LBYCPEI

**Object-Oriented Programming Laboratory**



*Image by @Criisalys: https://twitter.com/Criisalys/status/1261069576126844929/photo/1*

**Laboratory Module 4**

Java Built-in Objects and Collections

By

Jay Michael T. Carlos, LBYCPEI-EQ8

# INTRODUCTION

In this module, you will create a hangman game using Java’s built-in objects and collections. Additionally, you will use two separate files for the canvas and main game logic.

1. Objectives
2. To understand built-in Java objects and collections.
3. To implement an ArrayList in a program.
4. To handle separation of the ACM canvas and console.

**What are the materials used for this module?**

1. Materials
2. IntelliJ IDEA (The Java IDE)
3. Java SE 14
4. cpei.jar (acm.jar and karel.jar)

# PROCEDURES (Individual) / EXPERIMENTAL PLAN

1. In Lab Activity 1: "Hangman” First, download the starter file from here: <https://github.com/LBYCPEI/HangmanStarter> and open the file in IntelliJ. At the top of the file, import java.io.\*, java.util.ArrayList, java.util.Objects, java.util.Scanner, and java.util.Random. Inside the intro() method, paste the following lines:

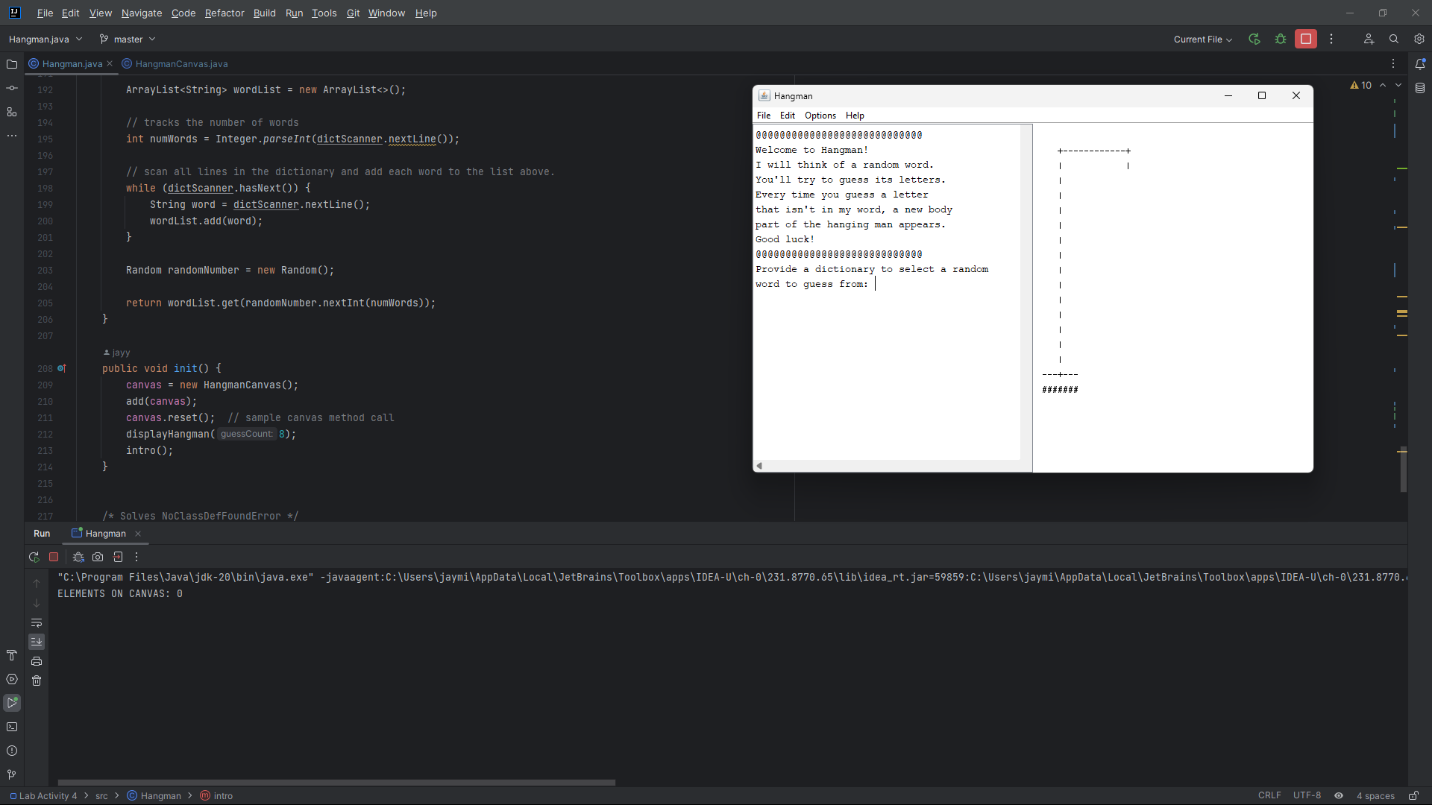
println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@");  
println("Welcome to Hangman!");  
println("I will think of a random word.");  
println("You'll try to guess its letters.");  
println("Every time you guess a letter");  
println("that isn't in my word, a new body");  
println("part of the hanging man appears.");  
println("Good luck!");  
println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@");

These lines just print messages on the left side of the screen, where the user will provide input. The main method to be used to start a game is playOneGame(String secretWord). It should return an int. Inside, create an integer guessCount with a value of 8 (this will serve as your lives), and two Strings guessedLetters and hintString. guessedLetters is a string containing the letters that the user has guessed, and hintString contains a string containing blanks and the letters that the user has guessed that match letters in the string. Then, create an infinite while loop by typing while (true). Inside this loop, you will type your main game code. Set hintString to the return value of the createHint() method with secretWord and guessedLetters as parameters. Then, print three lines: “Secret word: “ + hintString, “Guesses left: “ + guessCount, “Your guessed letters: “ + guessedLetters. Then, call the displayHangman() method with guessCount as the parameter. Then, create an if-else block that checks for the game over conditions. To do this, check if guessCount == 0, print the game over message, and break the loop. Also check if hintString does not contain a “\_ “. Remember that hintString contains blanks if a letter in the word has not been guessed yet. In this case, print a “you win” message and break the loop. Outside the if-block, create a character latestUserGuess that acts as a temporary variable that stores the return value of readGuess() with guessedLetters as the parameter. Then, check if secretWord does not contain latestUserGuess + “”. This checks if the secret word does not contain the letter that the user guessed and should decrease guessCount by 1 if so. After that if condition, add latestUserGuess to guessedLetters. Outside the while loop, create a conditional statement that checks if guessCount > 0. If so, increment gamesWon by 1, and inside, check if guessCount > bestScore. If so, set bestScore to the value of guessCount and set bestScoreWord to secretWord. After the if statement, increment gamesCount by 1, create a string userChoice that stores the uppercase return value of getLine() with a prompt asking the user if they want to play another game. Then, create an if statement that checks if userChoice is equal to “Y” by using the Objects.equals() method. If so, return 1; otherwise, return 0. Then, in the method createHint() that returns a string, and accepts two string parameters secretWord and guessedLetters. Inside, create an integer lengthOfSecretWord that stores secretWord.length() and a blank string newGuessedLetters. Then, create a for loop that uses the loop variable i = 0, continues while i < lengthOfSecretWord, and increments i after each iteration. Inside the loop, add a conditional statement that checks if guessedLetters contains the ith character in secretWord + “”. If so, add the ith character in secretWord to the end of the variable newGuessedLetters. Otherwise, add a “\_ “ to the end of newGuessedLetters. Lastly, return newGuessedLetters. Then, in the readGuess(String guessedLetters) method, create an infinite while loop with a string getCharacter that accepts input from the user with a prompt telling them to guess a letter. Then, set getCharacter to uppercase with Java’s toUpperCase method. Then, create a conditional statement that checks if getCharacter’s length is not 1, prompting the user to input one letter if so. Then, it should check if guessedLetters contains getCharacter, informing the user that they have already guessed the letter. If neither of these conditions are met, return the 0th character in getCharacter. Then in the displayHangman(int guessCount) method, create a new File object called file with the pathname “assets/display” + guessCount + “.txt”. Then, create a null Scanner object scanner and call the clear() function from HangmanCanvas like so: canvas.clear(). Open the file in a try-catch block by setting scanner to be a new Scanner object, with file as a parameter. Then, create a while loop that continues while scanner.hasNextLine() is true. Inside the loop, set scanner.nextLine() to the string data, and call the printText() method in HangmanCanvas, with data as a parameter. Outside of the loop, close the scanner with scanner.close(). Catch the FileNotFoundException e and call e.printStackTrace() to print a detailed error message in the console in case an error occurs reading the file. Then, in the getRandomWord(String filename) method, create a null Scanner object dictScanner and open the file as a File object dictionary with the parameter filename. Then, catch the generic Exception exception and call exception.printStackTrace(). Outside the try-catch block, create a new ArrayList of Strings wordList. Then, create an integer numWords that uses Integer.parseInt() with dictScanner.nextLine() as a parameter to keep track of the number of words. This just reads the first line of the file that contains a number representing the number of words in the file. Then, create a while loop that continues while dictScanner.hasNext() is true. Inside the loop, create a string word that stores dictScanner.nextLine() and add word to wordList with wordList.add(). The last task in this method is to create a Random object randomNumber and return wordList.get(randomNumber.nextInt(numWords)), which is a random word from wordList. Then, in your init() method, create a new instance of the HangmanCanvas() class, like so: canvas = new HangmanCanvas(), add it to the screen, call the canvas.reset() method, then the displayHangman() method with 8 as a parameter, and the intro() method. Then, do not forget to start your program in the main() method, like so: new Hangman().start(args).

In the next part, open your HangmanCanvas.java class and in the printText(String text) method, use .setFont() to set the font to be monospaced (examples include Courier New and JetBrains Mono). Then, below the comment that says “write your methods here”, create a clear() method that returns nothing, and inside, set the value of textX to TEXT\_X\_OFFSET and the value of textY to TEXT\_HEIGHT. Then, create a for loop with the loop variable i with a value of 0 that runs while i < getElementCount(), and i should increment by 1 each iteration. Inside the loop, create a new GObject element that gets the ith element on the canvas and removes it.

# RESULTS AND DISCUSSION

1. Lab Activity 1: "Hangman” Result:



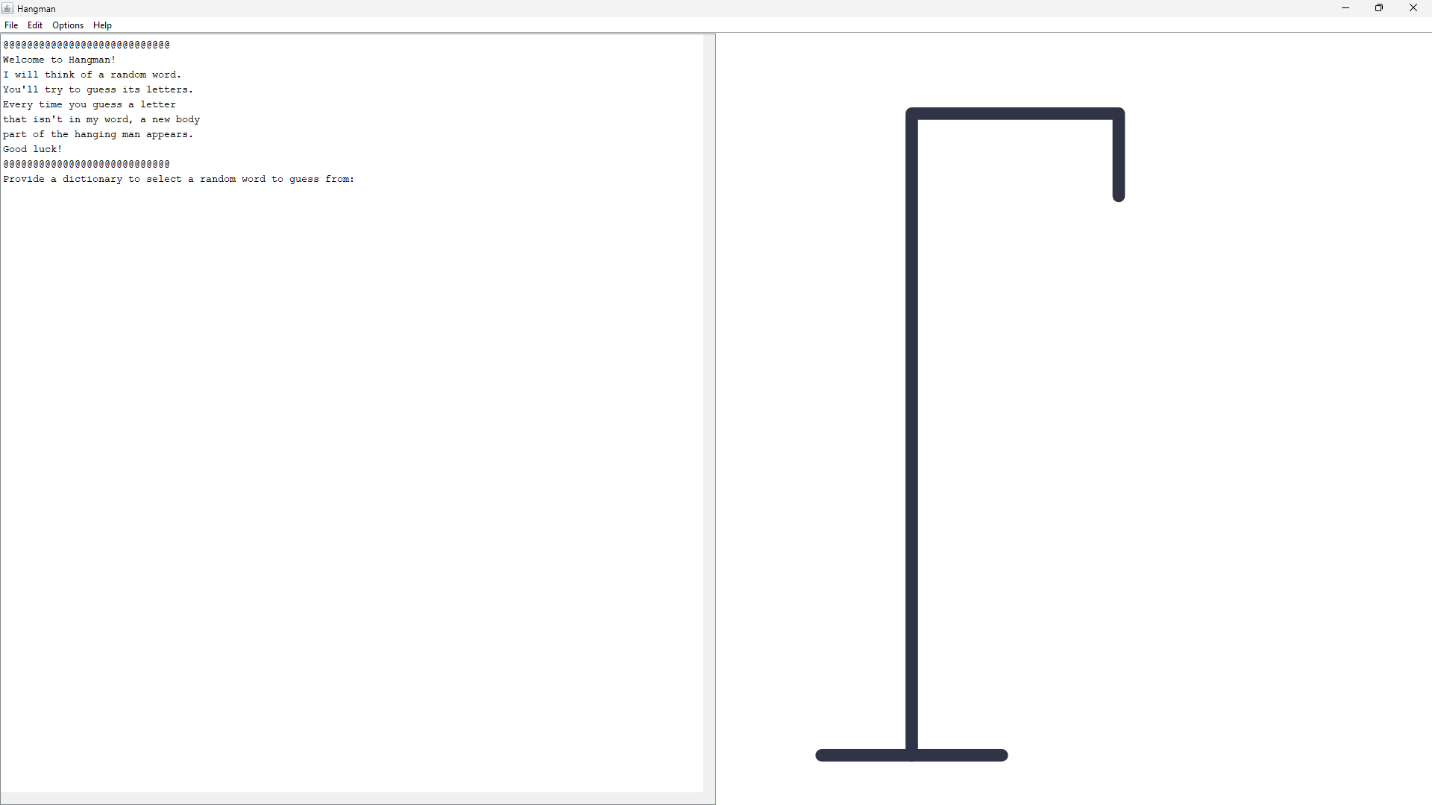
**Explanation**:

There are 9 methods you need to work in to create the Hangman game. run() starts the game and opens the dictionary file. After the main game program, the player’s statistics for the lifetime of the program are displayed. intro() just prints a message at the start of the program. playOneGame() returns an integer and accepts a string parameter secretWord. It runs the main gameplay loop of calling the other methods and displaying the messages that show the letters that the user has guessed and the remaining guesses. After each turn, it checks for game over conditions and breaks the loop if so. Outside the main loop, it determines whether the user has won the game or not and increments the appropriate statistics counters. The last thing the method does is that it returns either 1 or 0 depending on what the user answers in the prompt that asks them if they want to play another game or not. createHint() returns a string and accepts two string parameters secretWord and guessedLetters. It uses a for loop to check if each character in secretWord matches one of the letters in guessedLetters and add either a character or a blank depending on the result of the conditional to a new string. It then returns the newly generated string. readGuess() returns a character and accepts a string parameter guessedLetters. It contains an infinite loop that asks the user for input and checks whether that input is not one letter long and that guessedLetters contains an instance of the user’s input. If either of these conditions are met, the user gets an error message and the code loops; otherwise, the method returns the character the user inputs. displayHangman() returns nothing and accepts an integer parameter guessCount. It reads the display.txt files in the assets folder, clears the canvas, and prints the text from the file. stats() returns nothing and accepts three integer parameters gamesCount, gamesWon, and best. It displays the number of games played, games won, and the user’s best score (highest number of guesses remaining). getRandomWord() returns a string value and accepts a string parameter filename. The method returns a random word from a provided dictionary .txt file. init() initializes the game by creating a new instance of HangmanCanvas, where the graphics will be displayed, displays the actual initial hangman graphic, and calls the intro() method.

**A screenshot of a computer

Description automatically generated with medium confidence**

2. Lab Activity 2: "Hangman GImage” Result:



**Explanation**:

The second version is practically identical, except the text file containing the hangman graphic is replaced by GImages.

**A screenshot of a computer

Description automatically generated with medium confidence**

# CONCLUSION:

**Did you achieve your objectives for this module? Describe your achievement in this module one-by-one, objective by objective.**

**What have you learned? Both technically (focused on OOP), and in general (soft skills, attitude-related, etc.)**

**What are the common pitfalls, mistakes, and confusion that you have encountered? How did you overcome them?**

**What are your recommendations for those who will try the activity for the first time and what can you suggest to improve this module?**

I personally achieved the objectives I set for this module. Built-in Java objects and collections were used throughout the program, such as in storing the list of words from the dictionary. I have also learned to use multiple files to separate the main game logic and canvas. This is crucial to creating more complicated apps, as elements such as GUI are better kept separate from the main game logic. The most common mistake I made was mixing up the String type and character types for returns and input parameters. This would commonly result in an error, but luckily taking note of the return types helps make it easier to avoid these errors. I recommend that you experiment with string and character types, and how to convert between the two to avoid any confusion. Additionally, I would suggest that you read the documentation for ArrayLists so you have a better understanding of what they are and how they work.

# REFERENCES

1. Gosling, J., Joy, B., Steele, G., Bracha, G., & Buckley, A. (2015). The Java language specification. Oracle America, Inc.
2. Roberts, E. (2005). Karel the robot learns java. Department of Computer Science Stanford University.
3. Roberts, E. (2008). The Art & Science of Java. Pearson.
4. Oracle. Java Documentation. ( https://docs.oracle.com/en/java )
5. Schildt, H. (2017). Java: A Beginner's Guide, 7th Edition. McGraw-Hill Education.
6. Troccoli, N. (2017). CS 106A: Assignment #1: Karel the Robot. Stanford University
7. UML basics: The component diagram. http://www.ibm.com/developerworks/rational/library/dec04/bell/

# APPENDIX

1. Lab Activity 1.1: Hangman

import acm.program.\*;  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Objects;  
import java.util.Scanner;  
import java.util.Random;  
  
public class Hangman extends ConsoleProgram {  
 // tracks game stats during the lifetime of the program.  
 int bestScore = 0, gamesCount = 0, gamesWon = 0;  
 String bestScoreWord = "";  
  
 public void run() {  
 int gameContinue = 1;  
  
 // loop this to allow for multiple games.  
 while (gameContinue == 1) {  
 String fileName = promptUserForFile("Provide a dictionary to select a random word to guess from: ", "assets/");  
 gameContinue = playOneGame(getRandomWord(fileName));  
  
 stats(gamesCount, gamesWon, bestScore);  
 }  
 }  
  
 // this method introduces the player to hangman.  
 private void intro() {  
 println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@");  
 println("Welcome to Hangman!");  
 println("I will think of a random word.");  
 println("You'll try to guess its letters.");  
 println("Every time you guess a letter");  
 println("that isn't in my word, a new body");  
 println("part of the hanging man appears.");  
 println("Good luck!");  
 println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@");  
 }  
  
 // TODO: comment this method  
 private int playOneGame(String secretWord) {  
 // set the length of the secret word and  
 // the boolean condition that dictates whether the game is running or not.  
 int guessCount = 8;  
 String guessedLetters = "";  
 String hintString = "";  
  
 // run the methods necessary for displaying the game.  
 while (true) {  
 // display the blanks and guessed letters.  
 hintString = createHint(secretWord, guessedLetters);  
 println("Secret word: " + hintString);  
 println("Guesses left: " + guessCount);  
 println("Your guessed letters: " + guessedLetters);  
  
 // display the hangman graphic in the canvas.  
 displayHangman(guessCount);  
  
 // check for game over. this is placed here so that the function can exit before the user inputs a word.  
 if (guessCount == 0) {  
 println("Game over.");  
 println("The word was: " + secretWord);  
 break;  
 } else if (!hintString.contains("\_ ")) {  
 println("You won.");  
 break;  
 }  
  
 // add the user's guessed letter to a temp variable.  
 char latestUserGuess = readGuess(guessedLetters);  
  
 // if the user's guess is not in the word to be guessed, subtract 1 from lives (guessCount).  
 if (!secretWord.contains(latestUserGuess + "")) {  
 guessCount--;  
 }  
  
 // add the latest user guess to the guessedLetters string.  
 guessedLetters += latestUserGuess;  
 }  
  
 // add a win to the stats if the player won.  
 if (guessCount > 0) {  
 gamesWon++;  
  
 // if your guess count is higher, set it as your best score and keep track of the word.  
 if (guessCount > bestScore) {  
 bestScore = guessCount;  
 bestScoreWord = secretWord;  
 }  
 }  
  
 gamesCount++;  
  
 // ask the user if they want to continue or not.  
 String userChoice = "";  
 userChoice = getLine("Would you like to play another game? (Y/N): ").toUpperCase();  
  
 if (Objects.equals(userChoice, "Y")) {  
 return 1;  
 } else {  
 return 0;  
 }  
 }  
  
  
 // creates the hint that displays the blanks and the user's guessed letters.  
 private String createHint(String secretWord, String guessedLetters) {  
 // length of secret word.  
 int lengthOfSecretWord = secretWord.length();  
  
 // new string of guessed letters to be returned.  
 String newGuessedLetters = new String();  
  
 // display the blanks and guessed letters that match.  
 for (int i = 0; i < lengthOfSecretWord; i++) {  
 // checks if the "guessedLetters" list contains a letter equal to the ith letter in "secretWord".  
 // the quotation marks convert the character to a string.  
 // if a match is found, concatenate the ith letter in "secretWord" to newGuessedLetters.  
 // otherwise, concatenate a blank to newGuessedLetters.  
 if (guessedLetters.contains(secretWord.charAt(i) + "")) {  
 newGuessedLetters += secretWord.charAt(i);  
 } else {  
 newGuessedLetters += "\_ ";  
 }  
 }  
 return newGuessedLetters;  
 }  
  
 // reads input from the user, verifies the validity of the input, and returns the character.  
 private char readGuess(String guessedLetters) {  
 /\* Loop to make sure input is valid as explained above\*/  
 while(true) {  
 boolean notDuplicate = true;  
 /\* Input \*/  
 String getCharacter = getLine("Guess a letter: ");  
 /\* Change to Uppercase \*/  
 getCharacter = getCharacter.toUpperCase();  
 /\* Condition for [1] a single letter (A-Z) \*/  
 if (getCharacter.length() != 1) {  
 println("Input one letter.");  
 }  
 /\* Checker for [2] been guessed already \*/  
 else if (guessedLetters.contains(getCharacter)) {  
 println("You have already guessed this letter.");  
 }  
 /\* if neither of those error conditions are met, add the character to guessedLetters and return that \*/  
 else {  
 return getCharacter.charAt(0);  
 }  
 }  
 }  
  
 // displays the hangman graphic on the canvas.  
 private void displayHangman(int guessCount) {  
 File file = new File("assets/display" + guessCount + ".txt");  
 Scanner scanner = null;  
 canvas.clear();  
  
 try {  
 scanner = new Scanner(file);  
 while (scanner.hasNextLine()) {  
 String data = scanner.nextLine();  
 canvas.printText(data);  
 }  
 scanner.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // displays the number of games played in a session, games won, and the best score.  
 private void stats(int gamesCount, int gamesWon, int best) {  
 println("\n\nGames played this session: " + gamesCount);  
 println("Games won this session: " + gamesWon);  
  
 if (gamesWon > 0) {  
 println("Your best score this session was " + best + " guesses remaining for the word '" + bestScoreWord + "'.\n\n");  
 } else {  
 println("You do not have a best score this session, as you have not won a game yet.");  
 }  
 }  
  
 // gets a random word from a provided dictionary .txt file.  
 private String getRandomWord(String filename) {  
 Scanner dictScanner = null;  
  
 try {  
 File dictionary = new File(filename);  
 dictScanner = new Scanner(dictionary);  
 } catch (Exception exception) {  
 exception.printStackTrace();  
 }  
  
 ArrayList<String> wordList = new ArrayList<>();  
  
 // tracks the number of words  
 int numWords = Integer.parseInt(dictScanner.nextLine());  
  
 // scan all lines in the dictionary and add each word to the list above.  
 while (dictScanner.hasNext()) {  
 String word = dictScanner.nextLine();  
 wordList.add(word);  
 }  
  
 Random randomNumber = new Random();  
  
 return wordList.get(randomNumber.nextInt(numWords));  
 }  
  
 public void init() {  
 canvas = new HangmanCanvas();  
 add(canvas);  
 canvas.reset(); // sample canvas method call  
 displayHangman(8);  
 intro();  
 }  
  
  
 /\* Solves NoClassDefFoundError \*/  
 public static void main(String[] args) {  
 new Hangman().start(args);  
 }  
  
  
 // private HangmanCanvas canvas;  
 private HangmanCanvas canvas;  
}

2. Lab Activity 1.2: Hangman Canvas

/\*  
 \* File: HangmanCanvas.java  
 \* ---------------------  
 \* This class holds the graphics elements to the Hangman game.  
 \* Author: Cobalt - M.Cabatuan  
 \* Date modified: 06/11/2019  
 \*/  
  
  
import acm.graphics.GCanvas;  
import acm.graphics.GLabel;  
import acm.graphics.GObject;  
  
public class HangmanCanvas extends GCanvas {  
  
 private static final int *TEXT\_HEIGHT* = 20; // you can modify this to suit your ascii art  
 private static final int *TEXT\_X\_OFFSET* = 12; // you can modify this to suit your ascii art  
 private int textX;  
 private int textY;  
  
  
 */\*\*  
 \* Resets the display so that only the hangman scaffold appears  
 \*/* public void reset() {  
 // Sample graphics object  
 textX = TEXT\_X\_OFFSET;  
 textY = TEXT\_HEIGHT;  
 }  
  
 public void printText(String text){  
 GLabel line = new GLabel(text);  
 line.setFont("Courier New");  
 textY += TEXT\_HEIGHT;  
 add(line, textX , textY );  
 }  
  
 /\* Write your methods here \*/  
  
 // clears all elements on the canvas.  
 public void clear() {  
 System.out.println("ELEMENTS ON CANVAS: " + getElementCount());  
 textX = TEXT\_X\_OFFSET;  
 textY = TEXT\_HEIGHT;  
  
 // iterate through each element and remove it.  
 for (int i = 0; i < getElementCount(); i++) {  
 System.out.println("element: " + getElement(i));  
 GObject element = getElement(i);  
 remove(element);  
 }  
 }  
}

3. Lab Activity 2.1: Hangman GImage

import acm.graphics.GImage;  
import acm.program.\*;  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Objects;  
import java.util.Scanner;  
import java.util.Random;  
  
public class Hangman extends ConsoleProgram {  
 // tracks game stats during the lifetime of the program.  
 int bestScore = 0, gamesCount = 0, gamesWon = 0;  
 String bestScoreWord = "";  
  
 public void run() {  
 int gameContinue = 1;  
  
 // loop this to allow for multiple games.  
 while (gameContinue == 1) {  
 String fileName = promptUserForFile("Provide a dictionary to select a random word to guess from: ", "assets/");  
 gameContinue = playOneGame(getRandomWord(fileName));  
  
 stats(gamesCount, gamesWon, bestScore);  
 }  
 }  
  
 // this method introduces the player to hangman.  
 private void intro() {  
 println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@");  
 println("Welcome to Hangman!");  
 println("I will think of a random word.");  
 println("You'll try to guess its letters.");  
 println("Every time you guess a letter");  
 println("that isn't in my word, a new body");  
 println("part of the hanging man appears.");  
 println("Good luck!");  
 println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@");  
 }  
  
 // *TODO: comment this method* private int playOneGame(String secretWord) {  
 // set the length of the secret word and  
 // the boolean condition that dictates whether the game is running or not.  
 int guessCount = 8;  
 String guessedLetters = "";  
 String hintString = "";  
  
 // run the methods necessary for displaying the game.  
 while (true) {  
 // display the blanks and guessed letters.  
 hintString = createHint(secretWord, guessedLetters);  
 println("Secret word: " + hintString);  
 println("Guesses left: " + guessCount);  
 println("Your guessed letters: " + guessedLetters);  
  
 // display the hangman graphic in the canvas.  
 displayHangman(guessCount);  
  
 // check for game over. this is placed here so that the function can exit before the user inputs a word.  
 if (guessCount == 0) {  
 println("Game over.");  
 println("The word was: " + secretWord);  
 break;  
 } else if (!hintString.contains("\_ ")) {  
 println("You won.");  
 break;  
 }  
  
 // add the user's guessed letter to a temp variable.  
 char latestUserGuess = readGuess(guessedLetters);  
  
 // if the user's guess is not in the word to be guessed, subtract 1 from lives (guessCount).  
 if (!secretWord.contains(latestUserGuess + "")) {  
 guessCount--;  
 }  
  
 // add the latest user guess to the guessedLetters string.  
 guessedLetters += latestUserGuess;  
 }  
  
 // add a win to the stats if the player won.  
 if (guessCount > 0) {  
 gamesWon++;  
  
 // if your guess count is higher, set it as your best score and keep track of the word.  
 if (guessCount > bestScore) {  
 bestScore = guessCount;  
 bestScoreWord = secretWord;  
 }  
 }  
  
 gamesCount++;  
  
 // ask the user if they want to continue or not.  
 String userChoice = "";  
 userChoice = getLine("Would you like to play another game? (Y/N): ").toUpperCase();  
  
 if (Objects.equals(userChoice, "Y")) {  
 return 1;  
 } else {  
 return 0;  
 }  
 }  
  
  
 // creates the hint that displays the blanks and the user's guessed letters.  
 private String createHint(String secretWord, String guessedLetters) {  
 // length of secret word.  
 int lengthOfSecretWord = secretWord.length();  
  
 // new string of guessed letters to be returned.  
 String newGuessedLetters = new String();  
  
 // display the blanks and guessed letters that match.  
 for (int i = 0; i < lengthOfSecretWord; i++) {  
 // checks if the "guessedLetters" list contains a letter equal to the ith letter in "secretWord".  
 // the quotation marks convert the character to a string.  
 // if a match is found, concatenate the ith letter in "secretWord" to newGuessedLetters.  
 // otherwise, concatenate a blank to newGuessedLetters.  
 if (guessedLetters.contains(secretWord.charAt(i) + "")) {  
 newGuessedLetters += secretWord.charAt(i);  
 } else {  
 newGuessedLetters += "\_ ";  
 }  
 }  
 return newGuessedLetters;  
 }  
  
 // reads input from the user, verifies the validity of the input, and returns the character.  
 private char readGuess(String guessedLetters) {  
 /\* Loop to make sure input is valid as explained above\*/  
 while(true) {  
 boolean notDuplicate = true;  
 /\* Input \*/  
 String getCharacter = getLine("Guess a letter: ");  
 /\* Change to Uppercase \*/  
 getCharacter = getCharacter.toUpperCase();  
 /\* Condition for [1] a single letter (A-Z) \*/  
 if (getCharacter.length() != 1) {  
 println("Input one letter.");  
 }  
 /\* Checker for [2] been guessed already \*/  
 else if (guessedLetters.contains(getCharacter)) {  
 println("You have already guessed this letter.");  
 }  
 /\* if neither of those error conditions are met, add the character to guessedLetters and return that \*/  
 else {  
 return getCharacter.charAt(0);  
 }  
 }  
 }  
  
 // displays the hangman graphic on the canvas.  
 private void displayHangman(int guessCount) {  
 canvas.setImage("assets/display" + guessCount + ".png");  
 canvas.clear();  
 }  
  
 // displays the number of games played in a session, games won, and the best score.  
 private void stats(int gamesCount, int gamesWon, int best) {  
 println("\n\nGames played this session: " + gamesCount);  
 println("Games won this session: " + gamesWon);  
  
 if (gamesWon > 0) {  
 println("Your best score this session was " + best + " guesses remaining for the word '" + bestScoreWord + "'.\n\n");  
 } else {  
 println("You do not have a best score this session, as you have not won a game yet.");  
 }  
 }  
  
 // gets a random word from a provided dictionary .txt file.  
 private String getRandomWord(String filename) {  
 Scanner dictScanner = null;  
  
 try {  
 File dictionary = new File(filename);  
 dictScanner = new Scanner(dictionary);  
 } catch (Exception exception) {  
 exception.printStackTrace();  
 }  
  
 ArrayList<String> wordList = new ArrayList<>();  
  
 // tracks the number of words  
 int numWords = Integer.parseInt(dictScanner.nextLine());  
  
 // scan all lines in the dictionary and add each word to the list above.  
 while (dictScanner.hasNext()) {  
 String word = dictScanner.nextLine();  
 wordList.add(word);  
 }  
  
 Random randomNumber = new Random();  
  
 return wordList.get(randomNumber.nextInt(numWords));  
 }  
  
 public void init() {  
 canvas = new HangmanCanvas();  
 add(canvas);  
 canvas.reset(); // sample canvas method call  
 canvas.setImage("assets/display8.png");  
 intro();  
 }  
  
  
 /\* Solves NoClassDefFoundError \*/  
 public static void main(String[] args) {  
 new Hangman().start(args);  
 }  
  
  
 // private HangmanCanvas canvas;  
 private HangmanCanvas canvas;  
}

4. Lab Activity 2.2: Hangman Canvas GImage

/\*  
 \* File: HangmanCanvas.java  
 \* ---------------------  
 \* This class holds the graphics elements to the Hangman game.  
 \* Author: Cobalt - M.Cabatuan  
 \* Date modified: 06/11/2019  
 \*/  
  
  
import acm.graphics.GCanvas;  
import acm.graphics.GImage;  
import acm.graphics.GLabel;  
import acm.graphics.GObject;  
  
public class HangmanCanvas extends GCanvas {  
  
 private static final int *TEXT\_HEIGHT* = 20; // you can modify this to suit your ascii art  
 private static final int *TEXT\_X\_OFFSET* = 12; // you can modify this to suit your ascii art  
 private int textX;  
 private int textY;  
  
  
 */\*\*  
 \* Resets the display so that only the hangman scaffold appears  
 \*/* public void reset() {  
 // Sample graphics object  
 textX = *TEXT\_X\_OFFSET*;  
 textY = *TEXT\_HEIGHT*;  
 }  
  
 public void printText(String text){  
 GLabel line = new GLabel(text);  
 line.setFont("Courier New");  
 textY += *TEXT\_HEIGHT*;  
 add(line, textX , textY );  
 }  
  
 /\* Write your methods here \*/  
  
 // clears all elements on the canvas.  
 public void clear() {  
 System.*out*.println("ELEMENTS ON CANVAS: " + getElementCount());  
 textX = *TEXT\_X\_OFFSET*;  
 textY = *TEXT\_HEIGHT*;  
  
 // iterate through each element and remove it.  
 for (int i = 0; i < getElementCount(); i++) {  
 System.*out*.println("element: " + getElement(i));  
 GObject element = getElement(i);  
 remove(element);  
 }  
 }  
  
 // puts the image element on the canvas.  
 public void setImage(String filename) {  
 GImage hangman = new GImage(filename, 0, 0);  
 add(hangman);  
 }  
}