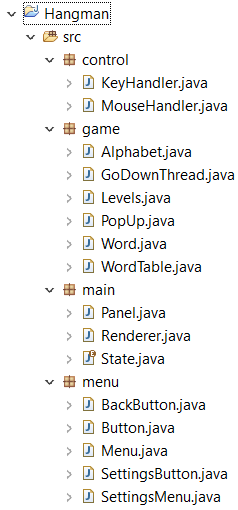
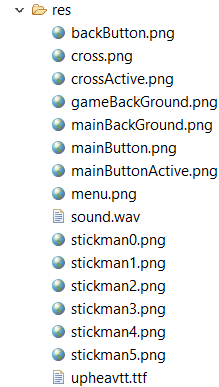
## Criterion C: Development

All Classes:



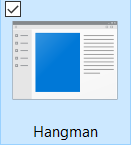
Resources Folder:



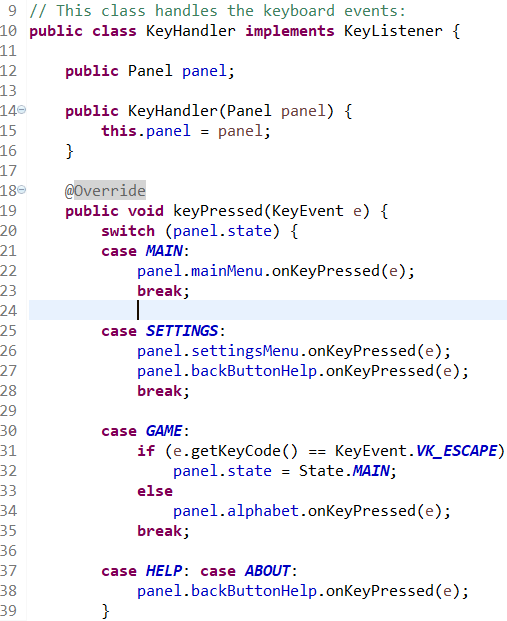
List of Techniques:

* Inheritance Object Oriented Programming Concept Used
* Implementation technique is used
* Static methods and variables
* Different methods are used in the program, all methods are created by me
* String arX
* 2D Graphics

Program File:

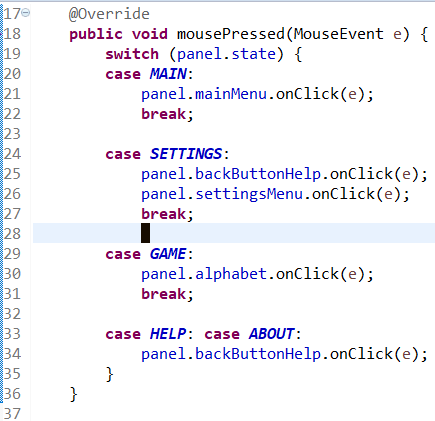


Key Handler:



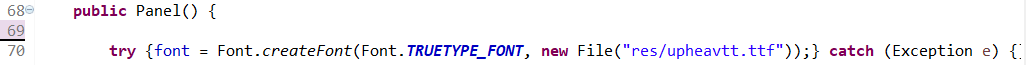
This code was written to make the keyboard compatible with the game functions. I used **@Override** annotation as it acts as a compile-time safeguard against a common programming errors. I also used **implements** so I could refer the interface using the implements keyword. The program is required to provide method logic for all non-default methods.

Mouse Handler:



This code was written to make the mouse compatible with the game functions. It has the same key features of the Key Handler as mentioned above.

Main Window:



In this constructor we are loading our resources, and this includes the custom font being load. The code after the **try** block will be attempted to be run. If the exception in the **catch** statement is thrown during the code in the try block is run, run the code in the catch block.



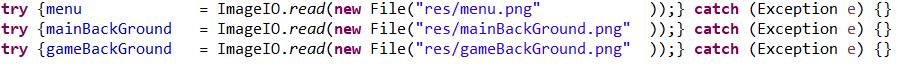
This code loads the Cross and Cross Active images which are used in game representing the amount of lives they have in game;



Figure 1 Cross.png



Figure 2 crossActive.png



This code loads the Background and Menu images;

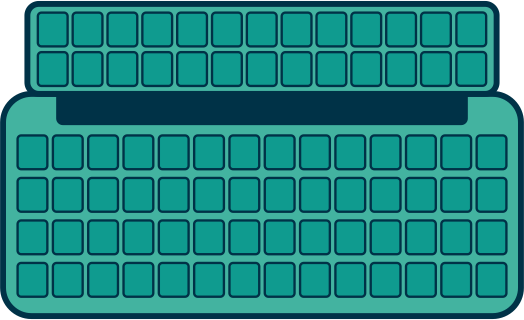


Figure 3 menu.png

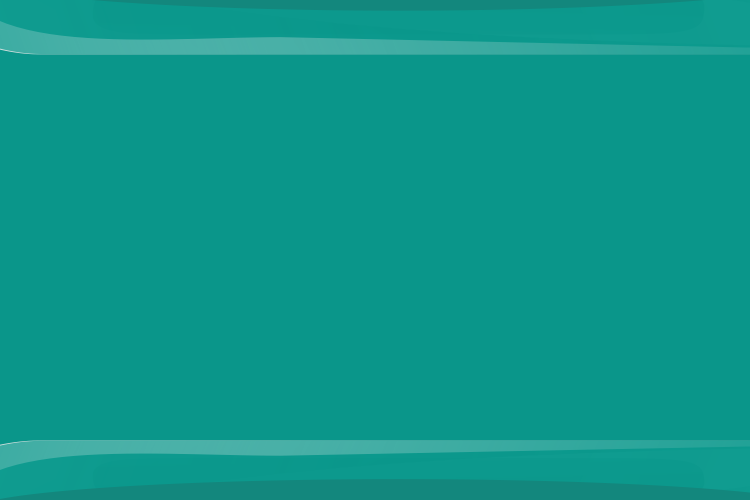


Figure 4 mainBackGround.png



Figure 5 gameBackGround.png



This code loads the buttons founds in the Main Menu;



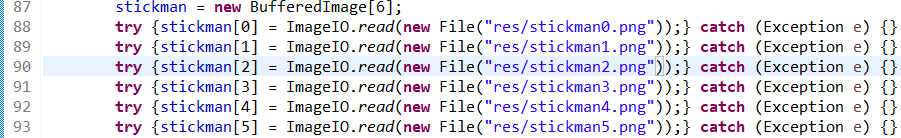
Figure 6 mainButton.png



Figure 7 mainButtonActive.png



Figure 8 backButton.png



This is a stickman arX which loads their images upon lives taken and an image of their body is formed;

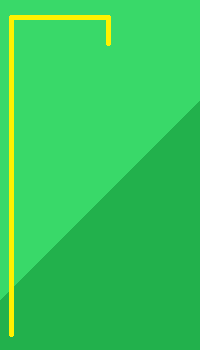


Figure 9 stickman0.png

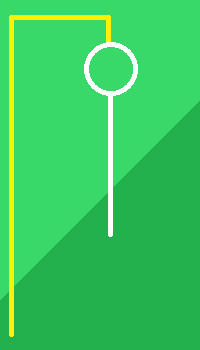


Figure 10 stickman1.png

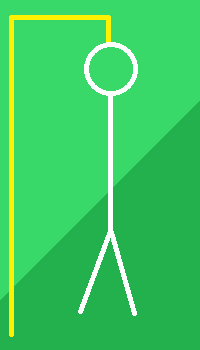


Figure 11 stickman2.png

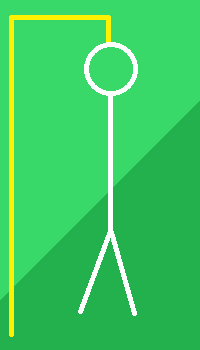


Figure 12 stickman3.png

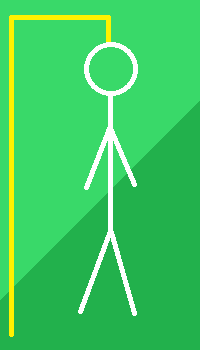


Figure 13 stickman4.png

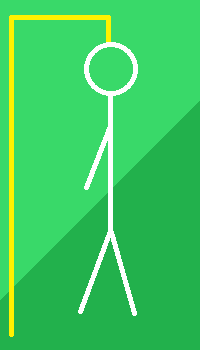
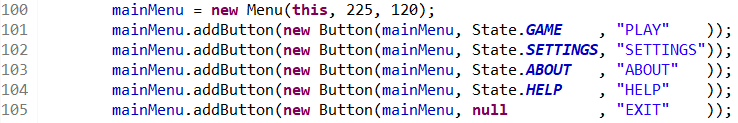
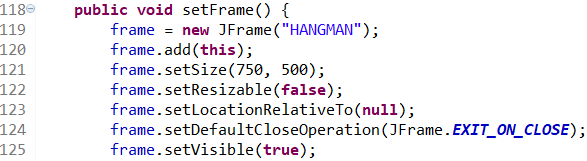


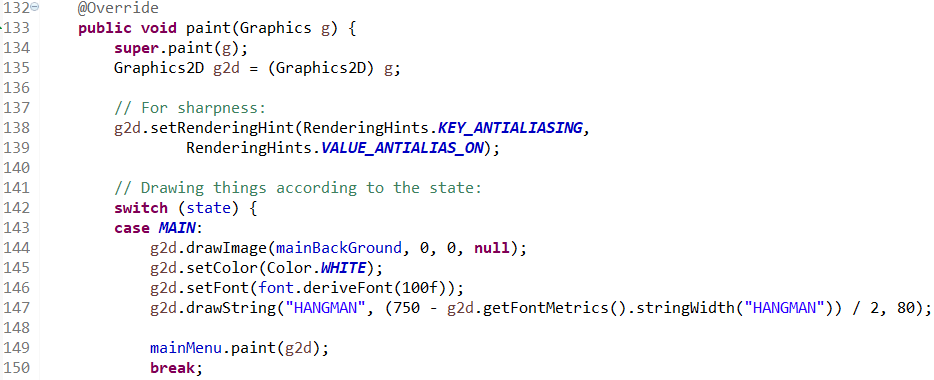
Figure 14 stickman5.png



This code is written to create the main menu and buttons have been added to them.



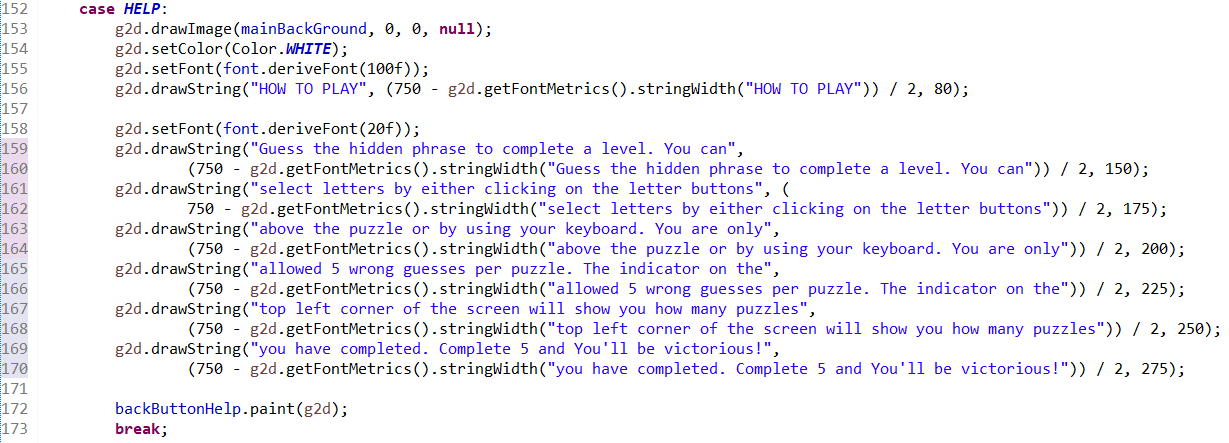
This method initializes the frame object, size and all the relevant features.



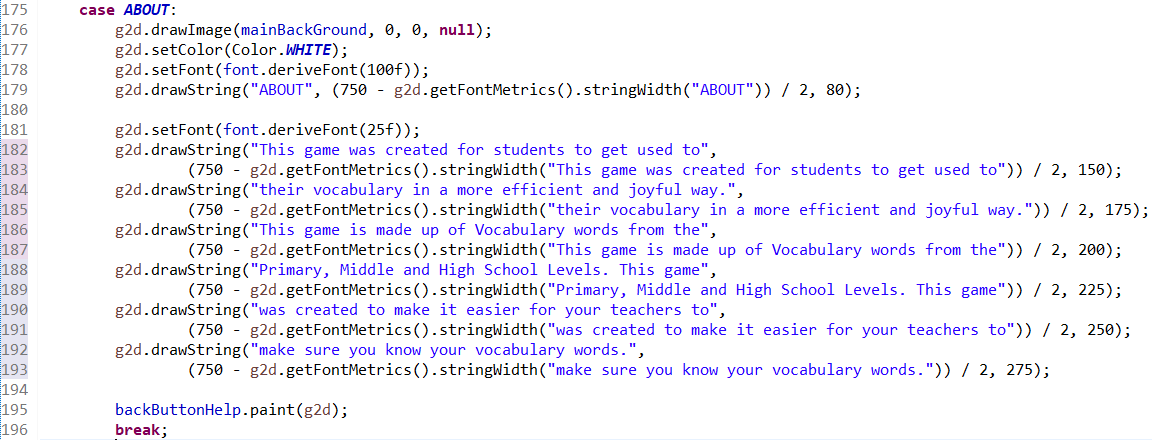
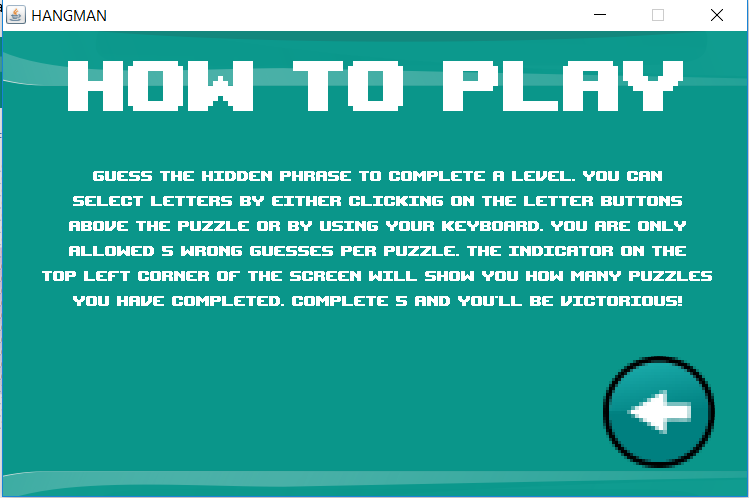
The following is a code for graphics to execute the word “Hangman” on the JFrame in white color with the derived font. The **switch** statement is allowing the variable to be tested for equality against a list of values, where each value is called **case**.



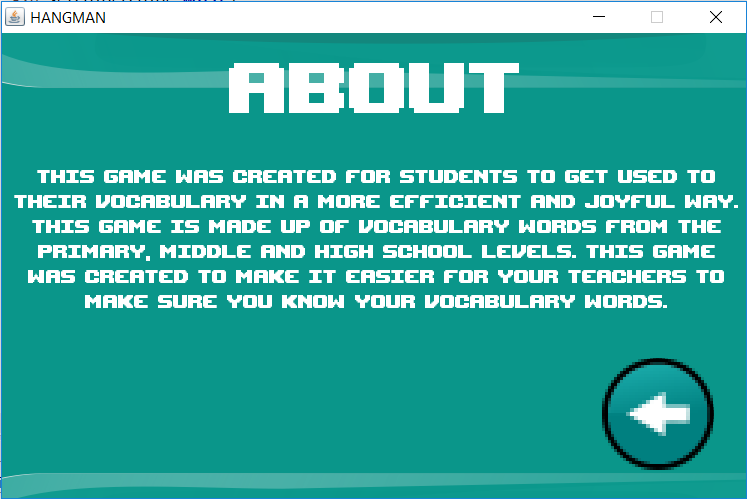
Figure 15 HANGMAN text



The following code is written to construct an image like the “Hangman” on the Main Menu with some context and back button.

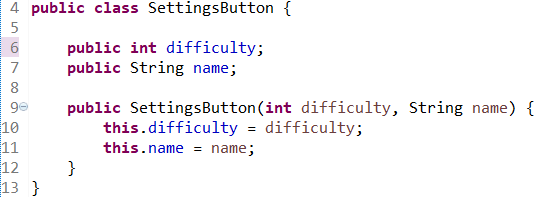


This code is written the same as the HELP but different context.

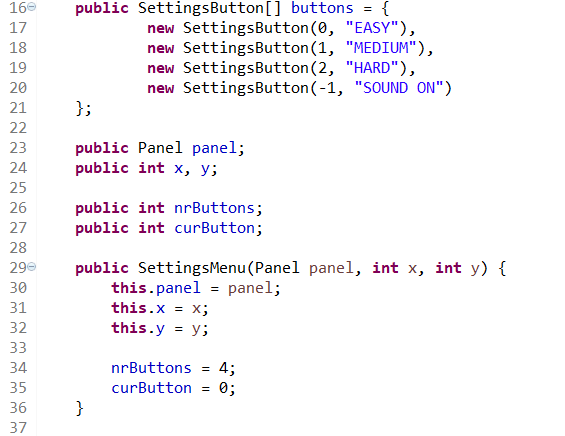


The following code is written for the text to fix the context of the others but has a little complexity with a combination of two other classes for a follow up.

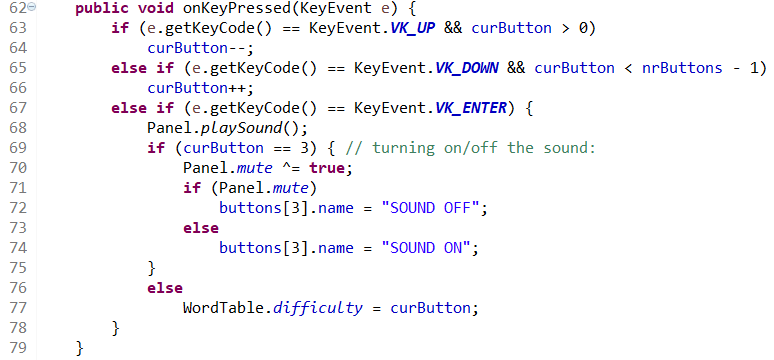
The method stating playsound() is responsible for the audio to be executed.



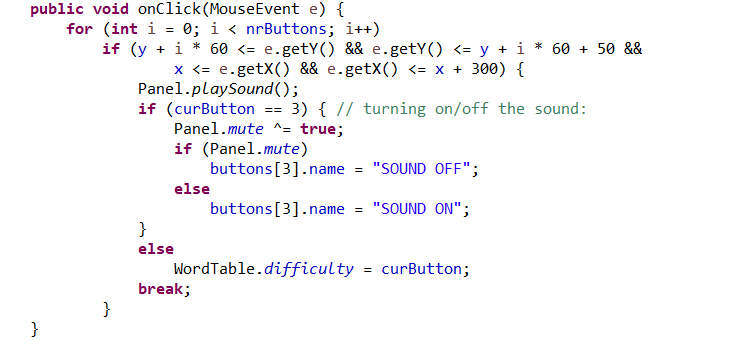
Here’s a simple class called Settings Button with information about the buttons from the settings menu where after the button is pressed, the difficulty changed to that.



Those are the buttons for the settings menu



This is the code that is written to help the Keyboard interact with the sound button to toggle on and off



The same idea with the mouse on click.

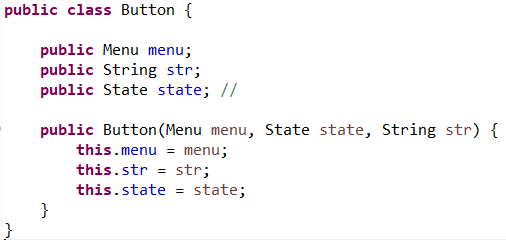


Figure 16 Toggle ON

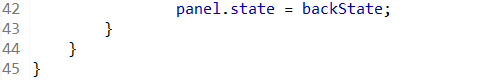
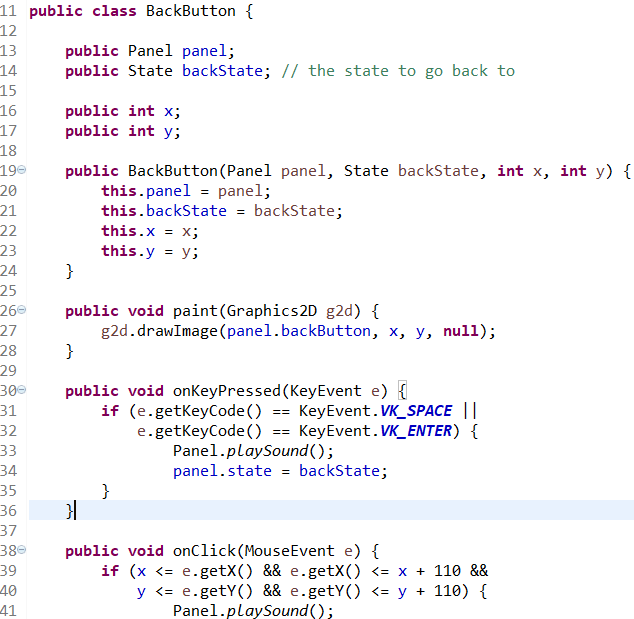


Figure 17 Toggle OFF

Buttons:

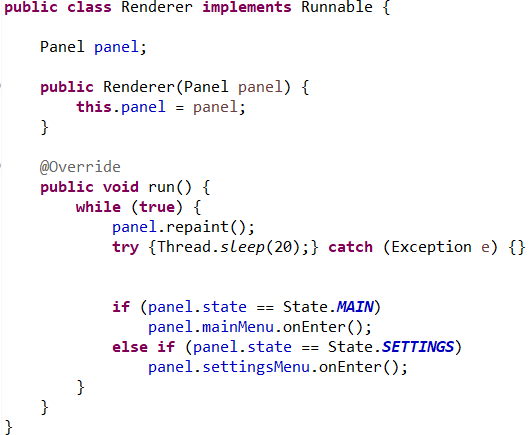


Referring to Figure 6 and Figure 7, Here’s a simple Button class with information about the buttons from the main menu it also shows the string that is displayed on the button the game state will be changed when the button is pressed.



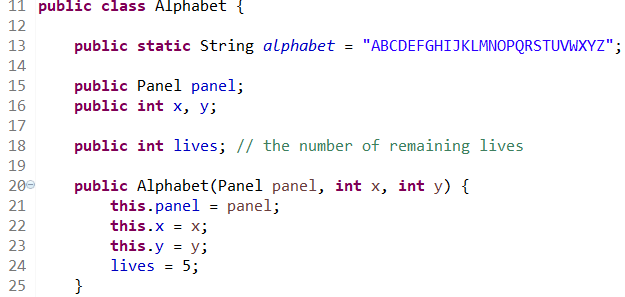
Referring to Figure 8, This shows the functionality of the Back Button and how it is executed. Upon the button pressed it should play a sound. The code written also shows the compatibility with the Keyboard and Mouse.

Rendering:

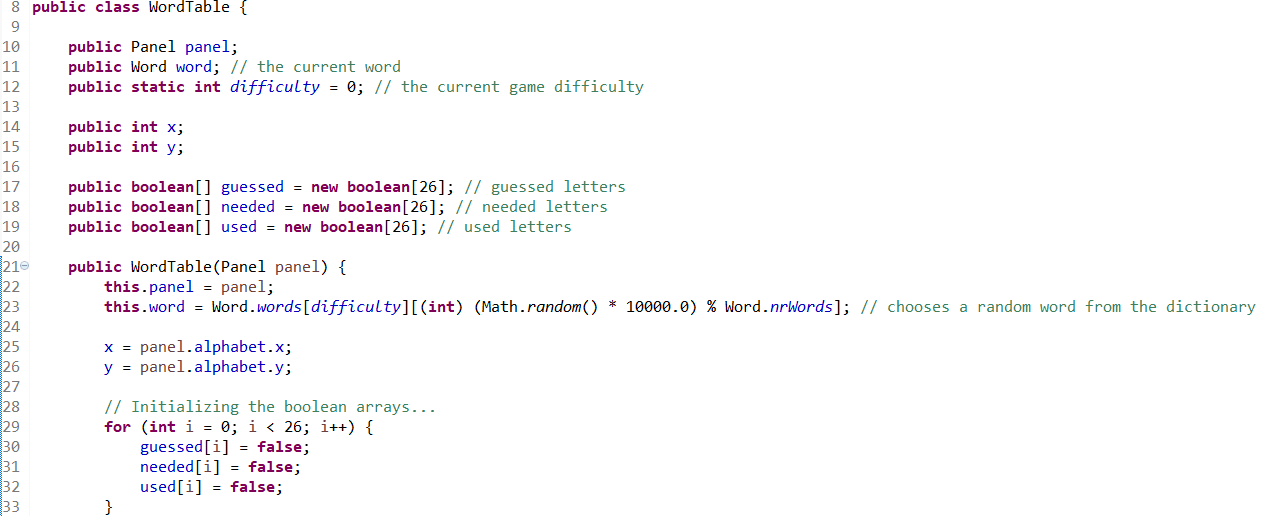


Here’s the thread class that repaints the panel and receives signals from mouse all the time followed up by checking the x and y of the mouse to update the buttons from menus. The Renderer Class is used for executing Buttons mainly for the Main Panel Class.

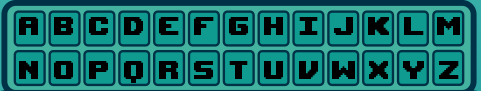
Game Screen:



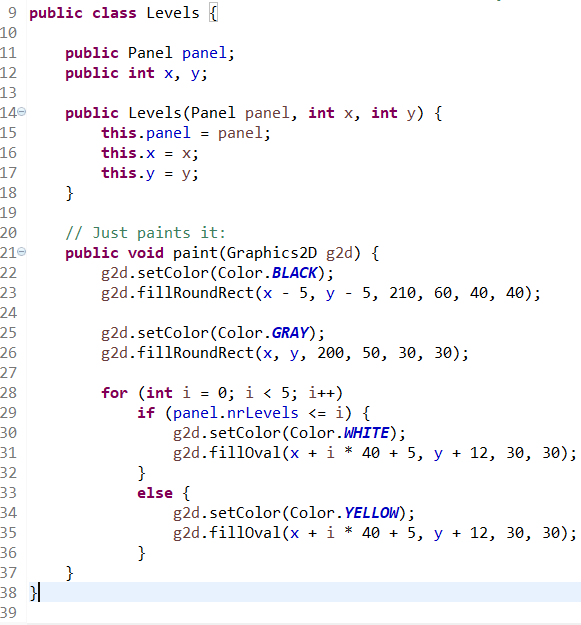
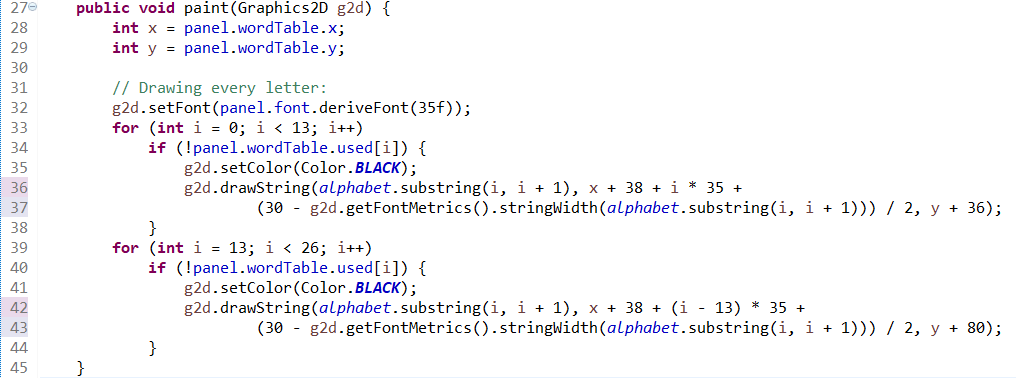
In the class Alphabet constructor 5 lives are given to construct the object



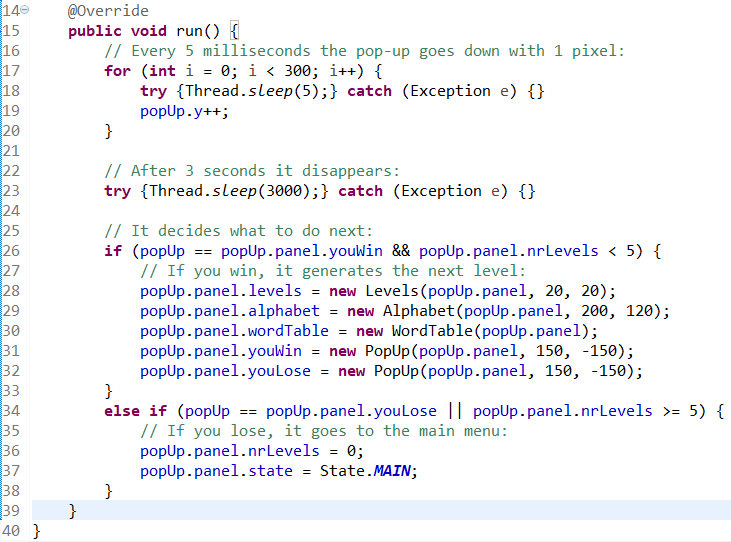
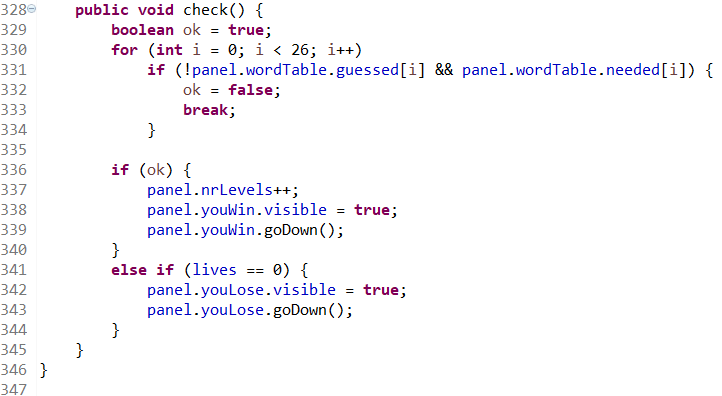
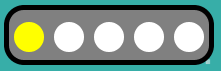
The Word Table shows an ArX of 26 alphabets being executed on the table set off as Booleans.



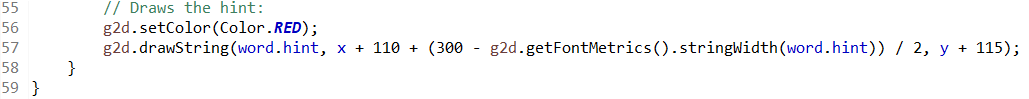
The following shows every letter colored in black and a word table drawn.



The following is the code written for the 5 stages of each level colored and painted.



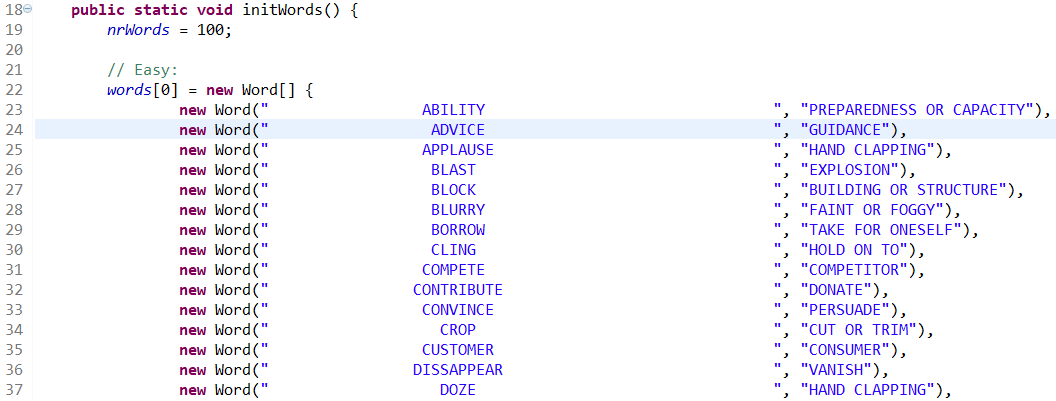
The following code shows the code executed to check whether the game has ended or not. If you have lost, you lose a life/return to main menu if not you proceed to the next stage.



The following code shows how the hint is drawn followed up by each word in the dictionary.



Here’s the sample of the word list **Easy** Difficulty through the method initWords()

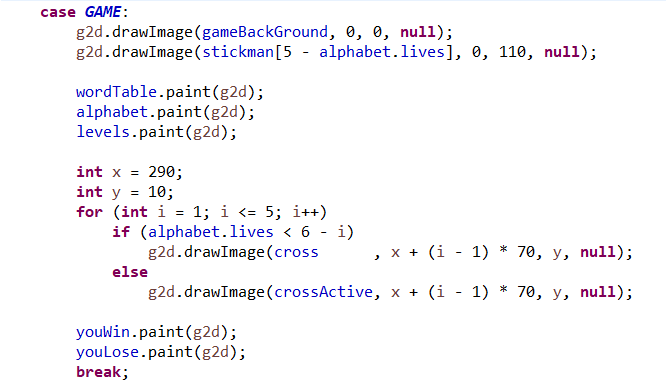


Here’s a sample of Medium Difficulty level



Here’s a sample of Hard Difficulty level

Stickman:



Referring to Figure 9 to Figure 14,   
stickman[0] is shown if you have 5 lives

stickman[1] is shown if you have 4 lives

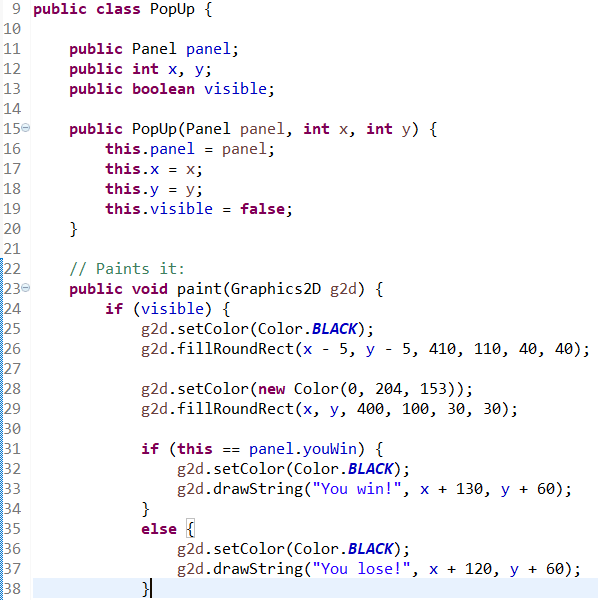
stickman[2] is shown if you have 3 lives

stickman[3] is shown if you have 2 lives

stickman[4] is shown if you have 1 lives

stickman[5] is shown if you have 0 lives

The sum of the index and the number of lives is always 5 (invariant).



Here’s a class about the popup (You win! or (You lose!)

