1. **Formulating the Problem**

**1.1 Problem Description**

Design and implement a Hangman game using Java. The program should generate a random word and allow the user to enter one letter at a time to guess the word. If the letter is correct the program should display the letter in the appropriate index of the word. If the letter is incorrect, a body part should be added to the hangman.

**1.2** **Verbalization**

What is the goal?

Build a Hangman game using Java.

What are the givens?

The letters of the word

What are the unknowns?

If the user wins or loses the game.

**1.3 Information Elicitation**

Goal Hangman game

Givens Letters

Unknowns Win or Lose

Condition The user must guess all the correct letters before the hangman is completely build.

**2. Planning the Solution**

**2.1. Solution Strategy**

Randomly generate a word and store each letter of the word as an element in an array, display the word as asterisks. Allow the user to make a guess and store it in a variable. Then check if that letter is in the letter's array. If it is display the letter in the appropriate index of the word. Otherwise, display a body part of the hangman. To generate a new game, reinitialize the global variables to their original state.

**2.2 Goal Decomposition**

Sub-goal 1

Randomly generate the word and display it.

Sub-goal 2

Get letter from user.

Sub-goal 3

Check if the letter is in the word.

Sub-goal 4

Display the results.

Sub-goal 5

Display a win/lose message.

Sub-goal 6

Ask user for a new game.

**2.4 Data Organization and Description**

**Inputs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Origin** | **Used in Sub-goal #** |
| selectedChar | Letter guessed by the user | User | 2 |

**Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Origin** | **Used in Sub-goal #** |
| lettersFound[] | Array to display the word to the screen | Screen | 4 |
| displayMan | Win/Lose message and Hangman body | Screen | 5 |

**3. Designing the Solution**

**3.1 Structure Chart**

First Level Decomposition

Goal Refinement

**Sub-goal 1**

Create an array of words.

**Sub-goal 1.1**

Randomly pick a word from the words array.

**Sub-goal 1.2**

Create another array that stores the letters of the active word.

**Sub-goal 1.3**

Create another array that stores an asterisk for each letter of the active word.

**Sub-goal 2**

Display the word on the screen.

**Sub-goal 3**

Get input from the user.

**Sub-goal 3.1**

Store the input.

**Sub-goal 4**

Compare each index of the letters array to the letter entered by the user.

**Sub-goal 4.1**

If an index of the letters array is equal to the letter entered by the user, store that letter in the same index of the asterisks array.

**Sub-goal 5**

Display the results.

**Sub-goal 5.1**

If the letter entered by the user was found in the letters array, display the updated asterisks array on the screen.

**Sub-goal 5.2**

If the letter entered by the user was not found in the letters array, increase the number of bad guesses count and display a body part of the hangman.

**Sub-goal 6**

Display a win or lose message.

**Sub-goal 6.1**

If the user enters all the correct words before the hangman is completed display a

you win message.

**Sub-goal 6.2**

If the number of bad guesses is equal to 6 display a you lose message.

**Sub-goal 7**

Give the user an option to play a new game.

Second Level Decomposition

**3.2 Module and Data Specifications**

**Name:** setCurrentWordLetters - create an array that stores letters of the active word.

**Input:** None.

**Output:** None.

**Logic:** Create an array that stores letters of the active word. Use a loop to store each letter of the word in the array.

**Name:** setLettersFound- create an array that stores asterisk for each letter of the word.

**Input:** None.

**Output:** None.

**Logic:** Create an array of the same length as the length of the active word.

**Name:** getCurrentWord - return the active word .

**Input:** None.

**Output:** currentWord.

**Logic:** Return the variable that stores the active word.

**Name:** getAsterisk- return asterisks for each letter of the word.

**Input:** None.

**Output:** asterisk.

**Logic:** Create a variable and use a loop to append asterisks for each letter of the word.

**Name:** displayChar - display the results if the user guessed the correct letter.

**Input:** ch, ch1.

**Output:** returnWord.

**Logic:** Get the letter guessed by the user and use a loop to compare it to each index of the array that stores the letters of the active word. If there is a match, update that same index of the asterisks array with the letter guessed by the user.

**Name:** displayHangman - display a body part of the hangman.

**Input:** None.

**Output:** displayMan.

**Logic:** Use a selection control to check the number of wrong guesses and accordingly display the body part of the hangman.

**Name:** newGame- set variables for a new game.

**Input:** None.

**Output:** None.

**Logic:** Reinitialize all global variables to their original state.

**Data:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Structure** |
| **numWrongGuess** | **Integer** | **Variable** |
| **Index** | **Integer** | **Variable** |
| **Words** | **String** | **Array** |
| **lettersFound** | **String** | **Array** |
| **currentWordLetters** | **Char** | **Array** |
| **currentWord** | **String** | **Variable** |
| **gameOver** | **Boolean** | **Variable** |

**3.3 Algorithm**

Logic

1.0 Create an array of words.

1.1 Pick a word.

1.2 Create an array of letters.

1.3 Create an array of asterisk.

2.0 Display the word on the screen.

3.0 Get input .

3.1 Store the input.

4.0 Compare the input to the letters array.

4.1 If the user input is in the word, store that letter in appropriate index of the asterisks array.

5.0 Display the result.

5.1 If the letter was found then display the updated asterisks array.

5.2 If the letter was not found then display a body part of the hangman.

6.0 Display a win or lose message.

6.1 Display a you win message if the user guesses the word before the hangman is build .

6.2. Display a you lose message if the hangman is build before the user guesses the word.

7.0 Promote the user for a new game;

Algorithm Description

The algorithm for the Hangman game is very straight forward. The program starts by displaying the user interface to the user. In the background a new object of the Game class is created. The Game class creates three major arrays: words, lettersFound and currentWordLetters. The words array contains words to be used in the game. A random number is generated which represents one random index of the words array. The word in that specific index of the words array is used for the game. Once the word to be used in the game is determined, a loop is used to add each letter of the word to each index of the currentWordLetters. The lettersFound array is the same size as the currentWordLetters and stores asterisks in each index.

The word is displayed to the user and the program waits for the user to select a letter by clicking a button. Once the user clicks a button representing a letter, that letter is used to check against the currentWordLetters to determine if the letter is in the currentWordLetters array. If a match is found then the appropriate index of the lettersFound array is updated with the letter selected by the user. The lettersFound array is displayed to the user.

This process continues either until the user guesses the word correctly before the hangman is build or until the hangman is build completely (all 6 body parts).

**4 Translation**

**4.1 Source Code**

//===================================================

// Name : Nidhi Patel

// SID : 31379144

// Course : IT114

// Section : 452

// Instructor : Maura Deek

// Assignment # : Programming Assignment 3

// Date : 03/25/2016

// Description : This program is a Hangman game

//===================================================

import java.util.Random;

import java.lang.\*;

import java.util.Arrays;

public class Game{

int numWrongGuess,

index;

String words[] = {"AUTUMN", "UNIVERSITY","SUMMER","OXYGEN","JANUARY","PIZZA","FOOTBALL","MOVIE","JIGSAW","PENNSYLVANIA","BOOKWORM","FISHING","JAVA","PROGRAMMING"};

String[] lettersFound = {};

char[] currentWordLetters;

String currentWord ;

Boolean gameOver = false;

Random rand = new Random();

public Game(){

numWrongGuess = 0;

index = rand.nextInt(words.length);

currentWord = words[index];

setCurrentWordLetters();

setLettersFound();

}

public void setCurrentWordLetters()

{

currentWordLetters = new char[currentWord.length()];

for(int i = 0; i < currentWord.length(); i++){

currentWordLetters[i] = currentWord.charAt(i);

}

}

public void setLettersFound()

{

lettersFound = new String[currentWord.length()];

}

public String getCurrentWord()

{

return currentWord;

}

public String getAsterisk()

{

String asterisk = "";

for(int i = 0; i < currentWordLetters.length;i++){

lettersFound[i] = "\*";

asterisk += "\*";

}

return asterisk;

}

public String displayChar(String ch,char ch1)

{

char selectedChar = ch1;

String selectedStr = ch;

String returnWord = "";

if(currentWord.contains(selectedStr)){

for(int x = 0; x < currentWord.length(); x++){

if(currentWordLetters[x] == selectedChar){

lettersFound[x] = selectedStr;

}

}

}

else{

numWrongGuess += 1;

}

for(int z = 0; z < lettersFound.length; z++){

returnWord += lettersFound[z];

}

if(gameOver == true){

returnWord = currentWord;

}

return returnWord;

}

public String displayHangman()

{

String displayMan = "";

if(numWrongGuess == 1){

displayMan = "\_\_\_\_\_\_\_\n| |\n| O\n| \n| \n| \n| \n|\_\_\_\_\_\_\_";

}

else if (numWrongGuess == 2){

displayMan = "\_\_\_\_\_\_\_\n| |\n| O\n| | \n| \n| \n| \n|\_\_\_\_\_\_\_";

}

else if (numWrongGuess == 3){

displayMan = "\_\_\_\_\_\_\_\n| |\n| O\n| /| \n| \n| \n| \n|\_\_\_\_\_\_\_";

}

else if (numWrongGuess == 4){

displayMan = "\_\_\_\_\_\_\_\n| |\n| O\n| /|\\ \n| \n| \n| \n|\_\_\_\_\_\_\_";

}

else if (numWrongGuess == 5){

displayMan = "\_\_\_\_\_\_\_\n| |\n| O\n| /|\\ \n| / \n| \n| \n|\_\_\_\_\_\_\_";

gameOver = true;

}

else if (numWrongGuess == 6){

displayMan = "\_\_\_\_\_\_\_\n| |\n| O\n| /|\\ \n| / \\\n| \n| \n|\_\_\_\_\_\_\_\n Game Over! You Lose. Click New Game to play again.";

}

else

{

displayMan = "\_\_\_\_\_\_\_\n| |\n| \n| \n| \n| \n| \n|\_\_\_\_\_\_\_";

}

if ((Arrays.asList(lettersFound).contains("\*")) == false){

displayMan = "\_\_\_\_\_\_\_\n| |\n| \n| \n| \n| \n| \n|\_\_\_\_\_\_\_\nYou Win! Click New Game to play again.";

}

return displayMan;

}

public void newGame()

{

numWrongGuess = 0;

index = rand.nextInt(words.length);

currentWord = words[index];

setCurrentWordLetters();

setLettersFound();

gameOver = false;

}

}

//===================================================

// Name : Nidhi Patel

// SID : 31379144

// Course : IT114

// Section : 452

// Instructor : Maura Deek

// Assignment # : Programming Assignment 3

// Date : 03/25/2016

// Description : This program is a Hangman game

//===================================================

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.stage.Stage;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.geometry.Insets;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import java.util.Random;

import javafx.scene.layout.ColumnConstraints;

import javafx.scene.control.TextArea;

import javafx.event.EventHandler;

import javafx.event.ActionEvent;

import javafx.scene.text.Font;

import java.lang.\*;

public class Hangman extends Application {

public static void main(String[]args){

launch(args);

}

public void start(Stage primaryStage) throws Exception{

primaryStage.setTitle("Hangman");

primaryStage.setResizable(true);

GridPane layout = new GridPane();

layout.setPadding(new Insets(10));

layout.setHgap(5);

layout.setVgap(5);

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

layout.getColumnConstraints().add(new ColumnConstraints(50));

TextArea txtArea = new TextArea();

txtArea.setPrefColumnCount(50);

txtArea.setWrapText(true);

txtArea.setEditable(false);

txtArea.setFont(Font.font("Courier", 14));

layout.add(txtArea,0,0,6,5);

TextArea txtArea2 = new TextArea();

txtArea2.setPrefColumnCount(50);

txtArea2.setEditable(false);

txtArea2.setPadding(new Insets(70,10,70,10));

txtArea2.setFont(Font.font("Courier", 15));

layout.add(txtArea2,6,0,4,5);

Button btnA = new Button("A");

Button btnB = new Button("B");

Button btnC = new Button("C");

Button btnD = new Button("D");

Button btnE = new Button("E");

Button btnF = new Button("F");

Button btnG = new Button("G");

Button btnH = new Button("H");

Button btnI = new Button("I");

Button btnJ = new Button("J");

Button btnK = new Button("K");

Button btnL = new Button("L");

Button btnM = new Button("M");

Button btnN = new Button("N");

Button btnO = new Button("O");

Button btnP = new Button("P");

Button btnQ = new Button("Q");

Button btnR = new Button("R");

Button btnS = new Button("S");

Button btnT = new Button("T");

Button btnU = new Button("U");

Button btnV = new Button("V");

Button btnW = new Button("W");

Button btnX = new Button("X");

Button btnY = new Button("Y");

Button btnZ = new Button("Z");

Button btnNewGame = new Button("NewGame");

btnA.setMaxWidth(Double.MAX\_VALUE);

btnA.setDisable(true);

btnB.setMaxWidth(Double.MAX\_VALUE);

btnB.setDisable(true);

btnC.setMaxWidth(Double.MAX\_VALUE);

btnC.setDisable(true);

btnD.setMaxWidth(Double.MAX\_VALUE);

btnD.setDisable(true);

btnE.setMaxWidth(Double.MAX\_VALUE);

btnE.setDisable(true);

btnF.setMaxWidth(Double.MAX\_VALUE);

btnF.setDisable(true);

btnG.setMaxWidth(Double.MAX\_VALUE);

btnG.setDisable(true);

btnH.setMaxWidth(Double.MAX\_VALUE);

btnH.setDisable(true);

btnI.setMaxWidth(Double.MAX\_VALUE);

btnI.setDisable(true);

btnJ.setMaxWidth(Double.MAX\_VALUE);

btnJ.setDisable(true);

btnK.setMaxWidth(Double.MAX\_VALUE);

btnK.setDisable(true);

btnL.setMaxWidth(Double.MAX\_VALUE);

btnL.setDisable(true);

btnM.setMaxWidth(Double.MAX\_VALUE);

btnM.setDisable(true);

btnN.setMaxWidth(Double.MAX\_VALUE);

btnN.setDisable(true);

btnO.setMaxWidth(Double.MAX\_VALUE);

btnO.setDisable(true);

btnP.setMaxWidth(Double.MAX\_VALUE);

btnP.setDisable(true);

btnQ.setMaxWidth(Double.MAX\_VALUE);

btnQ.setDisable(true);

btnR.setMaxWidth(Double.MAX\_VALUE);

btnR.setDisable(true);

btnS.setMaxWidth(Double.MAX\_VALUE);

btnS.setDisable(true);

btnT.setMaxWidth(Double.MAX\_VALUE);

btnT.setDisable(true);

btnU.setMaxWidth(Double.MAX\_VALUE);

btnU.setDisable(true);

btnV.setMaxWidth(Double.MAX\_VALUE);

btnV.setDisable(true);

btnW.setMaxWidth(Double.MAX\_VALUE);

btnW.setDisable(true);

btnX.setMaxWidth(Double.MAX\_VALUE);

btnX.setDisable(true);

btnY.setMaxWidth(Double.MAX\_VALUE);

btnY.setDisable(true);

btnZ.setMaxWidth(Double.MAX\_VALUE);

btnZ.setDisable(true);

layout.add(btnA,0,7);

layout.add(btnB,4,8);

layout.add(btnC,2,8);

layout.add(btnD,2,7);

layout.add(btnE,2,6);

layout.add(btnF,3,7);

layout.add(btnG,4,7);

layout.add(btnH,5,7);

layout.add(btnI,7,6);

layout.add(btnJ,6,7);

layout.add(btnK,7,7);

layout.add(btnL,8,7);

layout.add(btnM,6,8);

layout.add(btnN,5,8);

layout.add(btnO,8,6);

layout.add(btnP,9,6);

layout.add(btnQ,0,6);

layout.add(btnR,3,6);

layout.add(btnS,1,7);

layout.add(btnT,4,6);

layout.add(btnU,6,6);

layout.add(btnV,3,8);

layout.add(btnW,1,6);

layout.add(btnX,1,8);

layout.add(btnY,5,6);

layout.add(btnZ,0,8);

layout.add(btnNewGame,7,8,2,1);

Game newGame = new Game();

btnNewGame.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

btnNewGame.setDisable(false);

btnA.setDisable(false);

btnB.setDisable(false);

btnC.setDisable(false);

btnD.setDisable(false);

btnE.setDisable(false);

btnF.setDisable(false);

btnG.setDisable(false);

btnH.setDisable(false);

btnI.setDisable(false);

btnJ.setDisable(false);

btnK.setDisable(false);

btnL.setDisable(false);

btnM.setDisable(false);

btnN.setDisable(false);

btnO.setDisable(false);

btnP.setDisable(false);

btnQ.setDisable(false);

btnR.setDisable(false);

btnS.setDisable(false);

btnT.setDisable(false);

btnU.setDisable(false);

btnV.setDisable(false);

btnW.setDisable(false);

btnX.setDisable(false);

btnY.setDisable(false);

btnZ.setDisable(false);

newGame.newGame();

txtArea2.setText(newGame.getAsterisk());

txtArea.setText(newGame.displayHangman());

}

});

btnA.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnA.setDisable(true);

txtArea2.setText(newGame.displayChar("A",'A'));

txtArea.setText(newGame.displayHangman());

}

});

btnB.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnB.setDisable(true);

txtArea2.setText(newGame.displayChar("B",'B'));

txtArea.setText(newGame.displayHangman());

}

});

btnC.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnC.setDisable(true);

txtArea2.setText(newGame.displayChar("C",'C'));

txtArea.setText(newGame.displayHangman());

}

});

btnD.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnD.setDisable(true);

txtArea2.setText(newGame.displayChar("D",'D'));

txtArea.setText(newGame.displayHangman());

}

});

btnE.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnE.setDisable(true);

txtArea2.setText(newGame.displayChar("E",'E'));

txtArea.setText(newGame.displayHangman());

}

});

btnF.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnF.setDisable(true);

txtArea2.setText(newGame.displayChar("F",'F'));

txtArea.setText(newGame.displayHangman());

}

});

btnG.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnG.setDisable(true);

txtArea2.setText(newGame.displayChar("G",'G'));

txtArea.setText(newGame.displayHangman());

}

});

btnH.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnH.setDisable(true);

txtArea2.setText(newGame.displayChar("H",'H'));

txtArea.setText(newGame.displayHangman());

}

});

btnI.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnI.setDisable(true);

txtArea2.setText(newGame.displayChar("I",'I'));

txtArea.setText(newGame.displayHangman());

}

});

btnJ.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnJ.setDisable(true);

txtArea2.setText(newGame.displayChar("J",'J'));

txtArea.setText(newGame.displayHangman());

}

});

btnK.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnK.setDisable(true);

txtArea2.setText(newGame.displayChar("K",'K'));

txtArea.setText(newGame.displayHangman());

}

});

btnL.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnL.setDisable(true);

txtArea2.setText(newGame.displayChar("L",'L'));

txtArea.setText(newGame.displayHangman());

}

});

btnM.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnM.setDisable(true);

txtArea2.setText(newGame.displayChar("M",'M'));

txtArea.setText(newGame.displayHangman());

}

});

btnN.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnN.setDisable(true);

txtArea2.setText(newGame.displayChar("N",'N'));

txtArea.setText(newGame.displayHangman());

}

});

btnO.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnO.setDisable(true);

txtArea2.setText(newGame.displayChar("O",'O'));

txtArea.setText(newGame.displayHangman());

}

});

btnP.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnP.setDisable(true);

txtArea2.setText(newGame.displayChar("P",'P'));

txtArea.setText(newGame.displayHangman());

}

});

btnQ.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnQ.setDisable(true);

txtArea2.setText(newGame.displayChar("Q",'Q'));

txtArea.setText(newGame.displayHangman());

}

});

btnR.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnR.setDisable(true);

txtArea2.setText(newGame.displayChar("R",'R'));

txtArea.setText(newGame.displayHangman());

}

});

btnS.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnS.setDisable(true);

txtArea2.setText(newGame.displayChar("S",'S'));

txtArea.setText(newGame.displayHangman());

}

});

btnT.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnT.setDisable(true);

txtArea2.setText(newGame.displayChar("T",'T'));

txtArea.setText(newGame.displayHangman());

}

});

btnU.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnU.setDisable(true);

txtArea2.setText(newGame.displayChar("U",'U'));

txtArea.setText(newGame.displayHangman());

}

});

btnV.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnV.setDisable(true);

txtArea2.setText(newGame.displayChar("V",'V'));

txtArea.setText(newGame.displayHangman());

}

});

btnW.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnW.setDisable(true);

txtArea2.setText(newGame.displayChar("W",'W'));

txtArea.setText(newGame.displayHangman());

}

});

btnX.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnX.setDisable(true);

txtArea2.setText(newGame.displayChar("X",'X'));

txtArea.setText(newGame.displayHangman());

}

});

btnY.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnY.setDisable(true);

txtArea2.setText(newGame.displayChar("Y",'Y'));

txtArea.setText(newGame.displayHangman());

}

});

btnZ.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

txtArea2.setText("");

btnZ.setDisable(true);

txtArea2.setText(newGame.displayChar("Z",'Z'));

txtArea.setText(newGame.displayHangman());

}

});

Scene scene = new Scene(layout);

primaryStage.setScene(scene);

primaryStage.show();

}

}

**4.2 Program and Module Description**

**setCurrentWordLetters**

This method sets the size of the currentWordLetters array according to the word the is randomly picked. It also initializes each index of the array with the corresponding letter of the word.

**setLettersFound**

This method sets the size of the lettersFound array to the size of the word the is picked.

**getCurrentWord**

Returns the active word of the game.

**getAsterisk**

Returns a variable that stores the same number of asterisks as the number of letters in the active word.

**displayChar**

Takes the user input and compares it with each index of the currentWordLetters array to determine if the given letter is in the array. If the letter is in the array the appropriate index of the lettersFound array is updated and the content of the lettersFound array is returned.

**displayHangman**

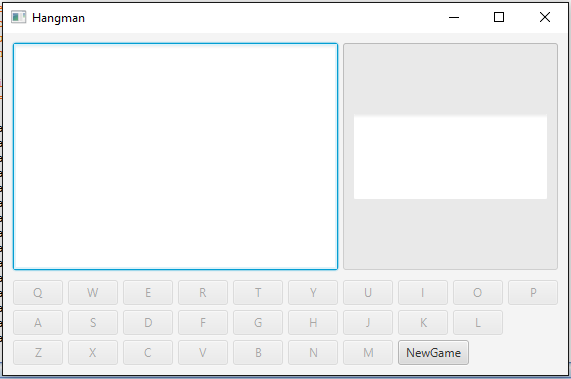
This methods checked the number of wrong guesses made by the user and depending on that number, the program adds a body part to the hangman.

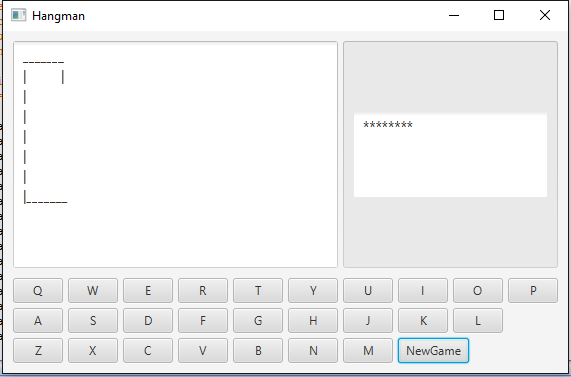
**newGame**

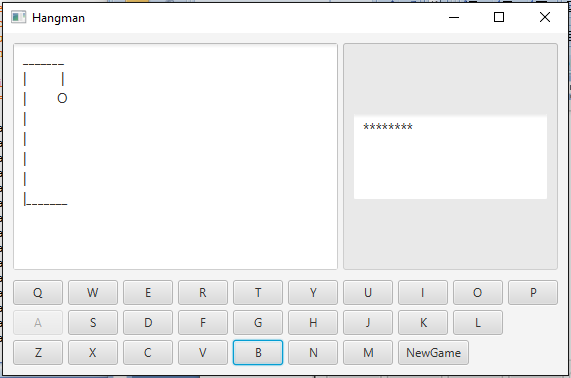
This method is used to reinitialize all the global variables to their original state to start a new game.

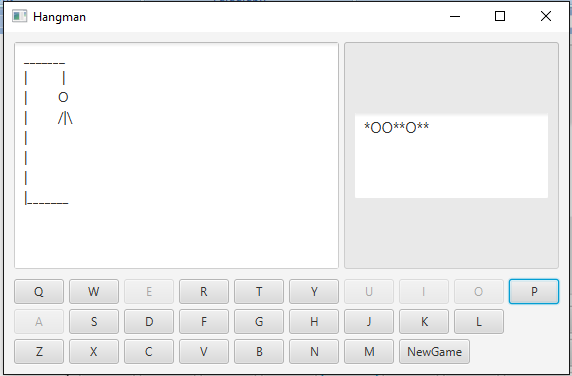
**5. I/O ScreenShots**

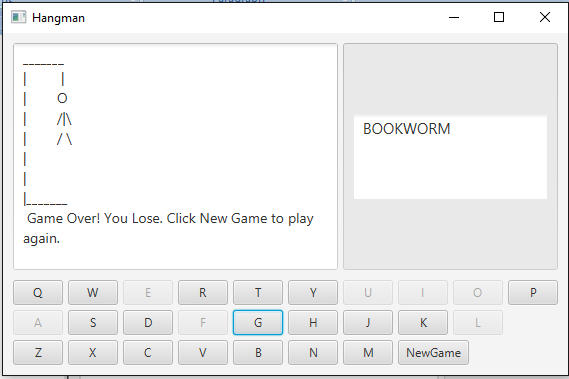
**New Game**

****

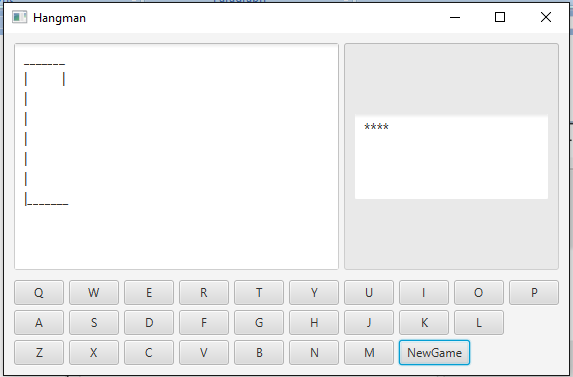
****

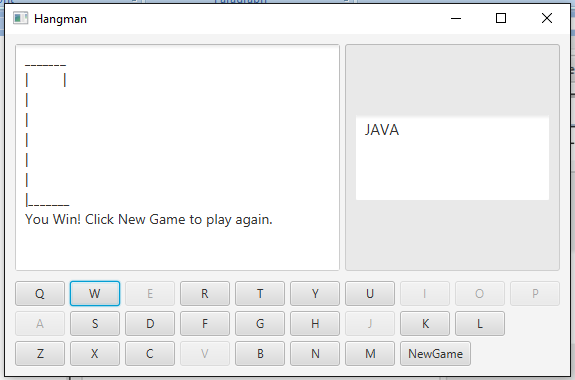
****

****

****

**New Game**

****

****