Lab 4 : Loops and the Design Process

Topics covered while loops, `break` and `continue` keywords, software design process

Exercise 1: Follow your tutor's instructions to complete the mid-semester survey for this course on Blackboard. All feedback (positive or negative) is appreciated and completely anonymous.

Exercise 2: On paper, draw a diagram for each of the following control structures:

1. an `if`/`else` block
2. a simple `while` loop
3. a `while` loop with a `break` statement
4. a `while` loop with a `continue` statement

Exercise 3: Draw a diagram for a program that would print all the odd numbers from 1 to 99, separated by commas. Make sure that there would be no trailing comma after the last number!

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.

Part 1. Discuss what the question is asking. What are the desired inputs and outputs of the program? What assumptions will you make about those inputs and outputs?

Part 2. What types of variables and control structures will be required in order to solve this task?

Part 3. Draw a diagram to describe the program using your answers from the previous part.

Part 4. Design a set of test data for the program. Does this data cover all possible paths in your diagram?

Part 5. Perform a desk check with the input data. Are all outputs as expected? If not, modify your diagram so that the program behaves as expected.

Part 6. Once you are happy with your diagram, convert it to Java code.

Exercise 5: Write a program called `Factors` that accepts a positive integer argument from the user and then prints all the factors of that number on one line.

You will possibly find this pseudocode helpful:

Algorithm 3: PrintFactors (n)

```
1  let $n$ be a positive integer
2  if ($n < 1$) then
3      print "this has no factors of interest"
4      return
5  let $i$ be an integer
6  $i \leftarrow n$
7  while ($i > 0$) do
8      if ($i$ is a factor of $n$) then
9          print $i$
10     $i \leftarrow i - 1$
11 print "All done."
```

Now make it print out the factors in increasing order. You'll have to change the initialisation of the loop (currently you set i to n: line 6 of the pseudocode), the condition to test (line 7), and the way you change i (line 10).

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"`.

Exercise 7: Once you think you have identified all the errors in the logic, identify what kind of control structure each of the following components are:

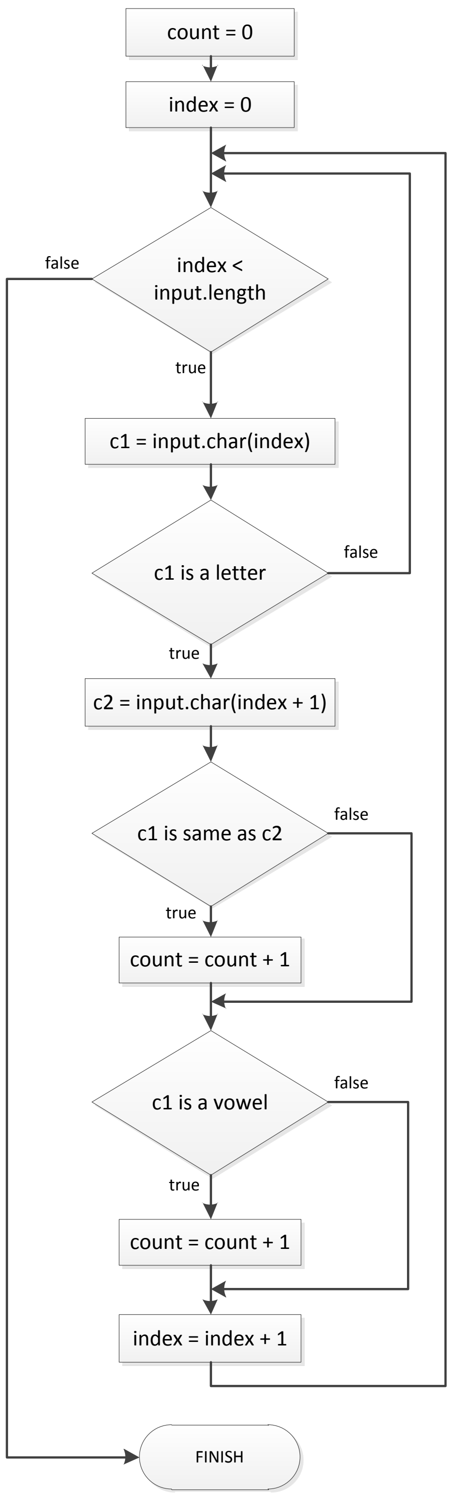
1. the condition containing `index < input.length`
2. the condition containing `c1 is same as c2`
3. the condition containing `c1 is a vowel`
4. the `false` branch of the condition `c1 is a letter`

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
// c1 = input.char(index)
char c1 = input.charAt(index);

// c1 is a letter
boolean isLetter =
    Character.isLetter(c1);

// c1 is a vowel
boolean isVowel =
    "aeiouAEIOU".contains(""+c1);
```



Exercise 9: Write a program that uses an array to store integers read from Standard Input (Scanner).

You can store up to 20 elements, if -1 is enountered, the program should stop reading from Scanner and not include it in the array.

We encourage you try this on ed's lessons.

```java
import java.util.Scanner;

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


   }
}
```

The program should output:

```
Collected Numbers: { 1, 2, 3, 4, 5, 6 }
```