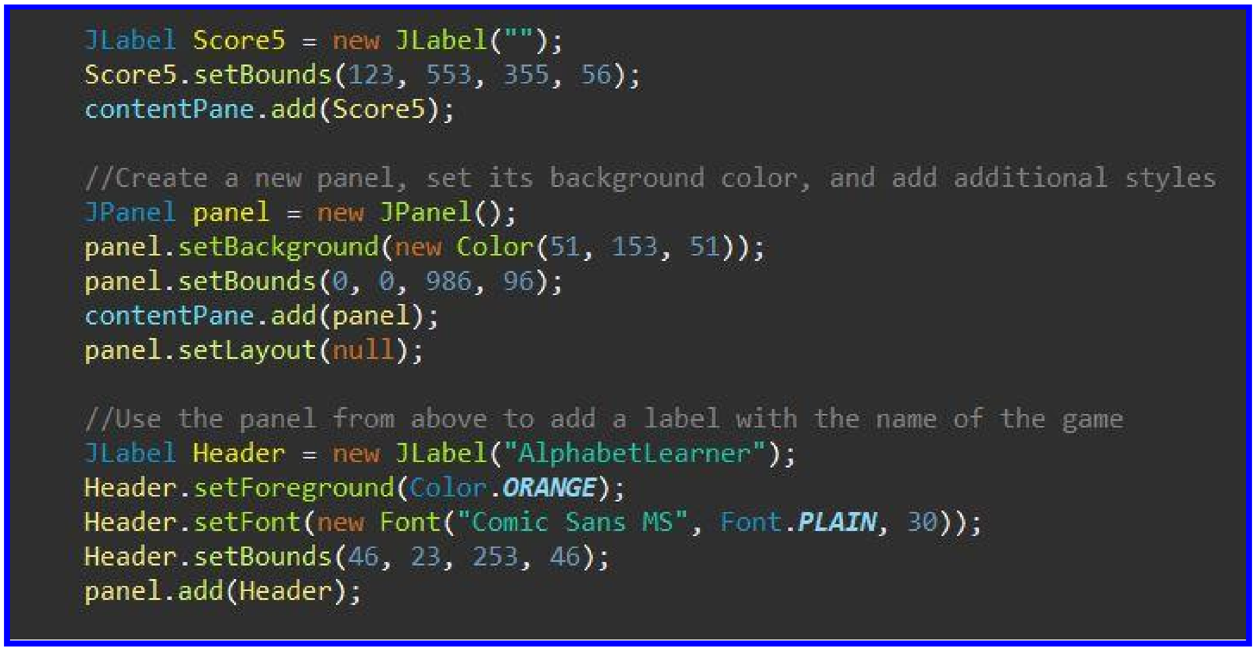
```
        //Set the title of the frame and add styles
        setTitle("Leaderboard");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 1000, 800);
        contentPane = new JPanel();
        contentPane.setBackground(new Color(153, 204, 153));
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        //Create a label that will hold an entry for the leaderboard and add styles
        JLabel Score = new JLabel();
        Score.setFont(new Font("Comic Sans MS", Font.PLAIN, 36));
        Score.setBounds(323, 134, 245, 85);
        Score.setText("Your score: " + Game.point);
        contentPane.add(Score);
```

**Figure 3A2**



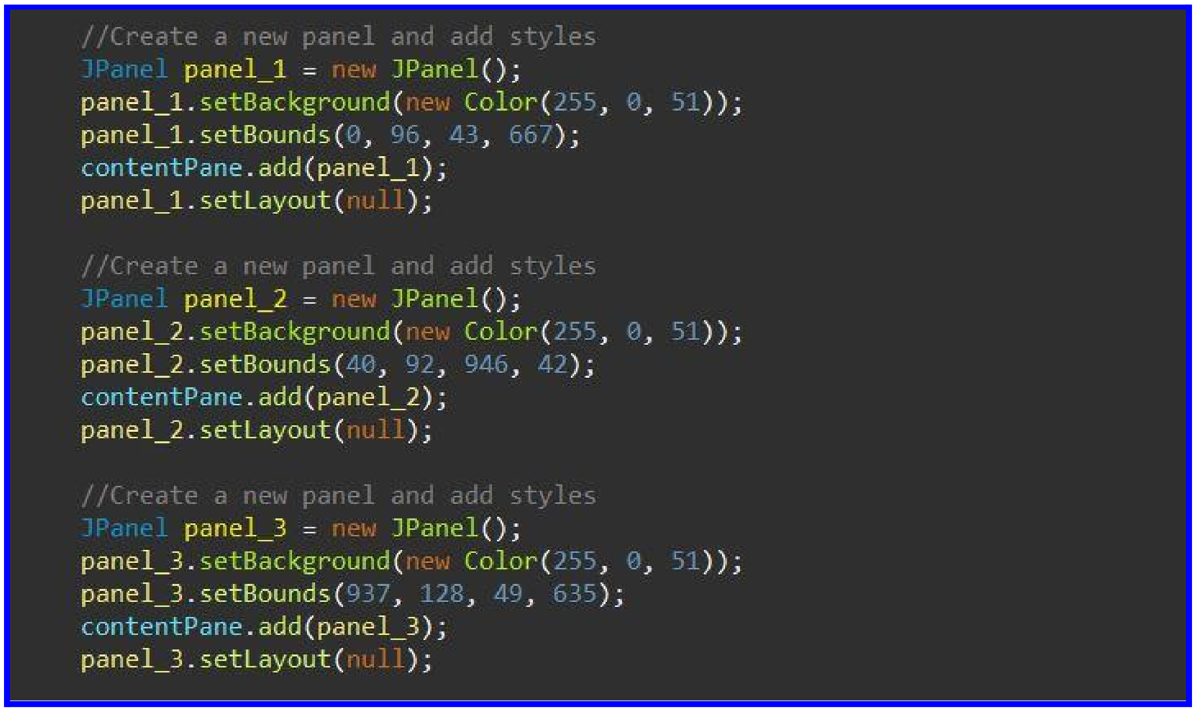
```
//Crate a label that will hold the "Leaderboard" title and add styles
JLabel Leaderboard = new JLabel("Leaderboard");
Leaderboard.setHorizontalAlignment(SwingConstants.CENTER);
Leaderboard.setFont(new Font("Comic Sans MS", Font.PLAIN, 50));
Leaderboard.setBounds(267, 205, 355, 73);
contentPane.add(Leaderboard);

//Create empty labels that will hold as placeholders for the top 5 scorers
JLabel Score1 = new JLabel("");
Score1.setBounds(123, 289, 355, 56);
contentPane.add(Score1);

JLabel Score2 = new JLabel("");
Score2.setBounds(123, 354, 355, 56);
contentPane.add(Score2);

JLabel Score3 = new JLabel("");
Score3.setBounds(123, 419, 355, 56);
contentPane.add(Score3);

JLabel Score4 = new JLabel("");
Score4.setBounds(123, 486, 355, 56);
contentPane.add(Score4);
```

**Figure 3B1**



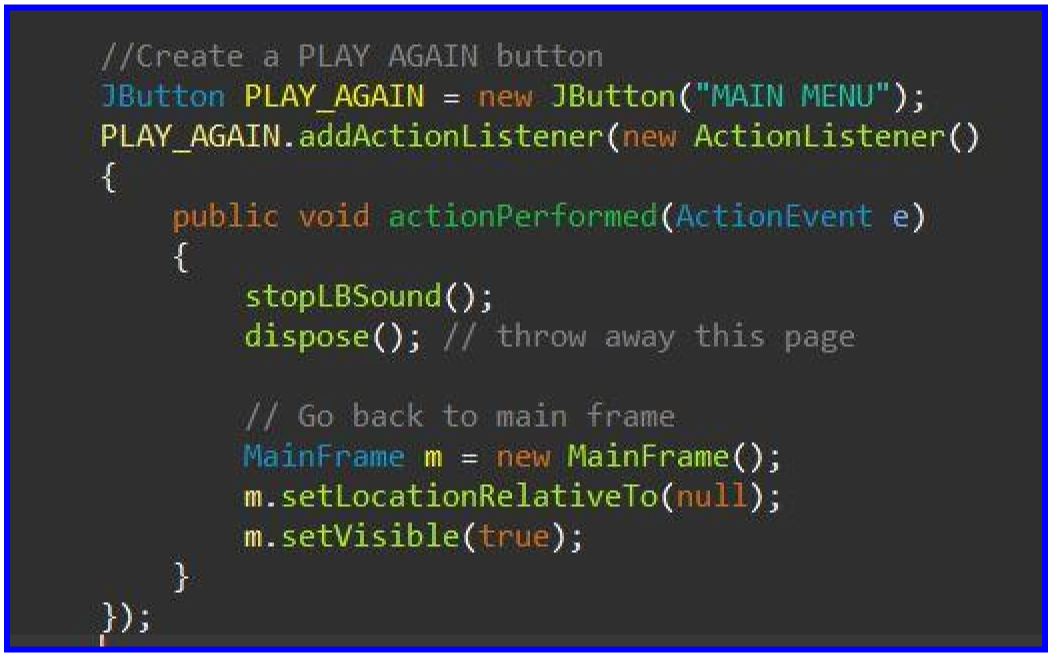
```
JLabel Score5 = new JLabel("");
Score5.setBounds(123, 553, 355, 56);
contentPane.add(Score5);

//Create a new panel, set its background color, and add additional styles
JPanel panel = new JPanel();
panel.setBackground(new Color(51, 153, 51));
panel.setBounds(0, 0, 986, 96);
contentPane.add(panel);
panel.setLayout(null);

//Use the panel from above to add a label with the name of the game
JLabel Header = new JLabel("AlphabetLearner");
Header.setForeground(Color.ORANGE);
Header.setFont(new Font("Comic Sans MS", Font.PLAIN, 30));
Header.setBounds(46, 23, 253, 46);
panel.add(Header);
```

**Figure 3B2**



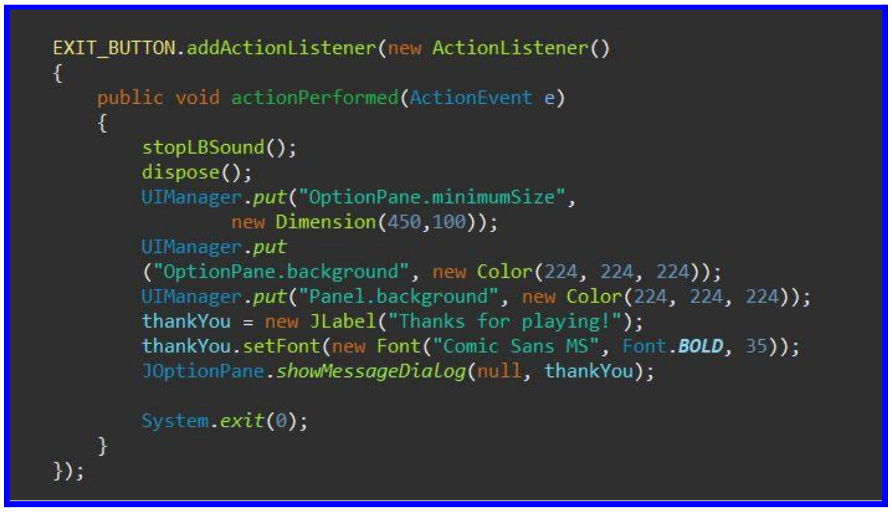
```
//Create a new panel and add styles
JPanel panel_1 = new JPanel();
panel_1.setBackground(new Color(255, 0, 51));
panel_1.setBounds(0, 96, 43, 667);
contentPane.add(panel_1);
panel_1.setLayout(null);

//Create a new panel and add styles
JPanel panel_2 = new JPanel();
panel_2.setBackground(new Color(255, 0, 51));
panel_2.setBounds(40, 92, 946, 42);
contentPane.add(panel_2);
panel_2.setLayout(null);

//Create a new panel and add styles
JPanel panel_3 = new JPanel();
panel_3.setBackground(new Color(255, 0, 51));
panel_3.setBounds(937, 128, 49, 635);
contentPane.add(panel_3);
panel_3.setLayout(null);
```

**Figure 3C**



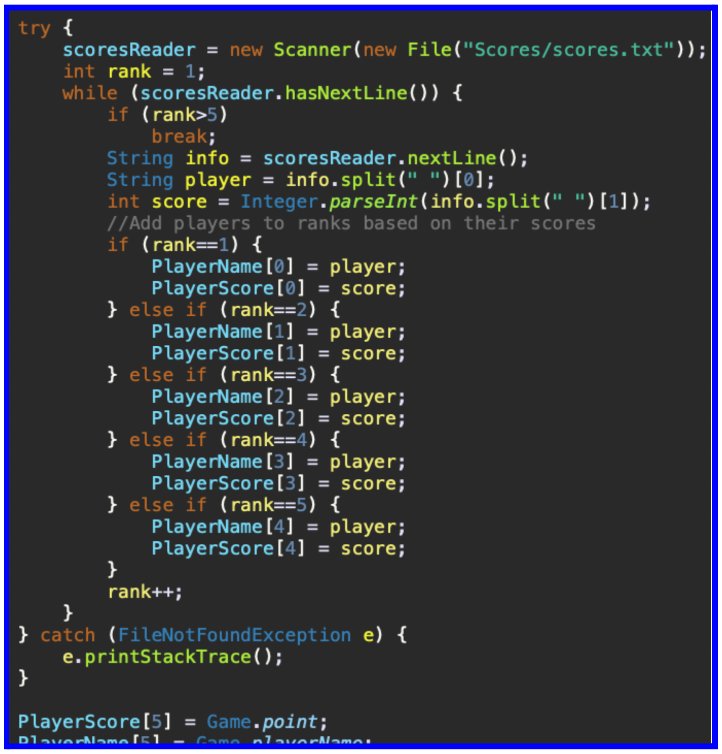
```
//Create a PLAY AGAIN button
JButton PLAY_AGAIN = new JButton("MAIN MENU");
PLAY_AGAIN.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        stopLBSound();
        dispose(); // throw away this page

        // Go back to main frame
        MainFrame m = new MainFrame();
        m.setLocationRelativeTo(null);
        m.setVisible(true);
    }
});
```

Event listeners are added for the exit button. The UI manager will display a thank you panel and the game will close.



```
EXIT_BUTTON.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        stopLBSound();
        dispose();
        UIManager.put("OptionPane.minimumSize",
                new Dimension(450,100));
        UIManager.put
        ("OptionPane.background", new Color(224, 224, 224));
        UIManager.put("Panel.background", new Color(224, 224, 224));
        thankYou = new JLabel("Thanks for playing!");
        thankYou.setFont(new Font("Comic Sans MS", Font.BOLD, 35));
        JOptionPane.showMessageDialog(null, thankYou);

        System.exit(0);
    }
});
```

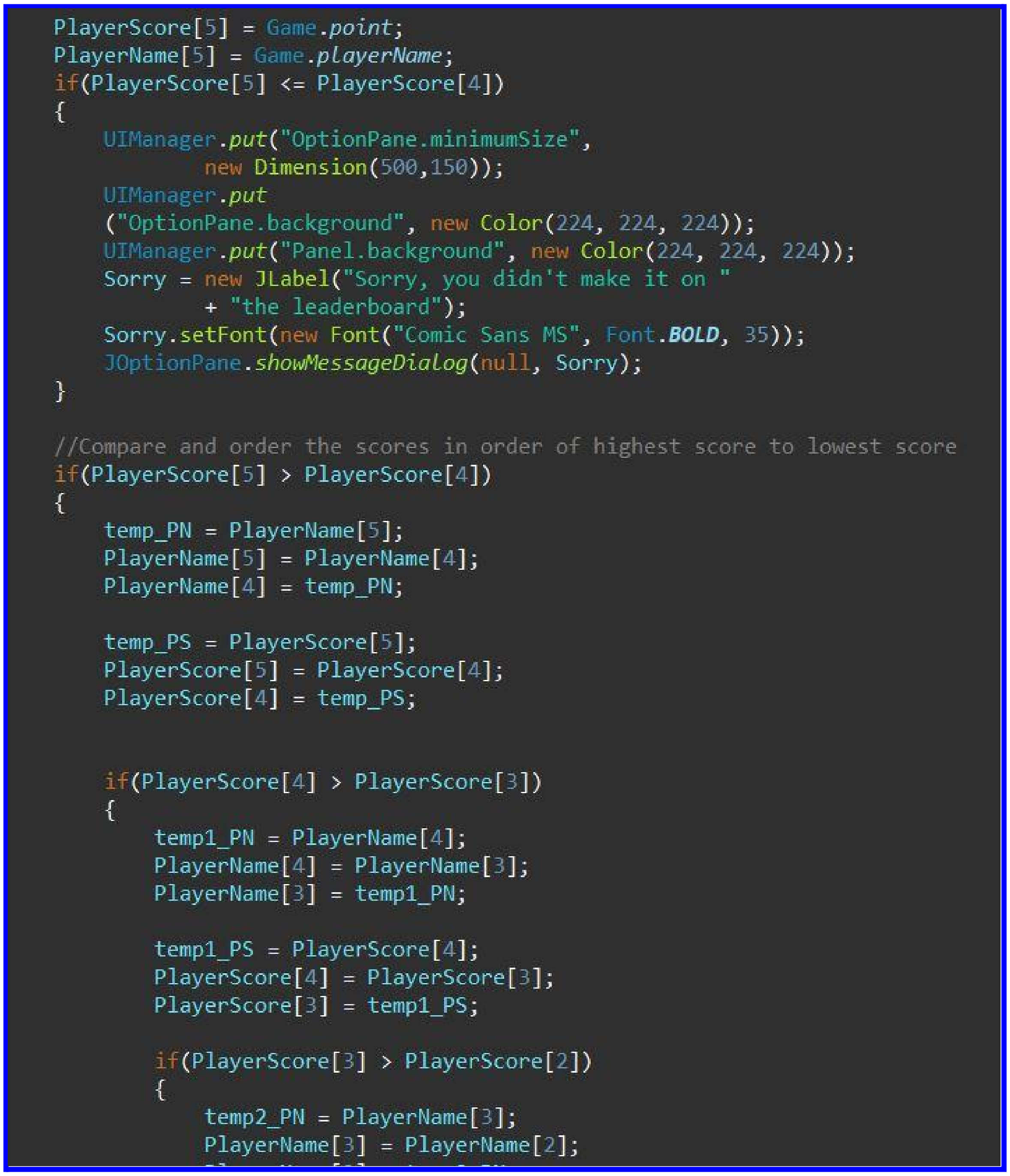
Next, we read scores.txt with our scanner from earlier. We have two arrays of players and scores and we assign the values to their corresponding array. The new score of the user once completing the game is added.



```
try {
    scoresReader = new Scanner(new File("Scores/scores.txt"));
    int rank = 1;
    while (scoresReader.hasNextLine()) {
        if (rank>5)
            break;
        String info = scoresReader.nextLine();
        String player = info.split(" ")[0];
        int score = Integer.parseInt(info.split(" ")[1]);
        //Add players to ranks based on their scores
        if (rank==1) {
            PlayerName[0] = player;
            PlayerScore[0] = score;
        } else if (rank==2) {
            PlayerName[1] = player;
            PlayerScore[1] = score;
        } else if (rank==3) {
            PlayerName[2] = player;
            PlayerScore[2] = score;
        } else if (rank==4) {
            PlayerName[3] = player;
            PlayerScore[3] = score;
        } else if (rank==5) {
            PlayerName[4] = player;
            PlayerScore[4] = score;
        }
        rank++;
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
}

PlayerScore[5] = Game.point;
PlayerName[5] = Game.playerName;
```

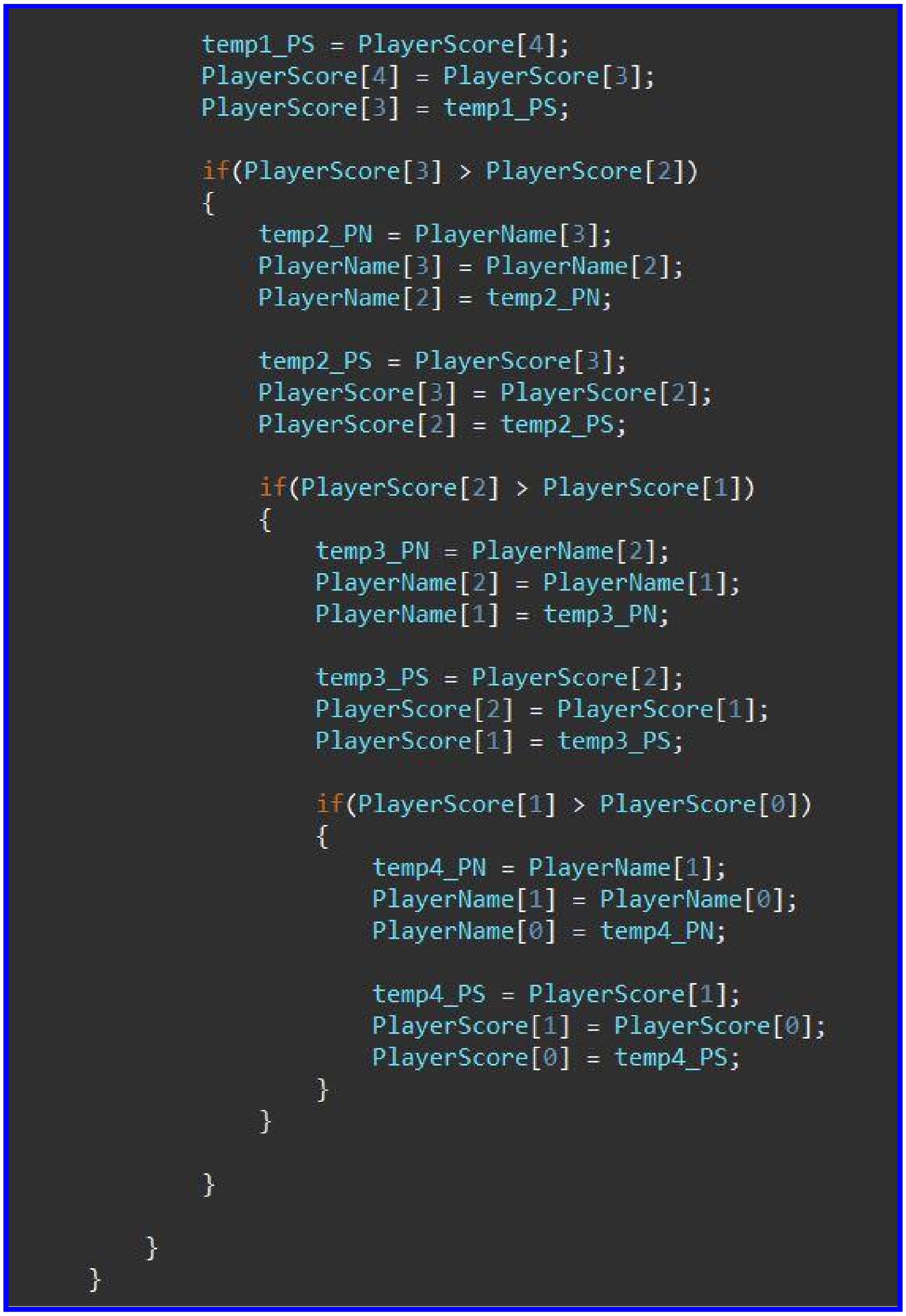
From there, both the score array is sorted using the new score. The top 5 scores and their corresponding users are saved. See Figure 4A, 4B.

**Figure 4A**



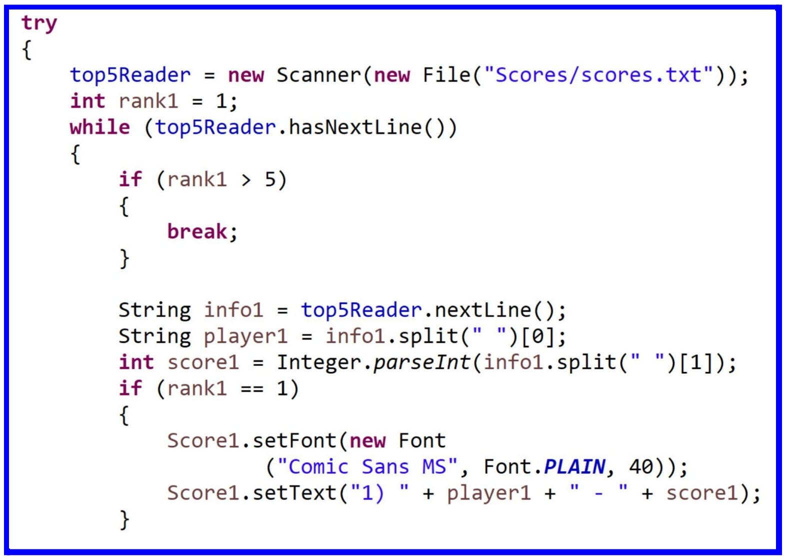
```
PlayerScore[5] = Game.point;
PlayerName[5] = Game.playerName;
if(PlayerScore[5] <= PlayerScore[4])
{
    UIManager.put("OptionPane.minimumSize",
            new Dimension(500,150));
    UIManager.put
    ("OptionPane.background", new Color(224, 224, 224));
    UIManager.put("Panel.background", new Color(224, 224, 224));
    Sorry = new JLabel("Sorry, you didn't make it on "
            + "the leaderboard");
    Sorry.setFont(new Font("Comic Sans MS", Font.BOLD, 35));
    JOptionPane.showMessageDialog(null, Sorry);
}

//Compare and order the scores in order of highest score to lowest score
if(PlayerScore[5] > PlayerScore[4])
{
    temp_PN = PlayerName[5];
    PlayerName[5] = PlayerName[4];
    PlayerName[4] = temp_PN;

    temp_PS = PlayerScore[5];
    PlayerScore[5] = PlayerScore[4];
    PlayerScore[4] = temp_PS;


    if(PlayerScore[4] > PlayerScore[3])
    {
        temp1_PN = PlayerName[4];
        PlayerName[4] = PlayerName[3];
        PlayerName[3] = temp1_PN;

        temp1_PS = PlayerScore[4];
        PlayerScore[4] = PlayerScore[3];
        PlayerScore[3] = temp1_PS;

        if(PlayerScore[3] > PlayerScore[2])
        {
            temp2_PN = PlayerName[3];
            PlayerName[3] = PlayerName[2];
```

**Figure 4B**

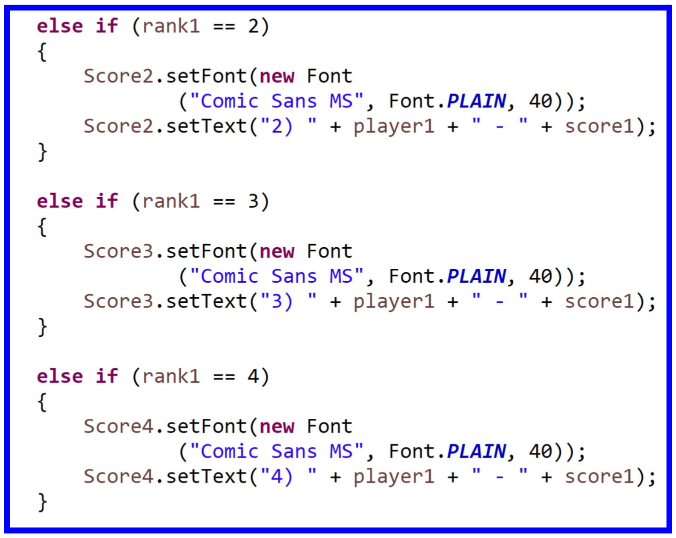


```
                temp1_PS = PlayerScore[4];
                PlayerScore[4] = PlayerScore[3];
                PlayerScore[3] = temp1_PS;

                if(PlayerScore[3] > PlayerScore[2])
                {
                    temp2_PN = PlayerName[3];
                    PlayerName[3] = PlayerName[2];
                    PlayerName[2] = temp2_PN;

                    temp2_PS = PlayerScore[3];
                    PlayerScore[3] = PlayerScore[2];
                    PlayerScore[2] = temp2_PS;

                    if(PlayerScore[2] > PlayerScore[1])
                    {
                        temp3_PN = PlayerName[2];
                        PlayerName[2] = PlayerName[1];
                        PlayerName[1] = temp3_PN;

                        temp3_PS = PlayerScore[2];
                        PlayerScore[2] = PlayerScore[1];
                        PlayerScore[1] = temp3_PS;

                        if(PlayerScore[1] > PlayerScore[0])
                        {
                            temp4_PN = PlayerName[1];
                            PlayerName[1] = PlayerName[0];
                            PlayerName[0] = temp4_PN;

                            temp4_PS = PlayerScore[1];
                            PlayerScore[1] = PlayerScore[0];
                            PlayerScore[0] = temp4_PS;
                        }
                    }

                }

            }
        }
```

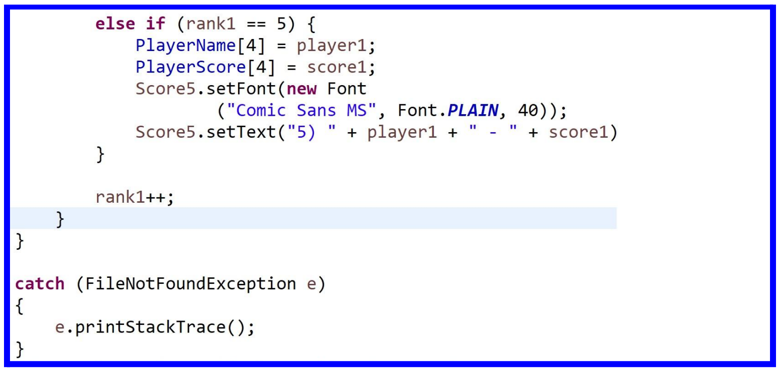
Finally, the scores are written back to scores.txt and using a loop we display the top 5 scores using their corresponding labels in the GUI.



```
try
{
    top5Reader = new Scanner(new File("Scores/scores.txt"));
    int rank1 = 1;
    while (top5Reader.hasNextLine())
    {
        if (rank1 > 5)
        {
            break;
        }

        String info1 = top5Reader.nextLine();
        String player1 = info1.split(" ")[0];
        int score1 = Integer.parseInt(info1.split(" ")[1]);
        if (rank1 == 1)
        {
            Score1.setFont(new Font
                    ("Comic Sans MS", Font.PLAIN, 40));
            Score1.setText("1) " + player1 + " - " + score1);
        }
```



```
else if (rank1 == 2)
{
    Score2.setFont(new Font
            ("Comic Sans MS", Font.PLAIN, 40));
    Score2.setText("2) " + player1 + " - " + score1);
}

else if (rank1 == 3)
{
    Score3.setFont(new Font
            ("Comic Sans MS", Font.PLAIN, 40));
    Score3.setText("3) " + player1 + " - " + score1);
}

else if (rank1 == 4)
{
    Score4.setFont(new Font
            ("Comic Sans MS", Font.PLAIN, 40));
    Score4.setText("4) " + player1 + " - " + score1);
}
```

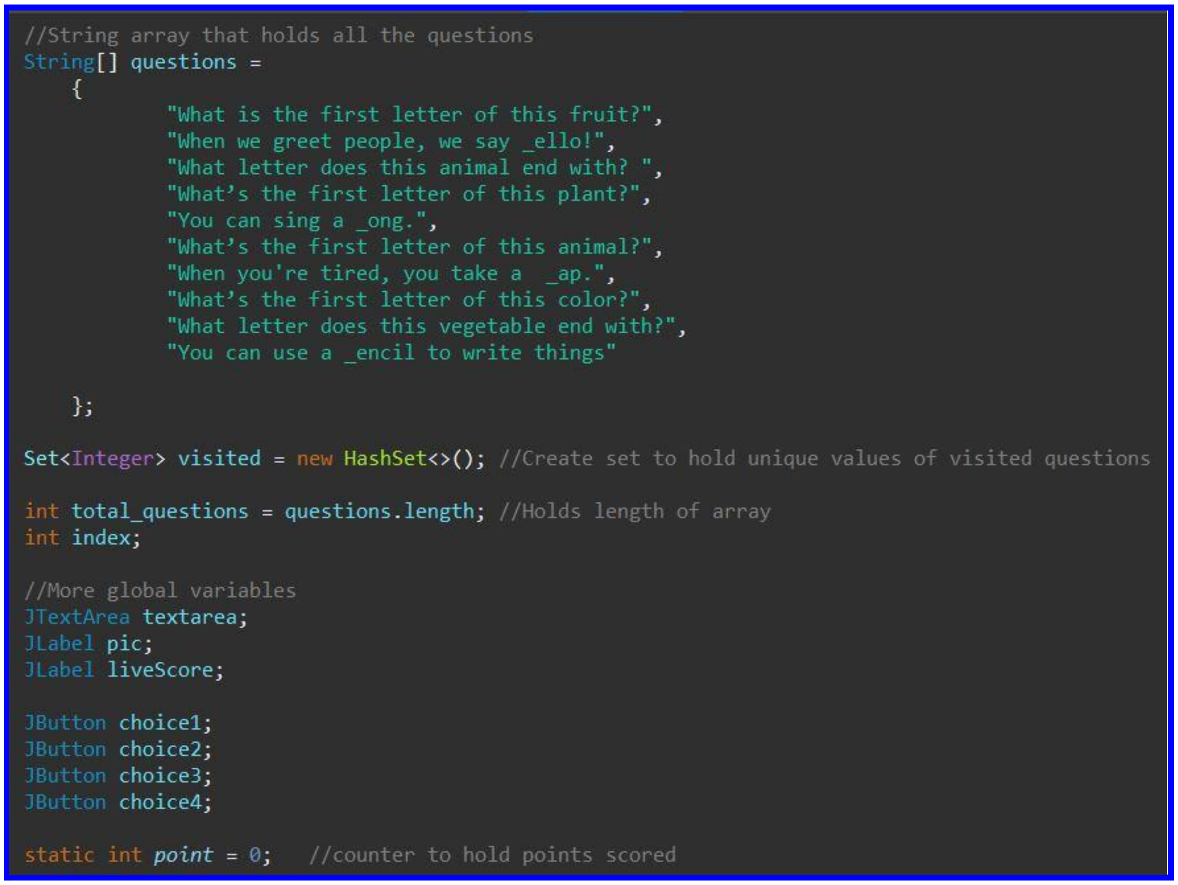


```
        else if (rank1 == 5) {
            PlayerName[4] = player1;
            PlayerScore[4] = score1;
            Score5.setFont(new Font
                    ("Comic Sans MS", Font.PLAIN, 40));
            Score5.setText("5) " + player1 + " - " + score1)
        }

        rank1++;
    }
}

catch (FileNotFoundException e)
{
    e.printStackTrace();
}
```

**Game.java**

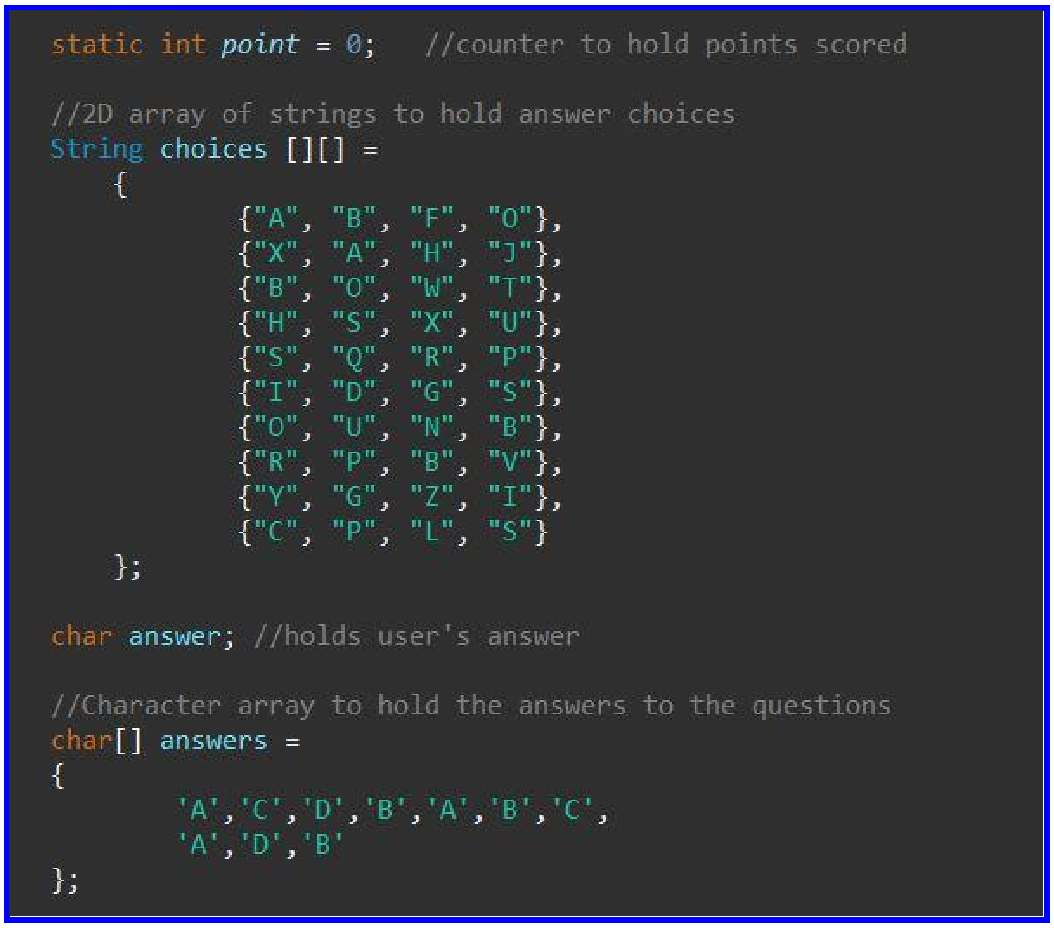
This class implements the game mechanics which deals with questions, answers, and points. Questions, answers, and answer choices are stored in String and char arrays. The arrays are related via their corresponding indices. The 2D array "choices" contain possible answer choices for a question, and the answers array contains the correct answer(answer key). Think of a multiple choice test. Answers are usually stored in the form of "A", "B", "C", or "D". That was the inspiration when building the "answers" array. For example, if questions[1] is "When we greet people, we say _ello!" and the corresponding choices[1] is {"X", "A", "H", "J"}, this means that the correct answer is "H" (if the third button is clicked). The user will earn a point only if the user's answer matches the character stored in answers[1]. See Figure A and FigureB.

**Figure A**



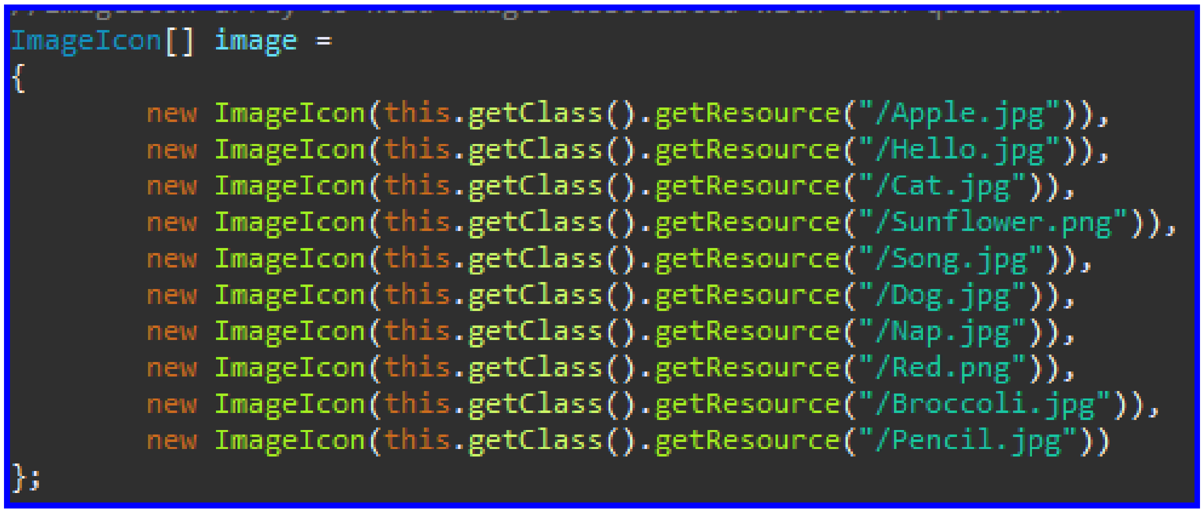
```
//String array that holds all the questions
String[] questions =
    {
            "What is the first letter of this fruit?",
            "When we greet people, we say _ello!",
            "What letter does this animal end with? ",
            "What’s the first letter of this plant?",
            "You can sing a _ong.",
            "What’s the first letter of this animal?",
            "When you're tired, you take a  _ap.",
            "What’s the first letter of this color?",
            "What letter does this vegetable end with?",
            "You can use a _encil to write things"

    };

Set<Integer> visited = new HashSet<>(); //Create set to hold unique values of visited questions

int total_questions = questions.length; //Holds length of array
int index;

//More global variables
JTextArea textarea;
JLabel pic;
JLabel liveScore;

JButton choice1;
JButton choice2;
JButton choice3;
JButton choice4;

static int point = 0;   //counter to hold points scored
```

**Figure B**



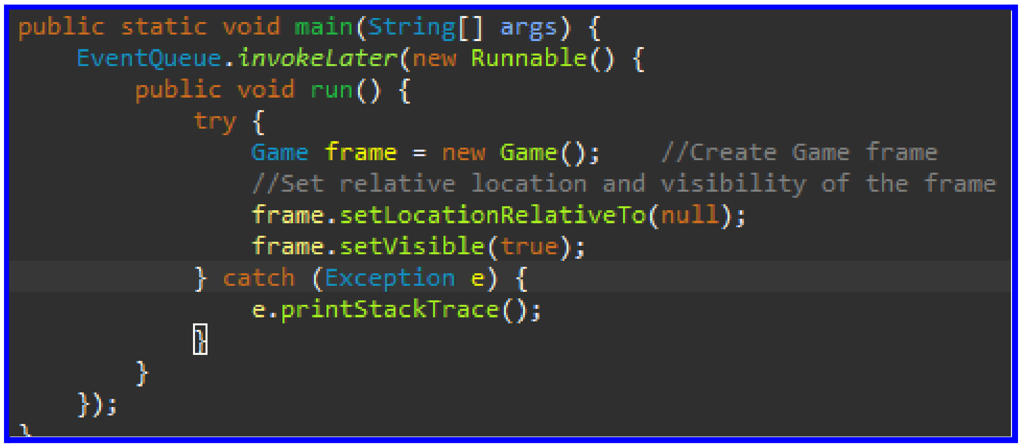
```java
static int point = 0;    //counter to hold points scored

//2D array of strings to hold answer choices
String choices [][] =
    {
            {"A", "B", "F", "O"},
            {"X", "A", "H", "J"},
            {"B", "O", "W", "T"},
            {"H", "S", "X", "U"},
            {"S", "Q", "R", "P"},
            {"I", "D", "G", "S"},
            {"O", "U", "N", "B"},
            {"R", "P", "B", "V"},
            {"Y", "G", "Z", "I"},
            {"C", "P", "L", "S"}
    };

char answer; //holds user's answer

//Character array to hold the answers to the questions
char[] answers =
{
        'A','C','D','B','A','B','C',
        'A','D','B'
};
```

The image array contains ImageIcons with images that correspond to each question / answer.



```java
ImageIcon[] image =
{
        new ImageIcon(this.getClass().getResource("/Apple.jpg")),
        new ImageIcon(this.getClass().getResource("/Hello.jpg")),
        new ImageIcon(this.getClass().getResource("/Cat.jpg")),
        new ImageIcon(this.getClass().getResource("/Sunflower.png")),
        new ImageIcon(this.getClass().getResource("/Song.jpg")),
        new ImageIcon(this.getClass().getResource("/Dog.jpg")),
        new ImageIcon(this.getClass().getResource("/Nap.jpg")),
        new ImageIcon(this.getClass().getResource("/Red.png")),
        new ImageIcon(this.getClass().getResource("/Broccoli.jpg")),
        new ImageIcon(this.getClass().getResource("/Pencil.jpg"))
};
```

The main method creates the game frame which displays the game itself.

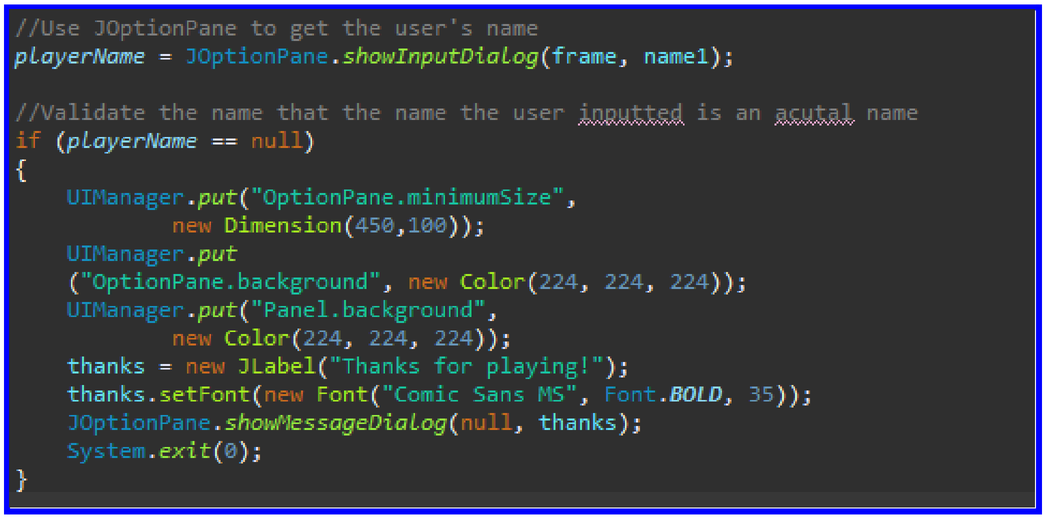


```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Game frame = new Game();     //Create Game frame
                //Set relative location and visibility of the frame
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
```

The game asks for the player's name first. It validates it by checking if it has length of zero or spaces. If the user breaks these conditions, the program will loop and continue to ask the user to enter in his/her name. The name can't have spaces because when saved, the format is Name 10 (where 10 is the score) and the score and name are separated by a space. See Figure 5A and 5B.

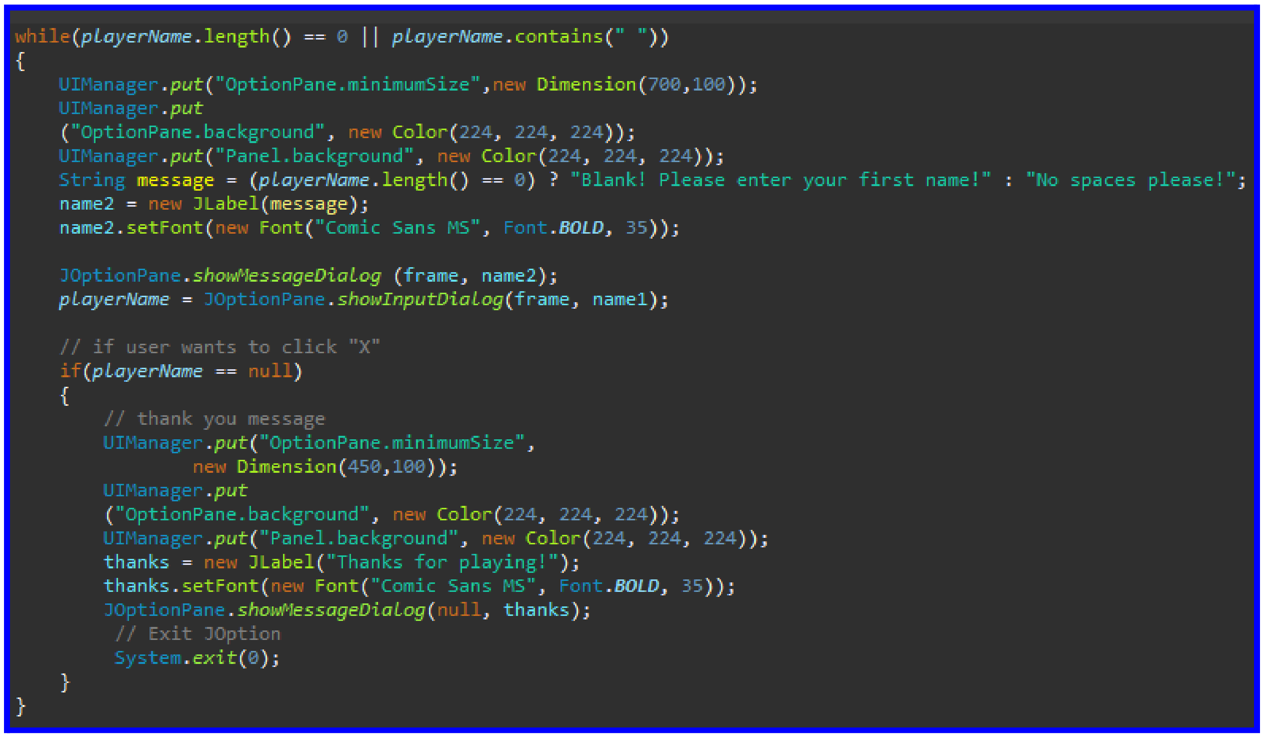
**Figure 5A**

// If the user clicks the "X" or cancel button, a Thank you message will appear and the program will exit.



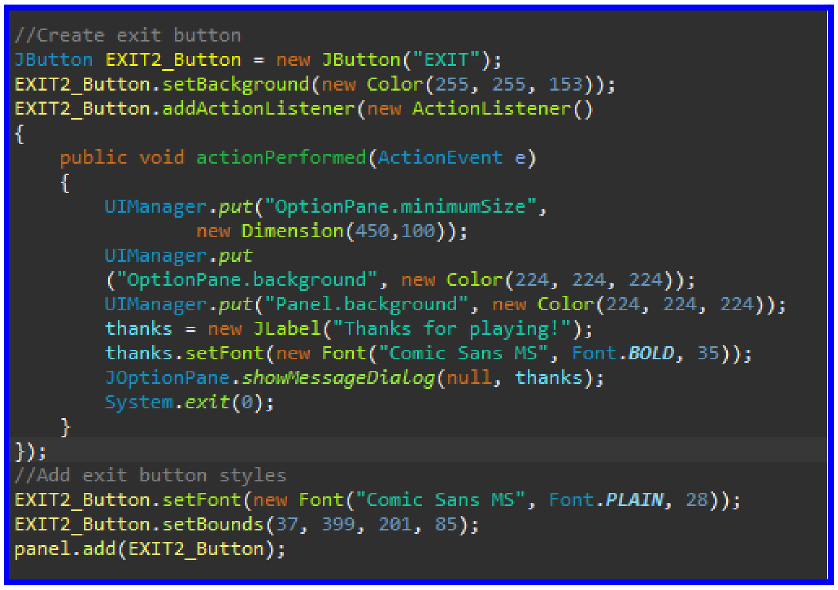
```
//Use JOptionPane to get the user's name
playerName = JOptionPane.showInputDialog(frame, name1);

//Validate the name that the name the user inputted is an acutal name
if (playerName == null)
{
    UIManager.put("OptionPane.minimumSize",
            new Dimension(450,100));
    UIManager.put
    ("OptionPane.background", new Color(224, 224, 224));
    UIManager.put("Panel.background",
            new Color(224, 224, 224));
    thanks = new JLabel("Thanks for playing!");
    thanks.setFont(new Font("Comic Sans MS", Font.BOLD, 35));
    JOptionPane.showMessageDialog(null, thanks);
    System.exit(0);
}
```

**Figure 5B**



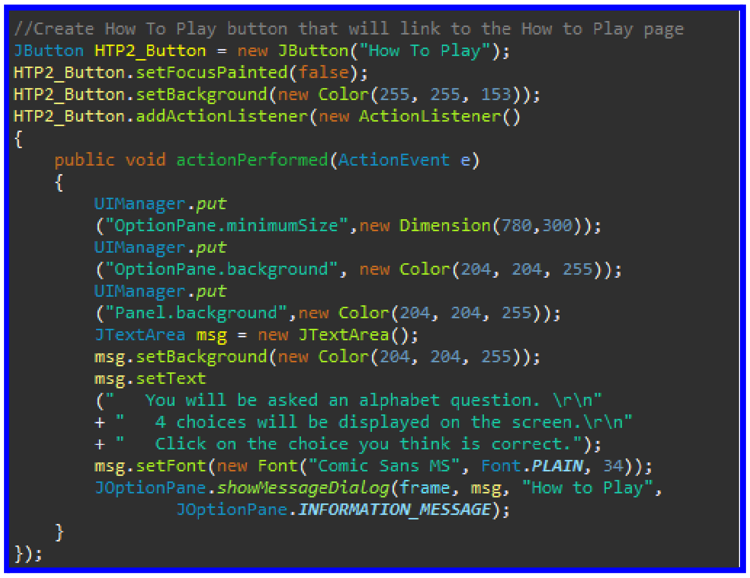
```
while(playerName.length() == 0 || playerName.contains(" "))
{
    UIManager.put("OptionPane.minimumSize",new Dimension(700,100));
    UIManager.put
    ("OptionPane.background", new Color(224, 224, 224));
    UIManager.put("Panel.background", new Color(224, 224, 224));
    String message = (playerName.length() == 0) ? "Blank! Please enter your first name!" : "No spaces please!";
    name2 = new JLabel(message);
    name2.setFont(new Font("Comic Sans MS", Font.BOLD, 35));

    JOptionPane.showMessageDialog (frame, name2);
    playerName = JOptionPane.showInputDialog(frame, name1);

    // if user wants to click "X"
    if(playerName == null)
    {
        // thank you message
        UIManager.put("OptionPane.minimumSize",
                new Dimension(450,100));
        UIManager.put
        ("OptionPane.background", new Color(224, 224, 224));
        UIManager.put("Panel.background", new Color(224, 224, 224));
        thanks = new JLabel("Thanks for playing!");
        thanks.setFont(new Font("Comic Sans MS", Font.BOLD, 35));
        JOptionPane.showMessageDialog(null, thanks);
         // Exit JOption
         System.exit(0);
    }
}
```

Next is a bunch of java layout code generated by WindowBuilder. Action listeners are added to exit and How to Play buttons to do their corresponding tasks. The exit button terminates the program.



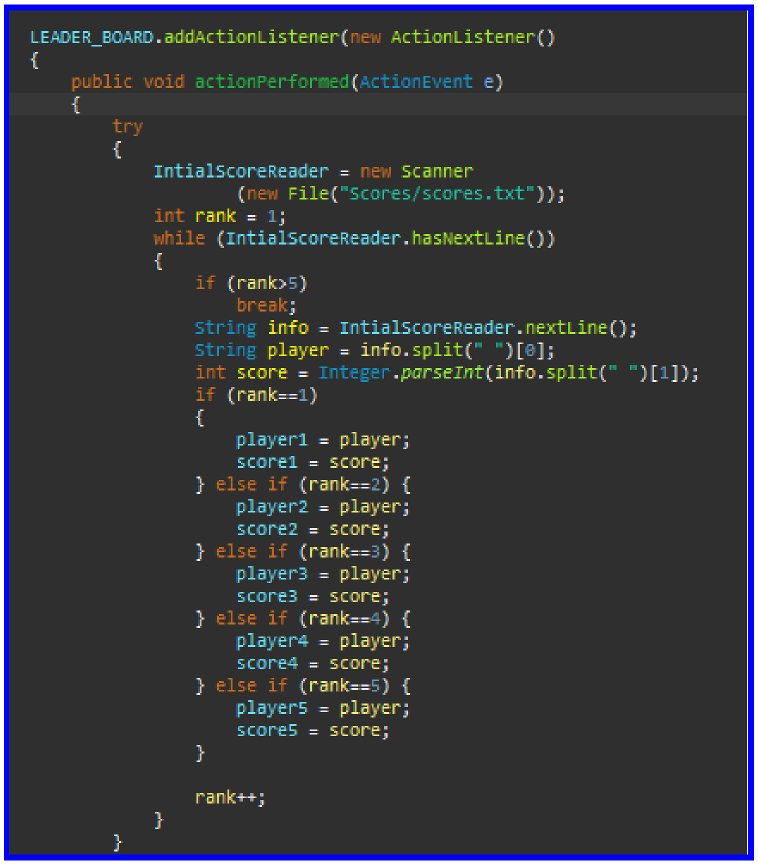
```
//Create exit button
JButton EXIT2_Button = new JButton("EXIT");
EXIT2_Button.setBackground(new Color(255, 255, 153));
EXIT2_Button.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        UIManager.put("OptionPane.minimumSize",
                new Dimension(450,100));
        UIManager.put
        ("OptionPane.background", new Color(224, 224, 224));
        UIManager.put("Panel.background", new Color(224, 224, 224));
        thanks = new JLabel("Thanks for playing!");
        thanks.setFont(new Font("Comic Sans MS", Font.BOLD, 35));
        JOptionPane.showMessageDialog(null, thanks);
        System.exit(0);
    }
});
//Add exit button styles
EXIT2_Button.setFont(new Font("Comic Sans MS", Font.PLAIN, 28));
EXIT2_Button.setBounds(37, 399, 201, 85);
panel.add(EXIT2_Button);
```

The How to Play button opens a JOptionPane message dialog.



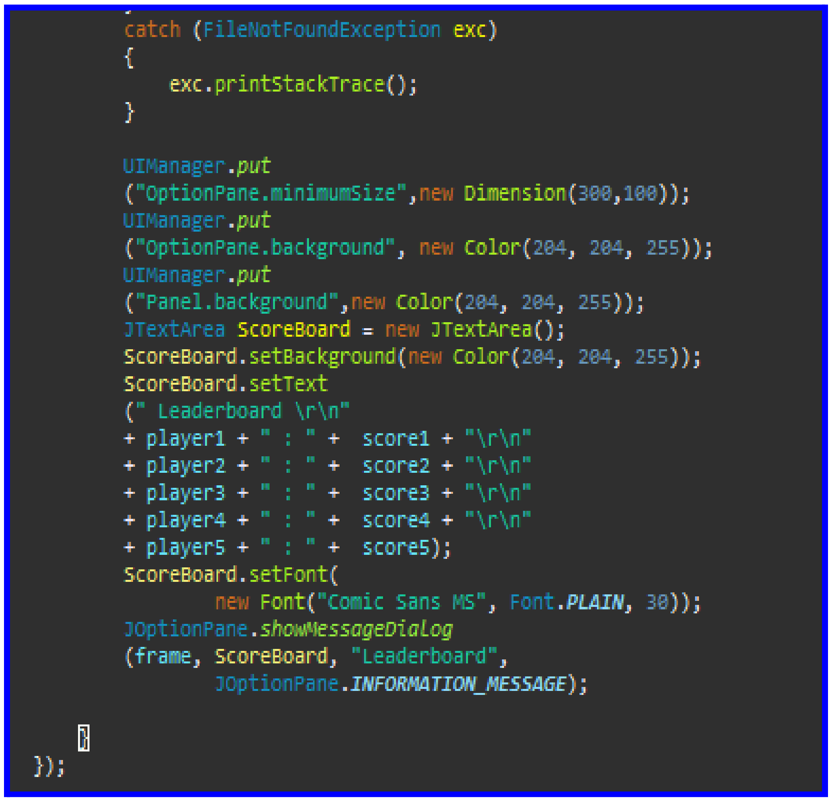
```
//Create How To Play button that will link to the How to Play page
JButton HTP2_Button = new JButton("How To Play");
HTP2_Button.setFocusPainted(false);
HTP2_Button.setBackground(new Color(255, 255, 153));
HTP2_Button.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        UIManager.put
        ("OptionPane.minimumSize",new Dimension(780,300));
        UIManager.put
        ("OptionPane.background", new Color(204, 204, 255));
        UIManager.put
        ("Panel.background",new Color(204, 204, 255));
        JTextArea msg = new JTextArea();
        msg.setBackground(new Color(204, 204, 255));
        msg.setText
        ("   You will be asked an alphabet question. \r\n"
        + "   4 choices will be displayed on the screen.\r\n"
        + "   Click on the choice you think is correct.");
        msg.setFont(new Font("Comic Sans MS", Font.PLAIN, 34));
        JOptionPane.showMessageDialog(frame, msg, "How to Play",
                JOptionPane.INFORMATION_MESSAGE);
    }
});
```

The leader board button fetches top 5 from the leaderboard text file, opens a JOptionPane message dialog, and displays the top 5. See here below and also see Figure 6B.



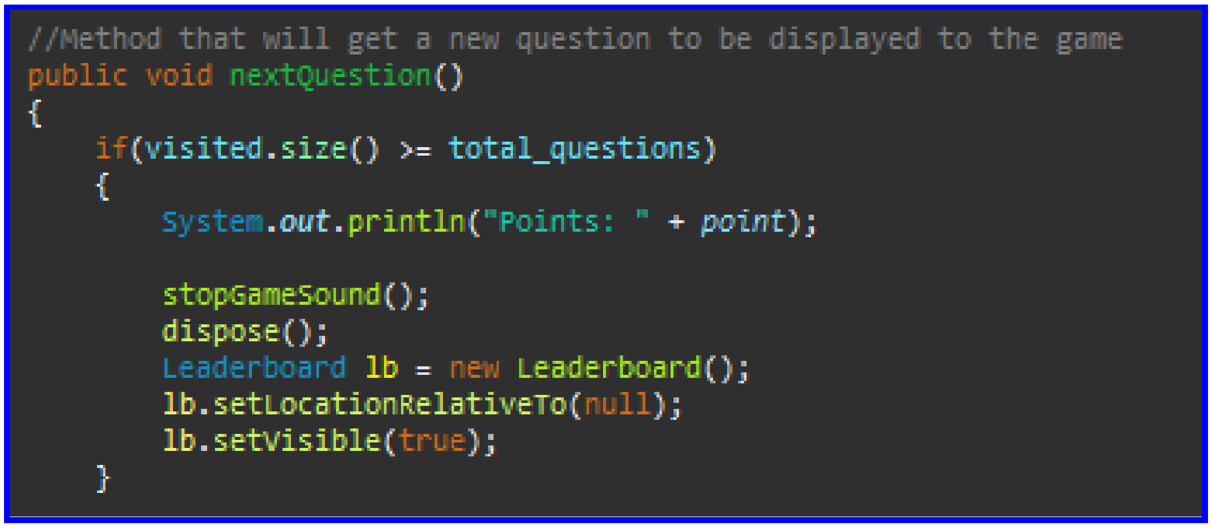
```
LEADER_BOARD.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            IntialScoreReader = new Scanner
                    (new File("Scores/scores.txt"));
            int rank = 1;
            while (IntialScoreReader.hasNextLine())
            {
                if (rank>5)
                    break;
                String info = IntialScoreReader.nextLine();
                String player = info.split(" ")[0];
                int score = Integer.parseInt(info.split(" ")[1]);
                if (rank==1)
                {
                    player1 = player;
                    score1 = score;
                } else if (rank==2) {
                    player2 = player;
                    score2 = score;
                } else if (rank==3) {
                    player3 = player;
                    score3 = score;
                } else if (rank==4) {
                    player4 = player;
                    score4 = score;
                } else if (rank==5) {
                    player5 = player;
                    score5 = score;
                }

                rank++;
            }
        }
```

Figure 6B

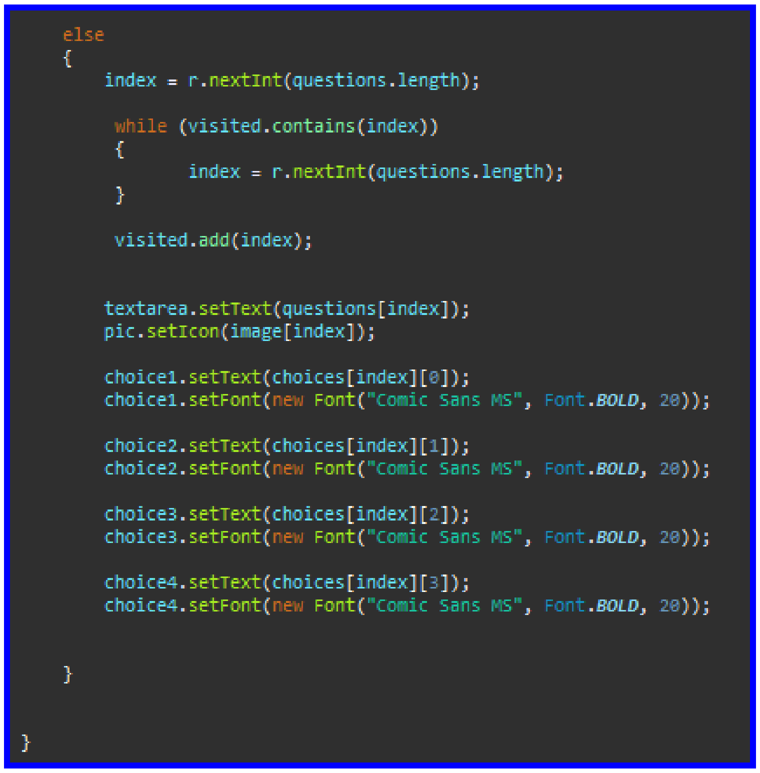


```
        catch (FileNotFoundException exc)
        {
            exc.printStackTrace();
        }

        UIManager.put
        ("OptionPane.minimumSize",new Dimension(300,100));
        UIManager.put
        ("OptionPane.background", new Color(204, 204, 255));
        UIManager.put
        ("Panel.background",new Color(204, 204, 255));
        JTextArea ScoreBoard = new JTextArea();
        ScoreBoard.setBackground(new Color(204, 204, 255));
        ScoreBoard.setText
        (" Leaderboard \r\n"
        + player1 + " : " +  score1 + "\r\n"
        + player2 + " : " +  score2 + "\r\n"
        + player3 + " : " +  score3 + "\r\n"
        + player4 + " : " +  score4 + "\r\n"
        + player5 + " : " +  score5);
        ScoreBoard.setFont(
                new Font("Comic Sans MS", Font.PLAIN, 30));
        JOptionPane.showMessageDialog
        (frame, ScoreBoard, "Leaderboard",
                JOptionPane.INFORMATION_MESSAGE);

    }
});
```

The nextQuestion method checks if the player is on their last question. If the player is finished with Question 10, it brings them to the leaderboard. Otherwise it fetches and displays the next question.

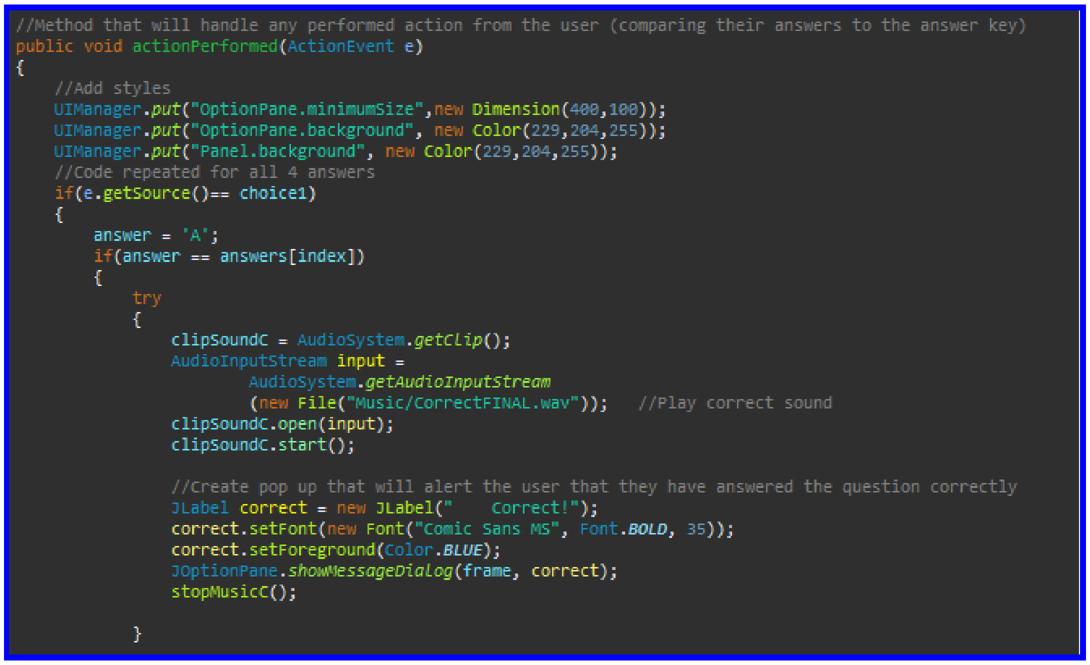


```
//Method that will get a new question to be displayed to the game
public void nextQuestion()
{
    if(visited.size() >= total_questions)
    {
        System.out.println("Points: " + point);

        stopGameSound();
        dispose();
        Leaderboard lb = new Leaderboard();
        lb.setLocationRelativeTo(null);
        lb.setVisible(true);
    }
```



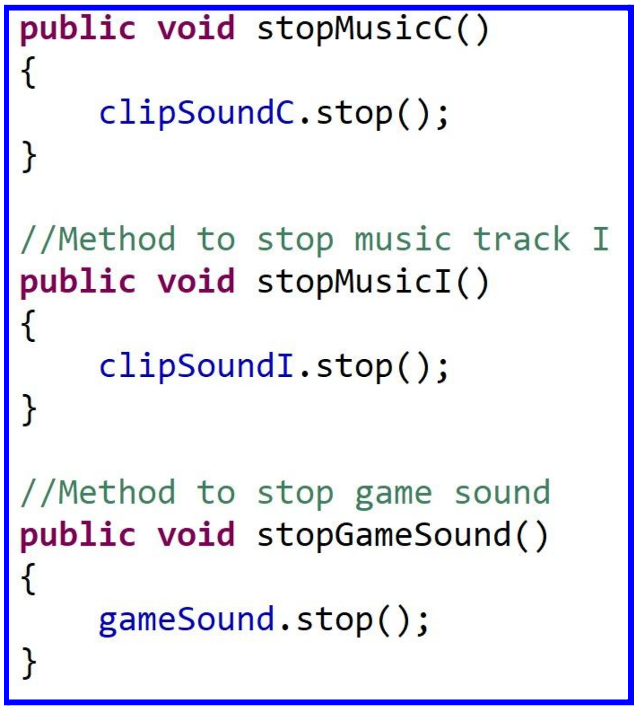
```
    else
    {
        index = r.nextInt(questions.length);

         while (visited.contains(index))
         {
                index = r.nextInt(questions.length);
         }

         visited.add(index);


        textarea.setText(questions[index]);
        pic.setIcon(image[index]);

        choice1.setText(choices[index][0]);
        choice1.setFont(new Font("Comic Sans MS", Font.BOLD, 20));

        choice2.setText(choices[index][1]);
        choice2.setFont(new Font("Comic Sans MS", Font.BOLD, 20));

        choice3.setText(choices[index][2]);
        choice3.setFont(new Font("Comic Sans MS", Font.BOLD, 20));

        choice4.setText(choices[index][3]);
        choice4.setFont(new Font("Comic Sans MS", Font.BOLD, 20));


    }

}
```

The actionPerformed method is called when a user answers a question. The ActionEvent argument is used to track which button was clicked. It then checks if the question was answered correctly and plays sound(correct sound if correct, incorrect sound if not correct).



```
//Method that will handle any performed action from the user (comparing their answers to the answer key)
public void actionPerformed(ActionEvent e)
{
    //Add styles
    UIManager.put("OptionPane.minimumSize",new Dimension(400,100));
    UIManager.put("OptionPane.background", new Color(229,204,255));
    UIManager.put("Panel.background", new Color(229,204,255));
    //Code repeated for all 4 answers
    if(e.getSource()== choice1)
    {
        answer = 'A';
        if(answer == answers[index])
        {
            try
            {
                clipSoundC = AudioSystem.getClip();
                AudioInputStream input =
                        AudioSystem.getAudioInputStream
                        (new File("Music/CorrectFINAL.wav"));   //Play correct sound
                clipSoundC.open(input);
                clipSoundC.start();

                //Create pop up that will alert the user that they have answered the question correctly
                JLabel correct = new JLabel("    Correct!");
                correct.setFont(new Font("Comic Sans MS", Font.BOLD, 35));
                correct.setForeground(Color.BLUE);
                JOptionPane.showMessageDialog(frame, correct);
                stopMusicC();

            }
```

Finally, the last 3 methods are used to stop different music tracks. The methods stopMusicC and stopMusicI refer to the stopping of the correct sound effect and incorrect sound effect respectively.



```
public void stopMusicC()
{
    clipSoundC.stop();
}

//Method to stop music track I
public void stopMusicI()
{
    clipSoundI.stop();
}

//Method to stop game sound
public void stopGameSound()
{
    gameSound.stop();
}
```

**Conclusion**

Despite having faced some challenges such as having meetings on days that fit with everyone's schedule, coding the questions page, and getting the leaderboard to work, this project was fun. All group members were able to learn a lot from it.

## Sources

### Pictures

- Welcome Page image: https://dlpng.com/png/3990640
- Thinking emoji: https://hotemoji.com/images/dl/5/thinking-emoji-by-twitter.png
- Apple: https://ih1.redbubble.net/image.1135321471.0303/flat,750x,075,f-pad,750x1000,f8f8f8.jpg
- Hello:
  - https://supersimple.com/wp-content/uploads/2017/02/singalongwithtobee_ep101_youtube-taf_en-1200w-blog-300x169.jpg
  - https://supersimple.com/song/hello-hello/
- Song: https://www.istockphoto.com/vector/girl-singing-with-microphone-singer-character-cartoon-vector-illustration-gm911743842-251022876
- Dog: https://www.goodhousekeeping.com/life/pets/g4531/cutest-dog-breeds/
- Nap: https://www.todaysparent.com/kids/kids-health/snooze-control-an-age-by-age-guide-to-naps/
- Red: https://www.kindpng.com/imgv/iobRmJx_red-cliparts-splatter-red-color-splash-clipart-hd
- Pencil: https://www.pinterest.com/pin/633248397597141348/
- Cat: https://wallpapersafari.com/w/z2wGcI
- Sunflower: https://pngtree.com/freepng/smiling-sunflower-flower-illustration_4508080.html
- Broccoli: https://chefsmandala.com/wp-content/uploads/2018/03/Broccoli.jpg
- Bells: https://creazilla.com/nodes/77429-jingle-bells-clipart
- Christmas Tree: https://starpng.com/photo/2964/christmas-tree-transparent-background-png-image

### Background Music

- ABC Song: https://www.singing-bell.com/abc-song-alphabet-for-kids/
- Bingo Song: https://www.singing-bell.com/bingo-song-mp3/
- Jingle Bells Song : https://incompetech.com/music/royalty-free/index.html?isrc=USUAN1100187#google_vignette
- We Wish You A Merry Christmas: https://www.karaoke-version.com/free/christmas-carol/we-wish-you-a-merry-christmas.html

### Sound Effects

- Correct Sound: https://www.myinstants.com/instant/kids-cheering-yay/
- Incorrect Sound: https://www.myinstants.com/instant/aww/