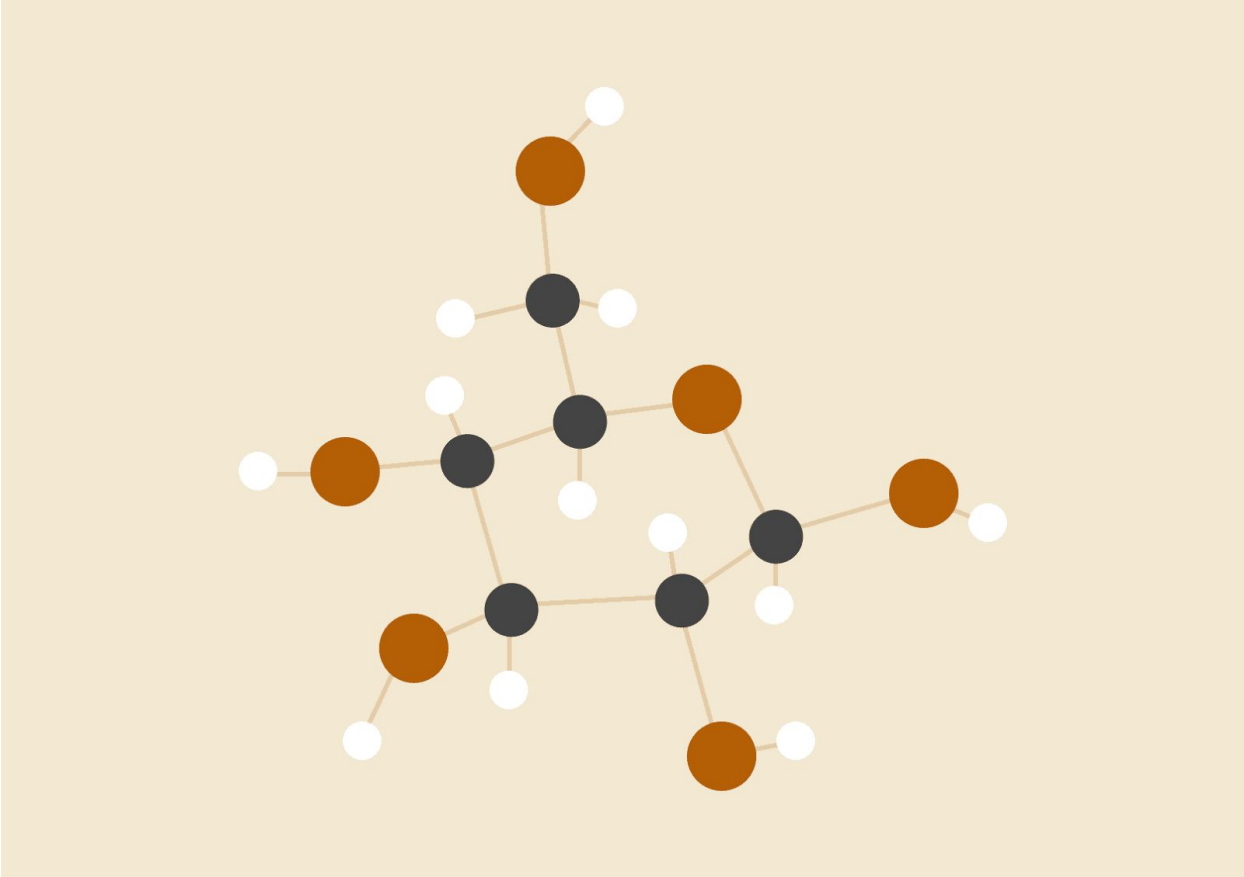
LBYCP12

Data Structures and Algorithm Analysis Laboratory



Laboratory Activity 3

Hangman

By

Your Name, LBYCP12-EQ1/2

INTRODUCTION

In this activity, the students will try to create an interactive game of Hangman, which combines a console program with a graphics canvas, and utilizes a lexicon that reads from a text file filled with thousands of words. In addition, this activity integrates an implementation of Linked List ADT in order to store the various words in the selection.

OBJECTIVES

- To learn how to work with multiple classes in a single application.
- To learn about the concept of stubs, which is creating a temporary simpler working version of a program implementation, that provides enough functionality to test the rest of the program.
- To create a program that can read data from a text file.
- To utilize a Linked List ADT Implementation in storing a huge database (of words).

MATERIALS

1. Java SE Development Kit (JDK) 8
2. NetBeans integrated development environment (IDE)
3. `acm.jar` by the ACM Java Task Force

PROCEDURE

1. Write a program that handles the user interaction component of the game—everything except the graphical display. Program must be able to do the following:
    a. Choose a random word to use as the secret word. That word is chosen from a word list.
    b. Keep track of the user's partially guessed word, which begins as a series of dashes and then gets updated as correct letters are guessed.
    c. Implement the basic control structure and manage the details (ask the user to guess a letter, keep track of the number of guesses remaining, print out the various messages, detect the end of the game, and so forth).

2. Use this stub to temporarily implement HangmanLexicon,java:

```
/*
* File: HangmanLexicon.java
* -------------------------
* This file contains a stub implementation of the HangmanLexicon
* class that you will reimplement for Part III of the assignment.
*/
import acm.util.*;
public class HangmanLexicon {
/** Returns the number of words in the lexicon. */
public int getWordCount() {
return 10;
}
/** Returns the word at the specified index. */
public String getWord(int index) {
switch (index) {
case 0: return "BUOY";
case 1: return "COMPUTER";
case 2: return "CONNOISSEUR";
case 3: return "DEHYDRATE";
case 4: return "FUZZY";
case 5: return "HUBBUB";
case 6: return "KEYHOLE";
case 7: return "QUAGMIRE";
case 8: return "SLITHER";
case 9: return "ZIRCON";
default: throw new ErrorException("getWord: Illegal index");
}
};
}
```

3. Create a new HangmanLexicon and store it in an instance variable. If you extend the program to allow the user to play multiple games, the creation of the HangmanLexicon should be performed outside the loop that plays the game repeatedly so that this operation is performed once rather than for every game.
4. The console program must satisfy these conditions:
    a. It should accept the user's guesses in either lower or upper case, even though all letters in the secret words are written in upper case.
    b. If the user guesses something other than a single letter, the program should

tell the user that the guess is illegal and accept a new guess.

c. If the user guesses a correct letter more than once, the program should simply do nothing.
d. Guessing an incorrect letter a second time should be counted as another wrong guess.

5. Extend the program already written so that it now keeps track of the Hangman graphical display.
6. In the program, the parts are added in the following order: head, body, left arm, right arm, left leg, right leg, left foot, right foot. Use these constants:

```
/*
* File: HangmanCanvas.java
* ------------------------
* This file keeps track of the Hangman display.
*/
import acm.graphics.*;
public class HangmanCanvas extends GCanvas {
/** Resets the display so that only the scaffold appears */
public void reset() {
/* You fill this in */
}
/**
* Updates the word on the screen to correspond to the current
* state of the game. The argument string shows what letters have
* been guessed so far; unguessed letters are indicated by hyphens.
*/
public void displayWord(String word) {
/* You fill this in */
}
/**
* Updates the display to correspond to an incorrect guess by the
* user. Calling this method causes the next body part to appear
* on the scaffold and adds the letter to the list of incorrect
* guesses that appears at the bottom of the window.
*/
public void noteIncorrectGuess(char letter) {
/* You fill this in */
}
/* Constants for the simple version of the picture (in pixels) */
private static final int SCAFFOLD_HEIGHT = 360;
private static final int BEAM_LENGTH = 144;
private static final int ROPE_LENGTH = 18;
private static final int HEAD_RADIUS = 36;
private static final int BODY_LENGTH = 144;
private static final int ARM_OFFSET_FROM_HEAD = 28;
```

```
private static final int UPPER_ARM_LENGTH = 72;
private static final int LOWER_ARM_LENGTH = 44;
private static final int HIP_WIDTH = 36;
private static final int LEG_LENGTH = 108;
private static final int FOOT_LENGTH = 28;
}
```

7. Declare an instance variable for the canvas by writing

   ```
   private HangmanCanvas canvas;
   ```

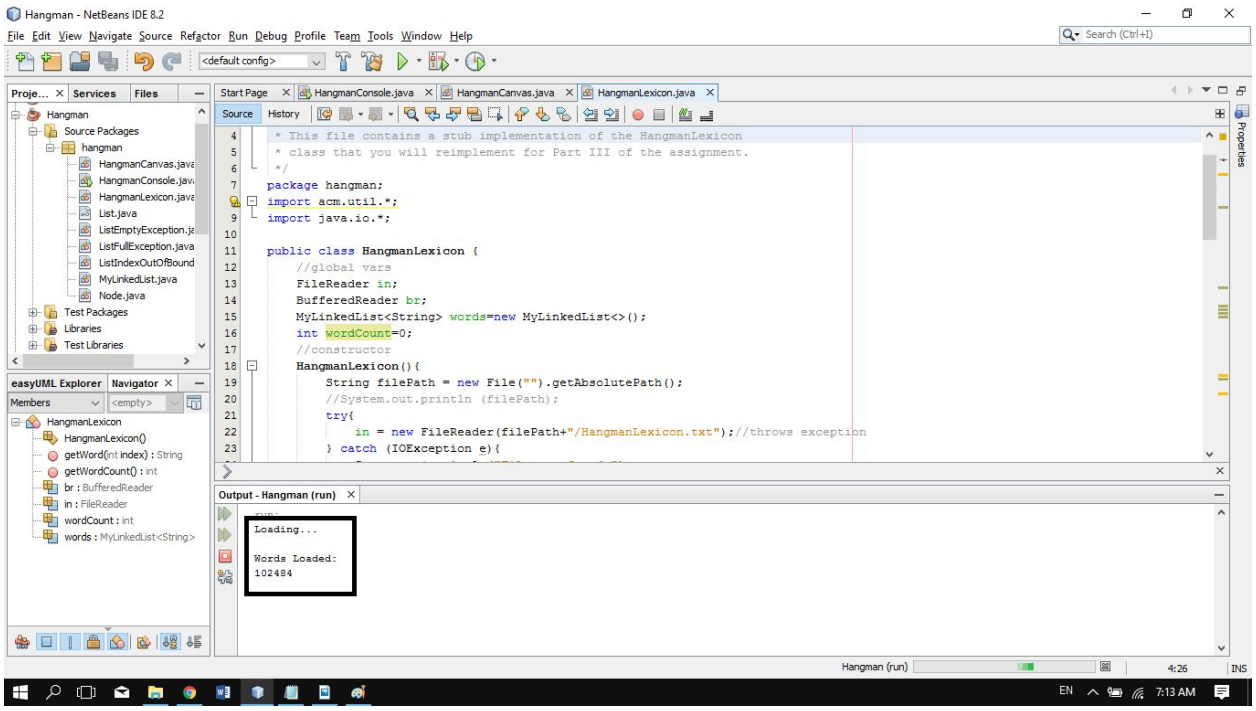
   and then add the following init method to the program:

   ```
   public void init() {
   canvas = new HangmanCanvas();
   add(canvas);
   }
   ```

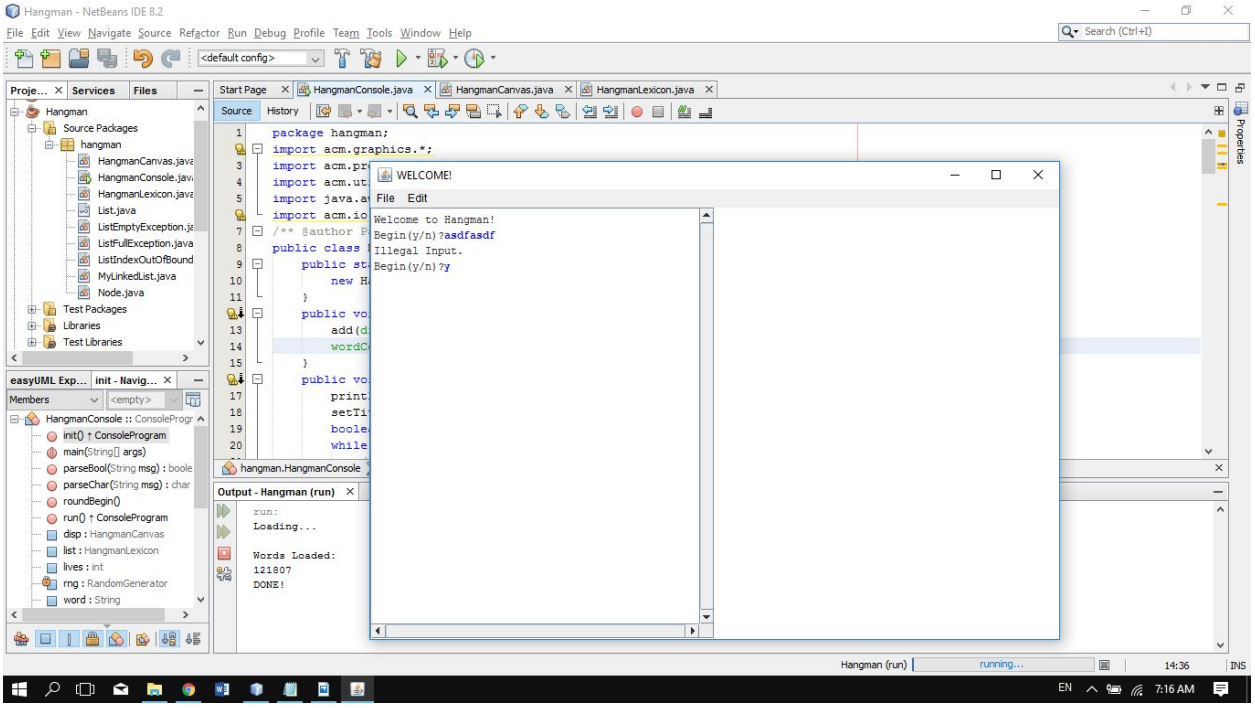
8. Go through and add the calls to the methods in HangmanCanvas.
9. The center line of the body should be centered horizontally on the screen, and the scaffold should be displayed a bit higher than the center so that there is room underneath for two labels: a label in a large font showing the secret word as it currently stands and a label in a smaller font showing the incorrect guesses.
10. Open the data file HangmanLexicon.txt using a BufferedReader that will allow you to read it line by line.
11. Read the lines from the file into an Linked List.
12. Reimplement the getWordCount and getWord methods in HangmanLexicon so that they use the Linked List from step 11 as the source of the words.

RESULTS AND DISCUSSION

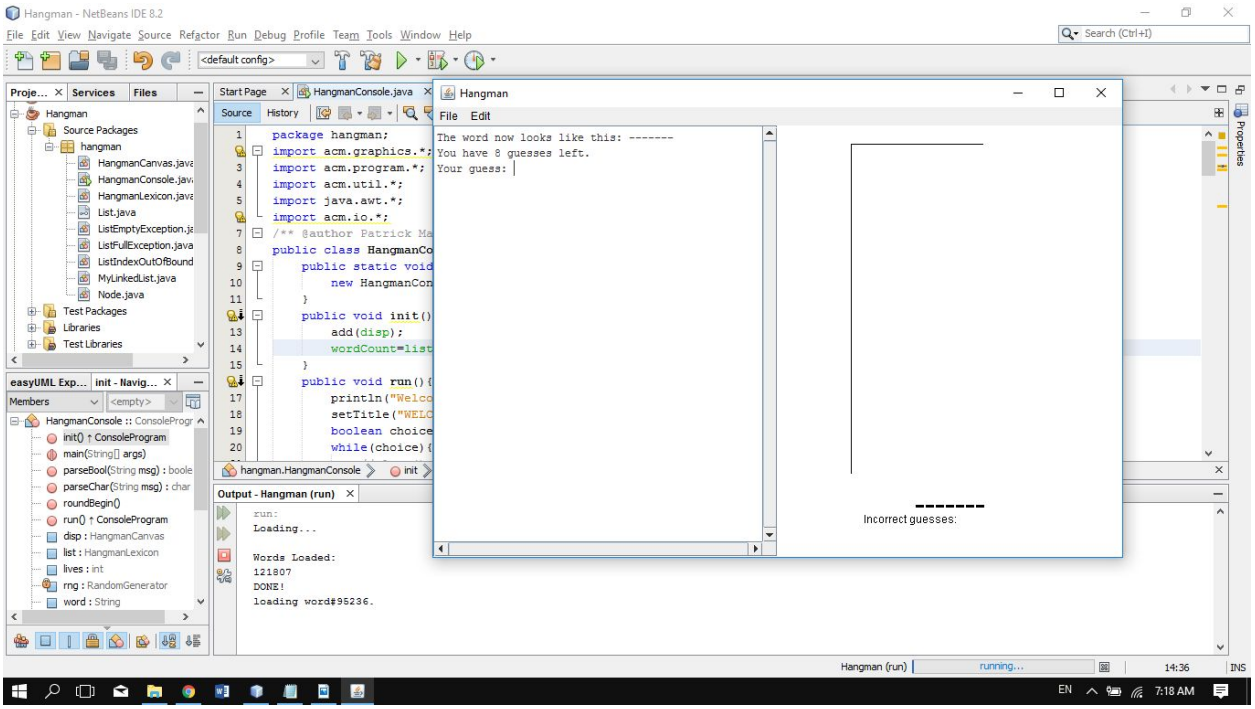
a) Screenshots



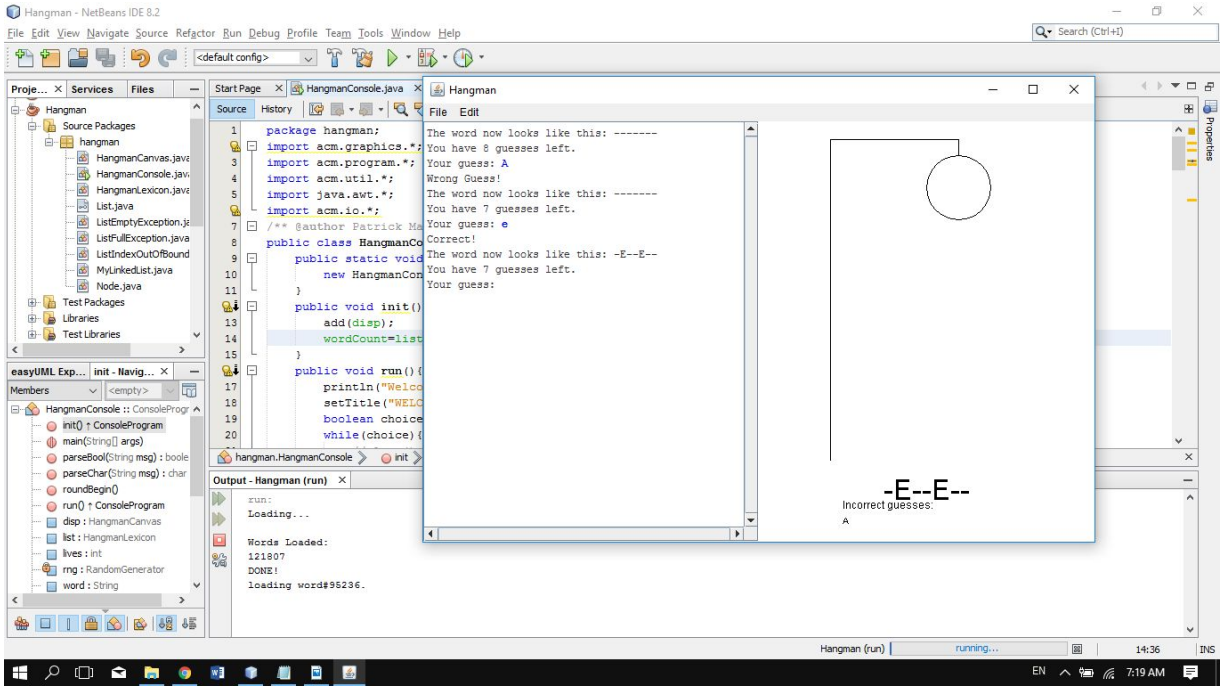
Program takes a while to begin game, as each line in the HangmanLexicon.txt is stored as a Linked List member before the game randomly chooses a word from these.



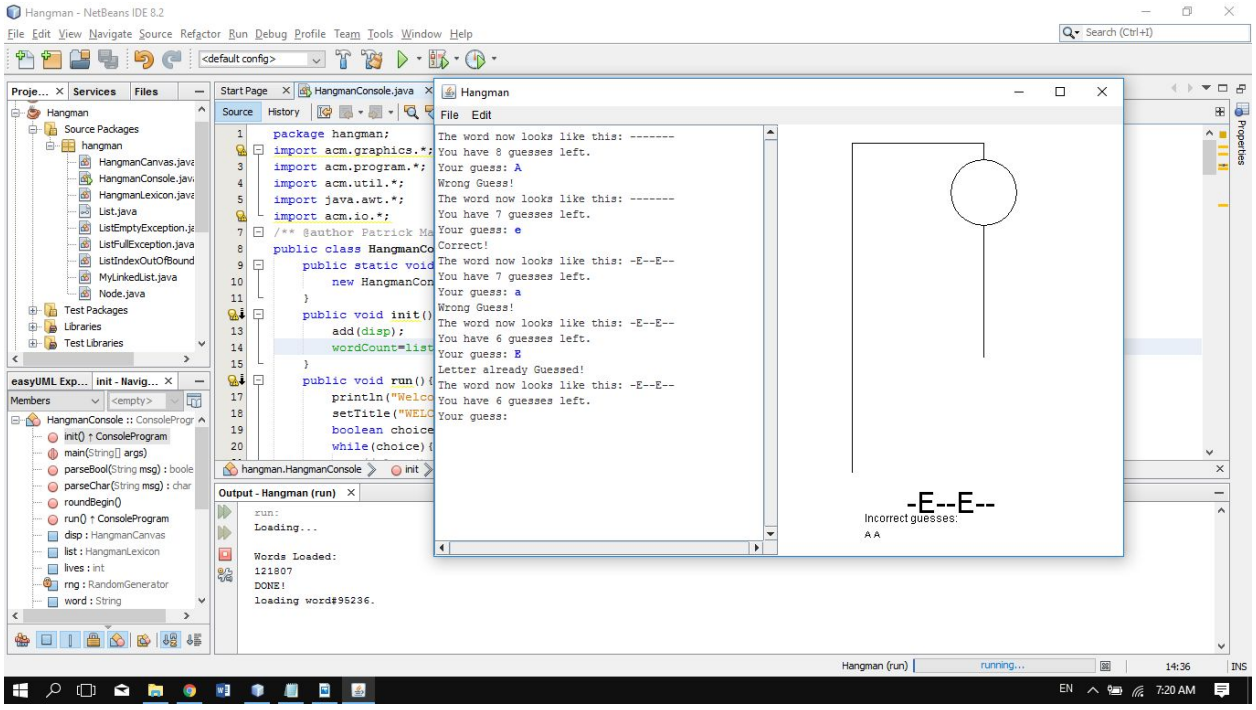
Program welcomes user to the game, and asks the user before starting the game. (user response is checked whether it is y/Y,n/N, or invalid)



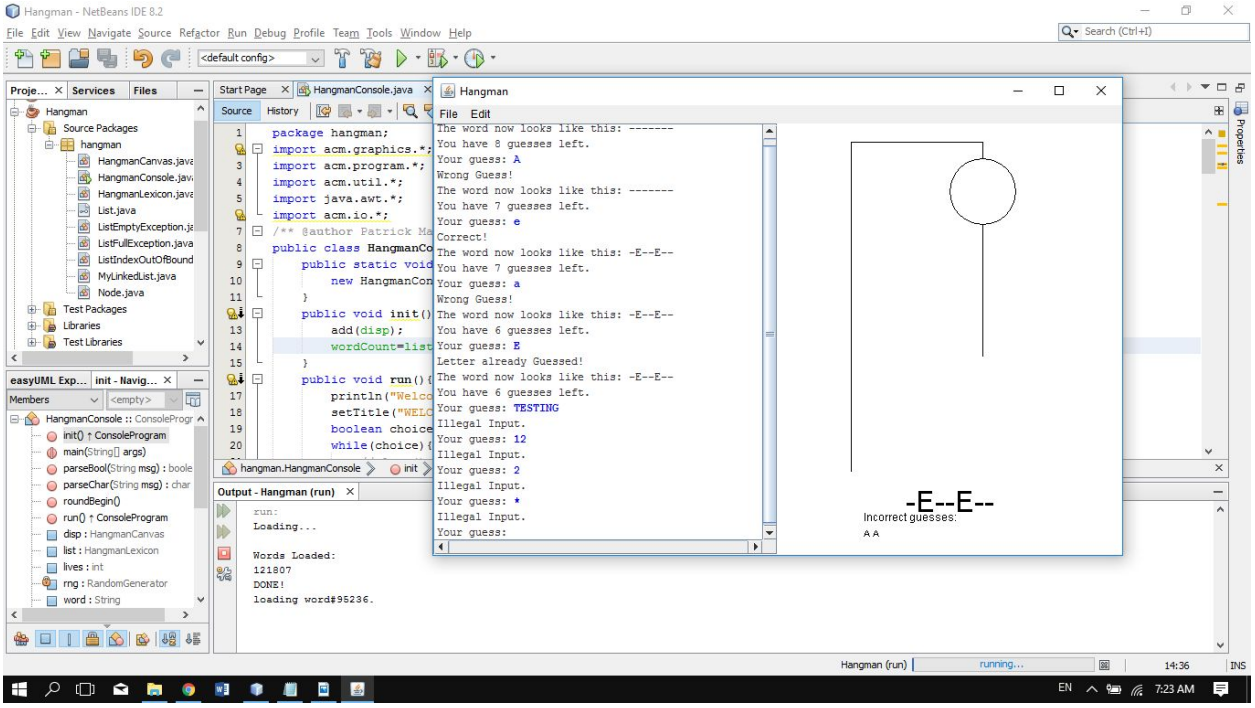
Program draws beam and scaffold as game begins, and chooses a random word to hide.



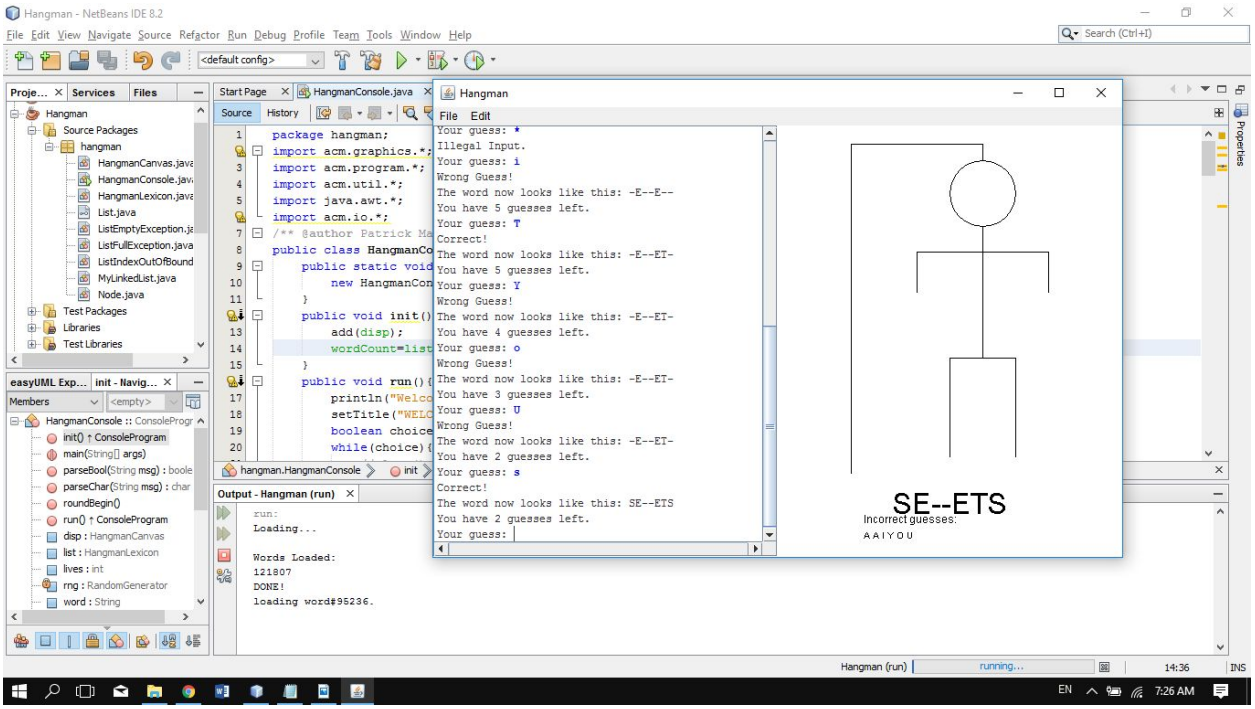
Incorrect guesses are noted, which also triggers game to draw the next part of the hangman, while correct guesses change the displayed hidden String.



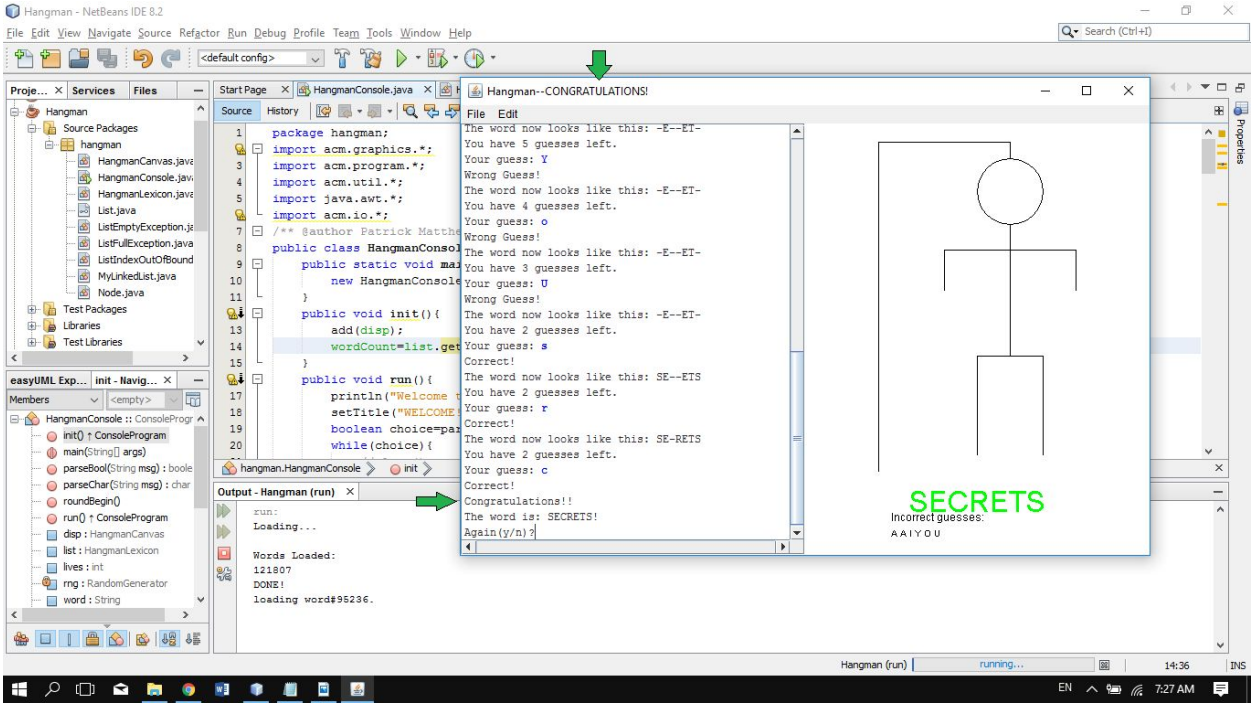
Repeat correct guesses are ignored, while repeat wrong guesses are counted as another error, as stated in the procedures.



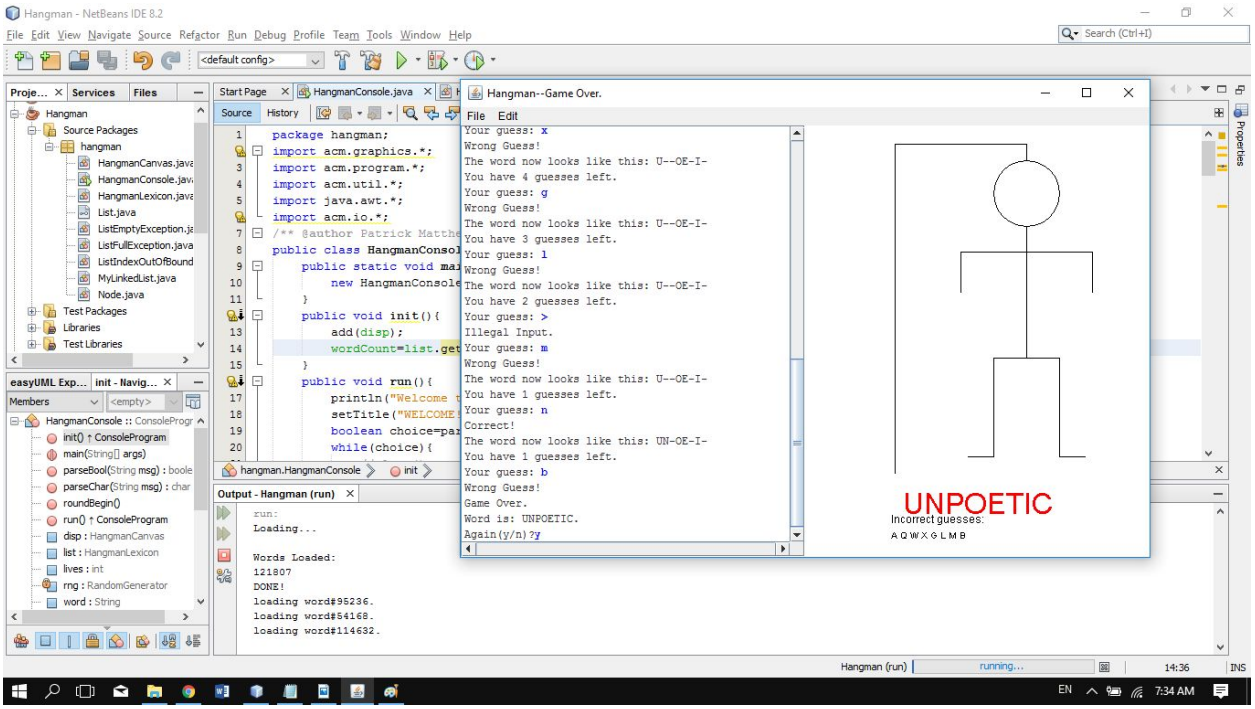
Program checks the input first, and ignores it when it is invalid, asking the user again for a valid input.



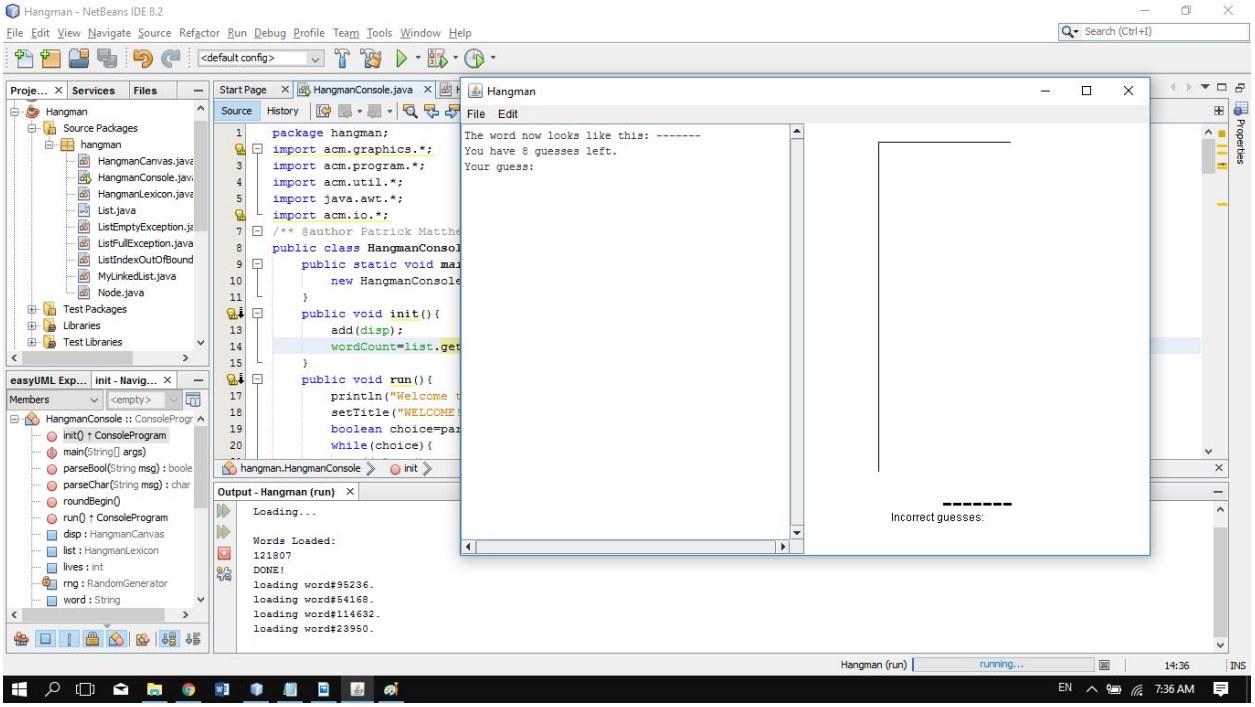
Program accepts both uppercase and lowercase inputs.



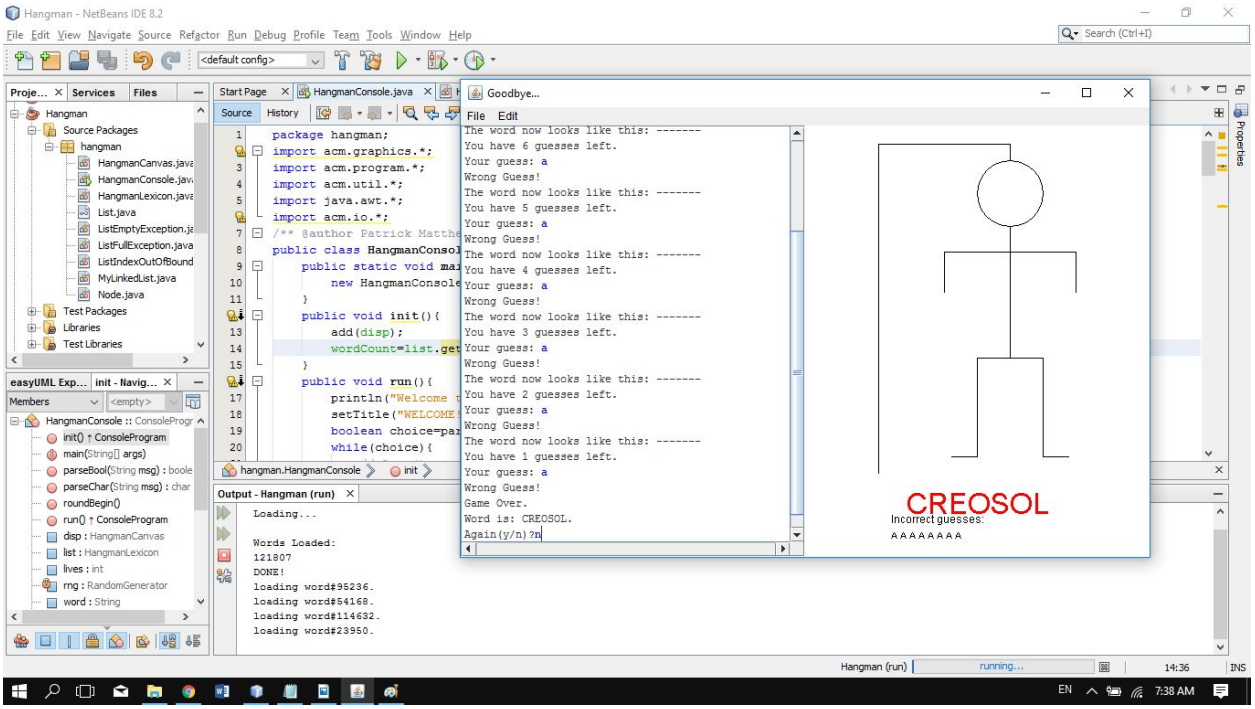
Program congratulates user when the word is correctly guessed within the given number of lives. The program also asks if the user wants to play another round.



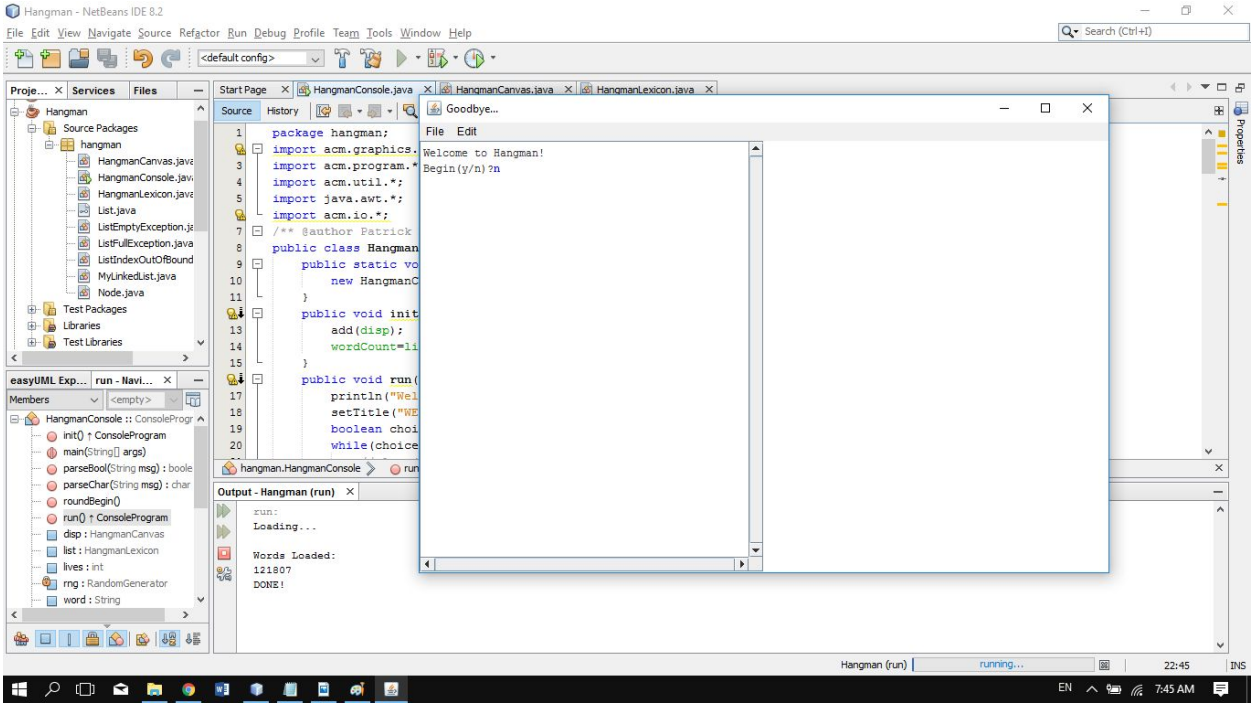
If the lives run out, the program reveals the word, and shows “Game Over.”



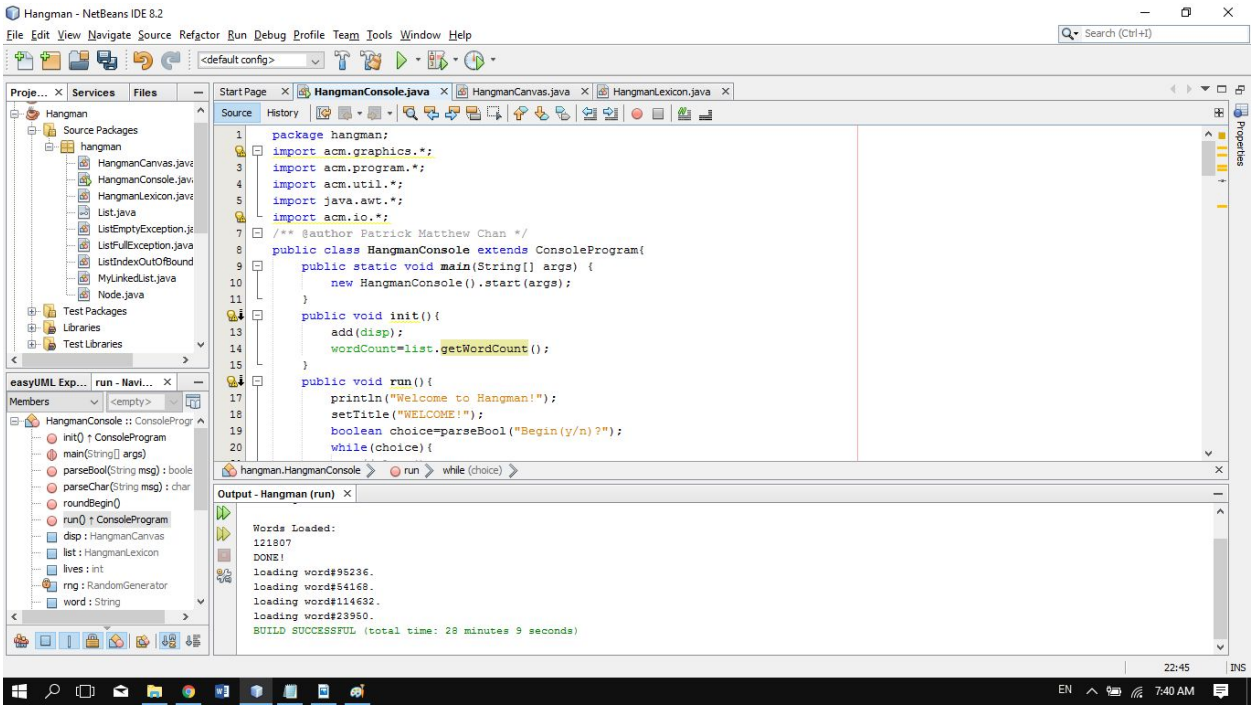
Program chooses a new word, clears console screen, and redraws the hangman on starting another round.



Program says “Goodbye” and then closes, upon choosing no.



The same goes for choosing no at the welcome screen.



Program terminates upon choosing “n” for no.

SUMMARY

One thing the I have learned about in this activity, is how to display a canvas and a functional console in the same program at the same time. This allows the program to become quite interactive, as the program can accept inputs and print outputs, while simultaneously modifying the graphics display based on the user input. This is achieved by creating my own class that extends GCanvas, and creating my own methods that manipulate the graphics part of the screen, and finally create an object of this class (HangmanCanvas), and add it to the program. So, calling the method of an object of the newly created class, can now make the console manipulate the graphics on the canvas.

Another thing I noticed, is that switching from Static List Implementation to Linked List was effortless, as both implementations implement the same class. In other words, both have the same set of methods, even though the processes done inside each method is different, so the only things I needed to change in the program, were the files I needed to copy into the source packages portion of the project.

Lastly, the introduction of stubs would prove to be very useful, as it would give way in creating the main framework of the program, before expanding the capabilities of the program itself. It is very nice because this allows creating a small scale version of the program, then only tweaking it a bit to expand the range of the program.

APPENDIX

the code:

HangmanConsole:

```
package ph.edu.dlsu.chan.hangman;
import acm.graphics.*;
import acm.program.*;
import acm.util.*;
import java.awt.*;
import acm.io.*;
```

```
/** @author Patrick Matthew Chan */
public class HangmanConsole extends ConsoleProgram{
   public static void main(String[] args) {
      new HangmanConsole().start(args);
   }
   public void init(){
      add(disp);
      println("Loading...");
      Thread loader=new Thread(new Runnable() {
         @Override
         public void run() {
            list=new HangmanLexicon();

            wordCount=list.getWordCount();
         }
      });
      loader.start();
      while(loader.isAlive()){
         pause(rng.nextInt(50, 500));
         getConsole().clear();
         for(int l=0;l<25;l++){println();}
         println("       Loading...");
         print("       \\ \\ \\ \\ \\ \\ \\ \\ \r");
         disp.setBackground(getCol(rng.nextInt(0,5)));
         pause(rng.nextInt(50, 300));
         getConsole().clear();
         for(int l=0;l<25;l++){println();}
         println("       Loading...");
         print("       | | | | | | | | \r");
         disp.setBackground(getCol(rng.nextInt(0,5)));
         pause(rng.nextInt(50, 200));
         disp.setBackground(Color.white);
         disp.add(new GLabel("Loading...", 100, 550));
         getConsole().clear();
         for(int l=0;l<25;l++){println();}
         println("       Loading...");
         print("       / / / / / / / \r");
         pause(rng.nextInt(50, 300));
         getConsole().clear();
         for(int l=0;l<25;l++){println();}
         println("       Loading...");
         print("       - - - - - - - \r");
         disp.removeAll();
         disp.setBackground(getCol(rng.nextInt(0,5)));
      }
      getConsole().clear();
      disp.removeAll();
```

```
        disp.setBackground(new Color(202,236,207));
    }
    public void run(){
        println("Welcome to Hangman!");
        setTitle("WELCOME!");
        boolean choice=parseBool("Begin(y/n)?");
        while(choice){
            //clear();
            //IOConsole a=this.getConsole();
            boolean choice2=parseBool("\nDo you want to guess a specific numbered
word(y/n)?");
            int numChoice2=-1;
            if(choice2){
                numChoice2=readInt("Please enter special word number: (input any integer
less than 1 to cancel)\n");
                System.out.println("Range: 1 - "+list.getWordCount());
                if(numChoice2>list.getWordCount() || numChoice2<0){
                    println("Invalid choice. Choosing a random word.");
                    numChoice2=-1;
                }
            }
            setTitle("Hangman");
            this.getConsole().clear();
            roundBegin(numChoice2);
            choice=parseBool("Again(y/n)?");
        }
        setTitle("Goodbye...");
        pause(500);
        exit();
    }
    public static Color getCol(int index){
        switch(index%6){
            case 1:
                return Color.GRAY;
            case 3:
                return Color.GREEN;
            case 4:
                return Color.MAGENTA;
            case 2:
                return Color.ORANGE;
            case 5:
                return Color.RED;
            default://case 0 here also
                return Color.BLUE;
        }
    }
```

```
//global variables
public HangmanCanvas disp= new HangmanCanvas();
public HangmanLexicon list=null;
public int wordCount;//lexicon^^ word count
private RandomGenerator rng = RandomGenerator.getInstance();
public int lives=0;
public String word="";

//functions i.e. methods
//hidden word display
public class Hide{
   public String decoded="";//normal
   public String coded="";//displayed
   public int length;

   //constructor
   Hide(String word){
      decoded=word;
      for(int i=1;i<=word.length();i++){
         coded=coded.concat("-");
      }
      length=coded.length();
   }
   //methods
   private void check(char a) throws InputCharException{
      if(a<'A' || a>'Z'){
         throw new InputCharException("Invalid Char: "+a);
      }
   }
   public boolean isDuplicate(char a){
      check(a);
      Character aa=a;
      return coded.contains(aa.toString());
   }
   public boolean isComplete(){
      return !(coded.contains("-"));
   }
   public boolean update(char a){//true if changed, ie correct guess
      check(a);
      boolean ans=false;
      String temp=coded;
      coded="";
      for(int i=1;i<=length;i++){
         if(decoded.charAt(i-1)==a){
            Character aa=a;
            coded=coded.concat(aa.toString());
            ans=true;
```

```
            } else if(temp.charAt(i-1)!='-'){
                Character temp2=temp.charAt(i-1);
                coded=coded.concat(temp2.toString());
            }
            else {
                coded=coded.concat("-");
            }
        }
        return ans;
    }
}
public class InputCharException extends RuntimeException{
    public InputCharException(String s){
        super(s);
    }//end constructor
}
public char parseChar(String msg){
    while(true){
        String a=readLine(msg);
        if(a.length()==1){
            char b=a.toUpperCase().charAt(0);
            if(b>='A' && b<='Z'){
                return b;
            }
        }
        println("Illegal Input.");
    }
}
//+++++ lang (y/n)
public boolean parseBool(String msg){
    while(true){
        String a=readLine(msg);
        if(a.length()==1){
            char b=a.toUpperCase().charAt(0);
            if(b=='Y'){
                return true;
            } else if(b=='N'){
                return false;
            }
        }
        println("Illegal Input.");
    }
}



public void roundBegin(int specialChoice){
```

```
    int num=specialChoice;
    if(num>wordCount || num<1){
       num=rng.nextInt(1,wordCount);
    }
    //System.out.println("loading word#"+num+".");
    println("Want to challenge your friends?\n"
          + "This is word #"+ num + "\n\n\n");
    word=list.getWord(num);
    lives=8;
    Hide h= new Hide(word);
    //println(h.decoded);//test
    disp.roundBegin(lives,h.coded);
    while(lives>0 && !(h.isComplete())){
       println("The word now looks like this: "+h.coded);
       println("You have "+lives+" guesses left.");
       char a=parseChar("Your guess: ");
       if(h.isDuplicate(a)){
          println("Letter already Guessed!");
       } else {
          if(h.update(a)){
             println("Correct!");
             disp.update(lives,h.coded);
          } else {
             println("Wrong Guess!");
             lives--;
             disp.update(lives,h.coded, a);
          }
       }
    }
    if(h.isComplete()){
      println("Congratulations!!") ;
      println("The word is: "+h.coded+"!");
      disp.update(lives,h.coded);
      disp.wordDisp.setColor(Color.GREEN);
      setTitle("Hangman--CONGRATULATIONS!");
    } else if(lives==0){
       println("Game Over.");
       println("Word is: "+h.decoded+".");
       disp.update(lives,h.decoded);
       disp.wordDisp.setColor(Color.RED);
       setTitle("Hangman--Game Over.");
    }

}
```

```
}
```

HangmanCanvas:

```
/*
 * File: HangmanCanvas.java
 * ------------------------
 * This file keeps track of the Hangman display.
 */
package ph.edu.dlsu.chan.hangman;
import acm.graphics.*;
import java.awt.Color;

public class HangmanCanvas extends GCanvas {
   /* Constants for the simple version of the picture (in pixels) */
   private static final int SCAFFOLD_HEIGHT = 360;
   private static final int BEAM_LENGTH = 144;
   private static final int ROPE_LENGTH = 18;
   private static final int HEAD_RADIUS = 36;
   private static final int BODY_LENGTH = 144;
   private static final int ARM_OFFSET_FROM_HEAD = 28;
   private static final int UPPER_ARM_LENGTH = 72;
   private static final int LOWER_ARM_LENGTH = 44;
   private static final int HIP_WIDTH = 36;
   private static final int LEG_LENGTH = 108;
   private static final int FOOT_LENGTH = 28;
   //offsets
   private static final int HANGMAN_Y_OFFSET = 20;
   private static final int WRONG_Y_OFFSET=20;
   private static final int WORD_OFFSET=10;
   //global vars
   GHangman man=new GHangman(0,0);
   GLabel wordDisp=new GLabel("");
   private static final String WORD_FONT="Century Schoolbook L-32";
   GLabel wrongDisp0=new GLabel("Incorrect guesses: ");
   GLabel wrongDisp=new GLabel("");
   private static final String WRONG_FONT="Century Schoolbook L-10";
   //GCompound Hangman
   public class LeftArm extends GCompound{
      //parts
      GLine leftArm1=new GLine(UPPER_ARM_LENGTH,0,0,0);
      GLine leftArm2=new GLine(0,0,0,LOWER_ARM_LENGTH);
      //constructor
```

```
    LeftArm(int x,int y){
        add(leftArm1,x,y);
        add(leftArm2,x-UPPER_ARM_LENGTH,y);
    }
}
public class RightArm extends GCompound{
    //parts
    GLine rightArm1=new GLine(0,0,UPPER_ARM_LENGTH,0);
    GLine rightArm2=new GLine(0,0,0,LOWER_ARM_LENGTH);
    //constructor
    RightArm(int x,int y){
        add(rightArm1,x,y);
        add(rightArm2,x+UPPER_ARM_LENGTH,y);
    }
}
public class LeftLeg extends GCompound{
    //parts
    GLine leftLeg1=new GLine(HIP_WIDTH,0,0,0);
    GLine leftLeg2=new GLine(0,0,0,LEG_LENGTH);
    //constructor
    LeftLeg(int x,int y){
        add(leftLeg1,x,y);
        add(leftLeg2,x-HIP_WIDTH,y);
    }
}
public class RightLeg extends GCompound{
    //parts
    GLine rightLeg1=new GLine(0,0,HIP_WIDTH,0);
    GLine rightLeg2=new GLine(0,0,0,LEG_LENGTH);
    //constructor
    RightLeg(int x,int y){
        add(rightLeg1,x,y);
        add(rightLeg2,x+HIP_WIDTH,y);
    }
}
public class GHangman extends GCompound{
    //parts
    GLine scaffold=new GLine(0,0,0,SCAFFOLD_HEIGHT);
    GLine beam=new GLine(0,0,BEAM_LENGTH,0);
    GLine rope=new GLine(0,0,0,ROPE_LENGTH);
    GOval head=new GOval(2*HEAD_RADIUS,2*HEAD_RADIUS);
    GLine body=new GLine(0,0,0,BODY_LENGTH);
    LeftArm leftArm=new LeftArm(0,0);
    RightArm rightArm=new RightArm(0,0);
    LeftLeg leftLeg=new LeftLeg(0,0);
    RightLeg rightLeg=new RightLeg(0,0);
    GLine leftFoot=new GLine(FOOT_LENGTH,0,0,0);
```

```
GLine rightFoot=new GLine(0,0,FOOT_LENGTH,0);
//coordinates
double originX=0;
double originY=0;
GHangman(double x,double y){
   drawAll(x,y);
}
private void drawAll(double x,double y){
   originX=x;
   originY=y;
   add(scaffold,x,y);
   add(beam,x,y);
   add(rope,x+BEAM_LENGTH,y);
   add(head,x+BEAM_LENGTH-HEAD_RADIUS,y+ROPE_LENGTH);
   add(body,x+BEAM_LENGTH,y+ROPE_LENGTH+2*HEAD_RADIUS);
   add(leftArm,x+BEAM_LENGTH,y+ROPE_LENGTH+2*HEAD_RADIUS+
      ARM_OFFSET_FROM_HEAD);
   add(rightArm,x+BEAM_LENGTH,y+ROPE_LENGTH+2*HEAD_RADIUS+
      ARM_OFFSET_FROM_HEAD);
   add(leftLeg,x+BEAM_LENGTH,y+ROPE_LENGTH+2*HEAD_RADIUS+
      BODY_LENGTH);
   add(rightLeg,x+BEAM_LENGTH,y+ROPE_LENGTH+2*HEAD_RADIUS+
      BODY_LENGTH);
   add(leftFoot,x+BEAM_LENGTH-HIP_WIDTH,y+ROPE_LENGTH+
      2*HEAD_RADIUS+BODY_LENGTH+LEG_LENGTH);
   add(rightFoot,x+BEAM_LENGTH+HIP_WIDTH,y+ROPE_LENGTH+
      2*HEAD_RADIUS+BODY_LENGTH+LEG_LENGTH);
}
public void redraw(int lives){
   remove(this);
   drawAll(originX,originY);
   //System.out.println("Lives: "+lives);
   switch(lives){
      case 8:
         remove(man.head);
         remove(man.rope);
      case 7:
         remove(man.body);
      case 6:
         remove(man.leftArm);
      case 5:
         remove(man.rightArm);
      case 4:
         remove(man.leftLeg);
      case 3:
         remove(man.rightLeg);
      case 2:
```

```
                remove(man.leftFoot);
            case 1:
                remove(man.rightFoot);
            case 0:
                break;

            default:
                throw new LivesException("Invalid lives: "+lives);
        }
    }
    public class LivesException extends RuntimeException{
        public LivesException(String s){
            super(s);
        }//end constructor
    }
}

/** Resets the display so that only the scaffold appears */
public void roundBegin(int lives,String coded){
    removeAll();
    wordDisp.setColor(Color.black);
    wrongDisp.setLabel("");
    add(man,(getWidth()-man.getWidth())/2,HANGMAN_Y_OFFSET);
    man.redraw(lives);
    wordDisp.setFont(WORD_FONT);
    wordDisp.setLabel(coded);
    wrongDisp.setFont(WRONG_FONT);
    add(wrongDisp,getWidth()/4,getHeight()-WRONG_Y_OFFSET);
    add(wrongDisp0,getWidth()/4,wrongDisp.getY()-wrongDisp.getHeight()*1.25);

add(wordDisp,(getWidth()-wordDisp.getWidth())/2,wrongDisp0.getY()-WORD_OFFSET);
}
public void update(int lives,String coded,char incorrect){
    man.redraw(lives);
    //coded
    double x=wordDisp.getX();
    double y=wordDisp.getY();
    remove(wordDisp);
    wordDisp.setLabel(coded);
    add(wordDisp,(getWidth()-wordDisp.getWidth())/2,y);
    //wrong
    x=wrongDisp.getX();
    y=wrongDisp.getY();
    remove(wrongDisp);
    wrongDisp.setLabel(wrongDisp.getLabel()+incorrect+" ");
    add(wrongDisp,x,y);
}
```

```
    public void update(int lives,String coded){
        man.redraw(lives);
        //coded
        double x=wordDisp.getX();
        double y=wordDisp.getY();
        remove(wordDisp);
        wordDisp.setLabel(coded);
        add(wordDisp,(getWidth()-wordDisp.getWidth())/2,y);
    }
}
```

HangmanLexicon:

```
/*
 * File: HangmanLexicon.java
 * -------------------------
 * This file contains a stub implementation of the HangmanLexicon
 * class that you will reimplement for Part III of the assignment.
 */
package ph.edu.dlsu.chan.hangman;
import acm.util.*;
import java.io.*;

public class HangmanLexicon {
    //global vars
    FileReader in;
    BufferedReader br;
    MyLinkedList<String> words=new MyLinkedList<>();
    int wordCount=0;
    //constructor
    HangmanLexicon(){
        String filePath = new File("").getAbsolutePath();
        //System.out.println (filePath);
        try{
            in = new FileReader(filePath+"/HangmanLexicon.txt");//throws exception
        } catch (IOException e){
            System.out.println("File not found.");
        }
        br=new BufferedReader(in);
        System.out.println("Loading...\n");
        System.out.println("Words Loaded:");
        //store words to linked list
        words.createList();
```

```
      String a="";
      while(a!=null){
         try{
            a=br.readLine();
         } catch(IOException e){
            System.out.println("File read error; wordCount:"+wordCount+
                  "\nlast item:"+words.size()+") "+
                  words.get(words.size()));
         }
         wordCount++;
         words.addLast(a);
         //if(wordCount%10==0)
         System.out.print(wordCount+"\r");
      }
      System.out.println("\nDONE!");
   }

   //reload preloaded words
   public HangmanLexicon(MyLinkedList<String> words) {
      System.out.println("The words have been preloaded...");
      this.words=words;
      wordCount=words.size();
      System.out.println("Words loaded: "+words.size()+"\n Done!");
   }


   /*//file read
   public BufferedReader read() throws IOException{
      String filePath = new File("").getAbsolutePath();
      //System.out.println (filePath);
      FileReader in = new FileReader(filePath+"/src/a.txt");//throws exception

   }*/

/** Returns the number of words in the lexicon. */
   public int getWordCount() {
         return wordCount;
   }

/** Returns the word at the specified index. */
   public String getWord(int index) {
      return words.get(index);
   }
}
```

List.java

```
/*
 * File: List.java
 * ----------------
 * This is the List ADT definition
 */
package ph.edu.dlsu.chan.hangman;

public interface List<E>{


  public void createList();
  // precondition: none
  // postcondition: Create an empty list


  public void add(int index, E item) throws ListIndexOutOfBoundsException;
// precondition: index (to be added) is within the position of the list of items,
1<=index<=size()+1
// postcondition: Insert item at position index of a list
// if 1<=index<= size()+1. If index <= size(), items
// at position index onwards are shifted one position
// to the right. Throws an exception when index is out of range.



  public void remove(int index) throws ListIndexOutOfBoundsException;
// precondition: index (to be removed) is within the position of the list of items,
1<=index<=size()
// postcondition: Remove item at position index of a list
// if 1<=index<= size(). Items at position
// index+1 onwards are shifted one position to the left
// Throws an exception when index is out of range, or if list is empty.



 public boolean isEmpty();
// precondition: none
// postcondition: Determine if a list is empty



public E get(int index) throws ListIndexOutOfBoundsException;
// precondition: index is within the position of the list of items, 1<=index<=size()
// postcondition: Returns item at position index of
```

```
// a list if 1<=index<=size(). Throws an exception if index is out of range.



public int size();
// precondition: none
// postcondition: Returns number of items in a list

}
```

ListEmptyException.java

```
/*
 * File: ListEmptyException.java
 * -----------------
 * This class handles a Empty list
 */
package ph.edu.dlsu.chan.hangman;
public class ListEmptyException extends RuntimeException{
    public ListEmptyException(String s){
        super(s);
    }//end constructor
} //end ListException
```

ListIndexOutOfBoundsException.java

```
/*
 * File: ListIndexOutOfBoundsException.java
 * -----------------
 * This class handles the case when list index given is out of bounds
 */
package ph.edu.dlsu.chan.hangman;

public class ListIndexOutOfBoundsException extends IndexOutOfBoundsException{
      public ListIndexOutOfBoundsException(String s){
        super(s);
      }//end constructor
} //end ListIndexOutofBoundsException
```

MyLinkedList.java (I improved the implementation here)

```
package ph.edu.dlsu.chan.hangman;
public class MyLinkedList<E> implements List<E>{//1-based
   private Node<E> head;//first node
   private Node<E> tail;//last node
   private int size;
   private Node<E> curNode;//previously used node(in getNode),chance--quicker
searches
   private int curNodeIndex;

   public MyLinkedList() {
      createList();
   }
   public void createList(){
     size = 0;
     head = null;
     tail = null;
     curNode=null;
     curNodeIndex=0;
   } //end constructor

   //methods:
   /**
    * Locate a specified node in a linked list:
    * @param index index of the node
    * @throws ListIndexOutOfBoundsException if
    *  index not w/in current size, or empty.
    *  (i.e. if index>numItems or index<1).
    * @return the node of specified index
    */
   private Node<E> getNode(int index){
   // precondition: index is the number of the desired node
   // postcondition: returns a reference to the desired node.
   // postcondition: must also reassign curNode accordingly.
   //*****************************************************//
   //Note: curNode must always be assigned a value here to
   //  prevent curnode indexing problems for add and delete.
   //*****************************************************//
      if(!isEmpty() && index>=1 && index<=size){//validity
         if(index==1){
            curNodeIndex=1;
            curNode=head;
            return head;
         } else if(index==size){
            //tail is useless as curnode
```

```
            curNodeIndex=0;
            curNode=null;//just to be safe,else prone to errors
            return tail;
         } else {
            //if curnode cannot be used
            if(curNode==null || curNodeIndex>index){
               curNode=head;
               curNodeIndex=1;
            }
            while(curNodeIndex<index){
               curNode=curNode.getNext();
               curNodeIndex++;
            }
            return curNode;
         }
      } else {
         throw new ListIndexOutOfBoundsException("find--error");
      }
   }

   //Checkers:
   public E get(int index) throws ListIndexOutOfBoundsException{
      try{
         return getNode(index).getItem();
      } catch (ListIndexOutOfBoundsException e){
         throw new ListIndexOutOfBoundsException("GET ERROR: List Index Out Of
Bounds");
      }
   }
   public int size(){
      return size;
   }
   public boolean isEmpty(){
      return size == 0;
   }

   //Modifiers:
   //ADD
   public void add(int index, E item) throws ListIndexOutOfBoundsException {
      if(index==1){//curNode always greater than this**
         addFirst(item);
      } else if(index==size+1){//curNode always less than this
         addLast(item);
      } else {//curnode auto reassigned to be 1 less than index
         Node<E> prev;
         try{
            prev=getNode(index-1);
```

```
        } catch (ListIndexOutOfBoundsException e){
            throw new ListIndexOutOfBoundsException("GET ERROR: List Index Out Of
Bounds");
        }
        Node<E> newNode=new Node<>(item,prev.getNext());
        prev.setNext(newNode);
        size++;
    }
  }
  public void addFirst(E item){
    Node<E> newNode=new Node<>(item,head);
    head=newNode;
    curNodeIndex++;
    size++;
    if(size==1){//if list empty prior
        tail=head;
    }
  }
  public void addLast(E item){
    if(size==0){//if empty list
        addFirst(item);
    } else {
        Node<E> newNode=new Node(item,null);
        tail.setNext(newNode);
        tail=newNode;
        size++;
    }
  }
  //REMOVE
  public void remove(int index) throws ListIndexOutOfBoundsException{
    if(index==1){//curNode always greater than this**
        removeFirst();//////////////////////////^^
    } else {//curnode always reassigned to 1 less than index
        Node<E> prev;
        try{
            prev=getNode(index-1);
        } catch (ListIndexOutOfBoundsException e){
            throw new ListIndexOutOfBoundsException("GET ERROR: List Index Out Of
Bounds");
        }
        prev.setNext(prev.getNext().getNext());
        if(prev.getNext()==null){
            tail=prev;
        }
        size--;
    }
  }
```

```
    public void removeFirst(){
        head=head.getNext();
        curNodeIndex--;
        size--;
        if(size==0){//if list becomes empty
            tail=null;
        }
    }
}//end class
```

Node.java

```
package ph.edu.dlsu.chan.hangman;
public class Node<E>{
   private E item;
   private Node<E> next;

   public Node(E newItem, Node<E> nextNode){
        item = newItem; // DATA
        next = nextNode; //POINTER/LINK
   }

   public Node(E newItem){
        item = newItem;
        next = null;
   }

   public void setItem(E newItem){
         item = newItem;
   }

   public E getItem(){
         return item;
   }

   public void setNext(Node<E> nextNode){
         next = nextNode;
   }

   public Node<E> getNext(){
         return next;
   }
}// end class Node
```

REFERENCES

1. E Roberts. *Art and Science of Java*. Pearson; 2013.
2. E Roberts, M Sahami, and M Stepp, *CS 106A: Programming Methodology (Java) Handouts*, Stanford University.