Lab 4 : Loops and the Design Process – Selected Solutions

Exercise 4: As a class, you are going to design a simple Java program that will calculate whether an input word is a palindrome or not. A palindrome is any string that reads the same forwards as it does backwards.

One possible solution:
Assumptions made: an input string is a palindrome if it reads the same forwards as it does backwards, ignoring all whitespace and case. Numbers and special symbols are not ignored.

```java
import java.util.Scanner;

public class Palindromes {
   public static void main(String[] args) {
      Scanner keyboard = new Scanner(System.in);

      System.out.println("Enter an input string.");
      String input = keyboard.nextLine();

      // convert all to lowercase so case is ignored
      input = input.toLowerCase();

      // remove all whitespace, so whitespace is ignored
      input = input.replaceAll("\\s+", "");

      int index = 0;
      while(index < input.length() / 2) {
         // get the letter in the first half of input
         char frontLetter = input.charAt(index);
         // get the corresponding letter in the second half
         char endLetter = input.charAt(input.length() - index - 1);

         if(frontLetter != endLetter) {
            System.out.println("Input is not a palindrome.");
            return;
         }

         index++;
      }

      System.out.println("Input is a palindrome.");
   }
}
```

Exercise 5: Create a `Pyramid` class to print pyramid pattern using asterisk character. The levels the pyramid triangle would have will be read from command-line argument. You will need to use *nested loops*(one loop inside another) for this exercise.

```java
public class Pyramid {
   public static void main(String[] args) {
      int pyramidLevel = Integer.parseInt(args[0]);
      for (int i = 0; i < pyramidLevel; i++) {
            for (int j = 0; j < pyramidLevel - i; j++) {
                System.out.print(" ");
            }
            for (int k = 0; k <= i; k++) {
                System.out.print("* ");
            }
            System.out.println();
            }
        }
}
```

Exercise 6: The diagram to the right represents a program that takes in a string input (called `input`), and counts the number of double letters (e.g. “mm”, “ss”) in that input. Double vowels (e.g. “oo”, “ee”) count as two sets of double letters. Anything that is not a letter is ignored.

For example, `input = "hello"` would get `count = 1`, `input = "book"` would get `count = 2` and `input = "balloon"` would get `count = 3`.

There are three errors in this diagram that make the logic incorrect. Perform a desk check on various inputs to the program in order to find and correct these errors. Draw the corrected diagram in your log book.

To get you started, try performing a desk check on `input = "value"`, `input = "Hi there`”!, and `input = "foo"`.

Incorrect code:

```
import java.util.Scanner;

public class DoubleLettersIncorrect {
   public static void main(String[] args) {
      Scanner keyboard = new Scanner(System.in);
      String input = keyboard.next();

      int count = 0;
      int index = 0;

      while(index < input.length()) {
         char c1 = input.charAt(index);
         if(!Character.isLetter(c1)) {
            continue;
         }

         char c2 = input.charAt(index + 1);
         if(c1 == c2) {
            count = count + 1;
         }

         if("AEIOUaeiou".contains(""+c1)) {
            count = count + 1;
         }

         index = index + 1;
      }

      System.out.println("Count: " + count);
   }
}
```

Exercise 8: Check that you have correctly identified each of the errors with your tutor. If you have, convert the diagram to Java code. You will find conversions for some of the more complicated components below.

```java
import java.util.Scanner;

public class DoubleLetters {
   public static void main(String[] args) {
      Scanner keyboard = new Scanner(System.in);
      String input = keyboard.next();

      int count = 0;
      int index = 0;

      // Change to input.length() - 1 to prevent
      // out of bounds exceptions
      while(index < input.length() - 1) {
         char c1 = input.charAt(index);
         if(!Character.isLetter(c1)) {
            // Need to increase index, otherwise
            // you get an infinite loop
            index = index + 1;
            continue;
         }

         char c2 = input.charAt(index + 1);
         if(c1 == c2) {
            count = count + 1;
            // Vowel condition should only be checked
            // if the letters are doubled
            if("AEIOUaeiou".contains(""+c1)) {
               count = count + 1;
            }
         }

         index = index + 1;
      }

      System.out.println("Count: " + count);
   }
}
```

Extension 1: write a program called `Binary` to read a positive number from command-line argument, and ouput its binary formant. Please write your own method rather than apply some existing method in java, such as `Integer.toBinaryString(number)`.

```java
public class Binary {
    public static void main(String[] args) {
      int number = Integer.parseInt(args[0]);
      if (number < 0){
         System.out.println(Please input a positive number!);
         return;
      }

      // set testValue to the largest power of 2 that is less or equal to number
      int testValue = 1;
      while (testValue <= number/2) {
            testValue = testValue * 2;
      }
```

```
        // check for presence of powers of 2 in number from largest to smallest
        while (testValue > 0) {
            // testValue is not present in number
            if (number < testValue) {
                System.out.print(0);
            }
            // testValue is present in number
            else {
                System.out.print(1);
                number = number - testValue;
            }
            // next smallest power of 2
            testValue = testValue / 2;
        }
        System.out.println();
    }
}
```