Appendix

Consultation with client:

Client wanted a wheel of fortune type of activity to be used in the school lesson. Client also wanted to use it as an after school activity. Initially, the client wanted to have a hangman style game, however the wheel of fortune style activity offered more versatility within the school lesson.

Feedback from client:

The client was satisfied with the program. The activity worked well with the students within the school lessons. The activity was enjoyable for the students after school as well. However, the client disliked the cramped GUI. The options and the output along with the word being displayed were too close and hard to read at first. Also, the client wished to have an easier way of adding other phrases or money amount to have different difficulties. In conclusion, the client still enjoyed the product and was very useful.

```java
public class PlayGame {
    //main method
    public static void main(String[] args) {

        //Object of Game class
        Game game = new Game();
        game.gameMenu();//call method
    }
}
```

```java
import java.util.Scanner;

//class used to get user input
public class UserInputs {

    Scanner scan = new Scanner(System.in);

    //get string from user
    public String getInput(){
        String line = "";
        line = scan.nextLine();

        return line;
    }
}
```

```
//Libraries
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.Scanner;

//class
public class Game{

    //Instance of class

    // To read from the keyboard
    //private static final Scanner input = new Scanner(System.in);
    static UserInputs input  = new UserInputs();
    // Used to get random values for puzzle and wheel
    private static final Random random = new Random();

    // True if we want to show all letters
    private static boolean letters = false;
    private static int win_cash = 0;
    private static int vowel_Cash = 250;
    private static char char_puzzle=' ';

    private static final List<String> game_wedges = Arrays.asList(
            "$5000",
            "$600",
            "$500",
            "$300",
            "$500",
            "$800",
            "$550",
            "$400",
            "$300",
            "$900",
            "$500",
            "$300",
            "$900",
            "BANKRUPT",
            "$600",
            "$400",
            "$300",
            "$800",
            "$350",
            "$450",
            "$700",
            "$300",
            "$600"
        );

    /*
     * The number of wedges will not change throughout the game
     * We can cache the value so we're not calling .size() over and over
     */
```

```
private static final int wedge_count = game_wedges.size();

private static String chooseRandomWedgeValue() {
    // Choose a random index
    int randomWedgeIndex = random.nextInt(wedge_count);
    //System.out.println("Money is :" +randomWedgeIndex);

    // Return the corresponding wedge
    return game_wedges.get(randomWedgeIndex);
}

// The menu choices
private static final List<String> _menuChoices = Arrays.asList(
        "1. Spin the wheel",
        "2. Buy a vowel",
        "3. Solve the puzzle",
        "4. Check your balance ",
        "5. Quit the game"
    );
private static final int _quitChoiceNumber = 5;

// The different puzzles to choose from
private static final List<String> _puzzles = Arrays.asList(
        "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG",
        "WAR BEH KEH",
        "INFORMATION SCIENCES AND TECHNOLOGY",
        "A DIAMOND IN THE ROUGH",
        "BEATING THE ODDS",
        "HOW ARE YOU",
        "ZOOM IN ZOOM OUT"
    );

/*
 * The number of puzzles will not change throughout the game
 * We can cache the value so we're not calling .size() over and over
 */
private static final int _puzzlesCount = _puzzles.size();

private static Map<Character, Boolean> guessedLetters = new HashMap<>();

/*
 * Given a puzzle, return a masked version, with hidden letters
 */
public static String maskPuzzle(String puzzle, boolean revealLetters) {
    // Use a string builder, since Java strings are immutable
    StringBuilder maskedPuzzle = new StringBuilder();

    // For each letter in the puzzle
    for (int i = 0; i < puzzle.length(); i++) {
        // Current letter
        char_puzzle = puzzle.charAt(i);

        /*
         * Either we're revealing all letters, or we've already guessed the
         * letter
         */
```

```
        boolean isLetterGuessed = revealLetters || guessedLetters.containsKey(char_puzzle);

        /*
         * If the letter is not blank (we don't mask blanks), and the letter
         * has not been guessed, then we will mask it.
         */
        if (char_puzzle != ' ' && !isLetterGuessed){
            char_puzzle = '_';
        }
        // Put one space after each character (even a space) in the puzzle
        maskedPuzzle.append(char_puzzle + " ");
    }
    // Convert the string builder to a string and return it
    return maskedPuzzle.toString();
}

// Choose a random puzzle
public static String chooseRandomPuzzle() {
    // Choose a random puzzle index
    int randomPuzzleIndex = random.nextInt(_puzzlesCount);

    //Return the corresponding puzzle
    return _puzzles.get(randomPuzzleIndex);
}

// Determine if the given number choice actually appears on the menu
public static boolean isValidMenuChoice(int choice) {
    if ((choice < 1) || (choice > _menuChoices.size())) {
        return false;
    }
    // Subtrace 1 because arrays/lists are zero-based
    int index = choice - 1;
    String menuText = _menuChoices.get(index);

    return !menuText.equals("");
}

// Input a letter from the keyboard
public static char inputLetter(int inputCase) {
    char letter = ' ';
    boolean finished = false;
    while (!finished) {
        finished = true;
        System.out.print("Enter a letter: ");

        String line = input.getInput();
        if (line.length() != 1) {
            System.out.println("Enter just one letter");
            finished = false;
        } else {
            // Convert letter to upper case
            letter = Character.toUpperCase(line.charAt(0));
            if (!Character.isLetter(letter)) {
                System.out.println("That is not a letter");
                finished = false;
            }
```

```
            /*
            if(!guessedLetters.containsKey(letter)){

            System.out.println("Incorrect Letter -> You Loss");

            }*/
            if(guessedLetters.containsKey(letter)){
                System.out.println("Already Guessed, Enter another Letter");
                finished = false;
            }
            //Testing for vowel when spinning the wheel
            if (inputCase == 1){
                if (letter == 'A' || letter == 'E' || letter == 'I' || letter == 'O' || letter == 'U') {
                    System.out.println("That's a vowel!");
                    finished = false;
                }
            }
            //Testing for consonants when buying vowels
            if (inputCase == 2){
                if (letter == 'A' || letter == 'E' || letter == 'I' || letter == 'O' || letter == 'U') {

                } else {
                    System.out.println("That's not a vowel!");
                    finished = false;
                }
            }
        }
    }
    return letter;
}

// Display the game menu, and handle the choices made
public static void gameMenu() {
    // Choice from the menu
    int choice = 0;

    // Line entered from keyboard
    String line = "";

    // True when user wants to quit
    boolean quit = false;

    // Choose one of the puzzles at random
    String puzzle = chooseRandomPuzzle();

    String str = chooseRandomWedgeValue();
    String balance=str.substring(1);
    win_cash = win_cash + Integer.parseInt(balance);

    // Repeat the menu until the user chooses to quit
    while (!quit) {
        System.out.println("                    ----------------------");
        System.out.println("                    *        PUZZLE       *");
        System.out.println("                    ----------------------");
        System.out.println("                                          ");
```

```
System.out.println(maskPuzzle(puzzle, letters));
System.out.println();
if(win_cash <= 0){
    System.out.println("You don't have enough balance to play!!!");
    System.out.println("***GAME OVER***");
    System.exit(0);

}
// Loop over the menu choices, and display each one
for (String menuChoice : _menuChoices) {
    // Skip blank place-holder choices
    if (!menuChoice.equals("")) {
        System.out.println(menuChoice);
    }
}
System.out.print("Enter choice: ");
line = input.getInput();
try {
    // If the input was not an integer, then that error will be caught
    choice = Integer.parseInt(line);
} catch (NumberFormatException e) {
    // Error message, then go to the top of the loop
    System.out.println("Invalid input");
    win_cash = win_cash -50;
    //System.out.println("Remaining Cash is : $" +win_cash );
    continue;
}

// If not valid, then go back to the top of the loop
if (!isValidMenuChoice(choice)) {
    System.out.println("Not a menu choice");
    continue;
}

System.out.println("You chose: " + _menuChoices.get(choice - 1));
switch (choice) {
    case _quitChoiceNumber:
    // This will allow us to leave the menu loop
    quit = true;
    break;

    case 1: // Spin the wheel
    str = chooseRandomWedgeValue();
    if(str.equals("BANKRUPT")){
        win_cash = 0;
        System.out.println("You landed on: " + str + " ->" +"Your cash is : $" +win_cash);
        System.exit(0);
    }
    char letter = inputLetter(1);

    guessedLetters.put(letter, true);
    // guessedLetters.put(solve_puzzel, true);
    String guess_puzzle = maskPuzzle(puzzle, letters);
    //System.out.println(":("+guess1);
    String _n1guess= guess_puzzle.replaceAll("\\s+","");
```

```
String _p1guess= puzzle.replaceAll("\\s+","");
//System.out.println("Match :" +_n1guess +" :" +_p1guess);
if(_n1guess.equalsIgnoreCase(_p1guess)){
   System.out.println("YOU WON!!!" +win_cash);
   System.exit(0);
}
System.out.println("Your letter is: " + letter);
if(!puzzle.contains(letter+"")){
   System.out.println("Invalid Guess!!!");
   win_cash = win_cash - 150;
   //System.out.println("Your Cash is : $" + win_cash);
}
else{

   String won=str.substring(1);
   System.out.println("You Won: $" +won);
   win_cash = win_cash + Integer.parseInt(won);

}
break;

case 2:
char vowel = inputLetter(2);
if(win_cash < vowel_Cash){
   System.out.println("You don't have enough money to buy Vowel");
   break;
} else if(win_cash >=vowel_Cash ){
   System.out.println("Enter vowel: " + vowel);
   guessedLetters.put(vowel, true);
   win_cash = win_cash - vowel_Cash;
   //System.out.println("Your Cash is : $" + win_cash);
   break;
}
break;

case 3:
System.out.println("Puzzle : " + maskPuzzle(puzzle, letters)) ;
for(int i=0; i< puzzle.length() ; i++){
   //charPuzzel = puzzle.charAt(i);
   char solve_puzzel = inputLetter(3);
   guessedLetters.put(solve_puzzel, true);
   if(!puzzle.contains(solve_puzzel+"")){
      win_cash = 0;
      System.out.println("Incorrect Letter -> You Lose!!" +" Your cash is : $" +win_cash);
      System.out.println("***GAME OVER***");
      System.exit(0);

      //System.exit(1);
   }
   // guessedLetters.put(solve_puzzel, true);
   String guess = maskPuzzle(puzzle, letters);
   System.out.println(": "+ guess);
   String _nguess= guess.replaceAll("\\s+","");

   String _pguess= puzzle.replaceAll("\\s+","");
```

```
                //System.out.println("Match :" +_nguess);
                if(_nguess.equalsIgnoreCase(_pguess)){
                    System.out.println("YOU WON!!!");
                    System.exit(0);
                }
            }
            break;

            case 4:
            System.out.println("Your Cash is : $" + win_cash);
        }
    }
  }
}
```