

**Tshwane University of Technology**

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

# **Principles of Programming A (Extended)**

**PPAF05D/TROF05D**

## **Unit 5**

### **Iteration control structures in Java**

**DEVELOPED: 2019-2021**

**© COPYRIGHT: TSHWANE UNIVERSITY OF TECHNOLOGY (2020)**

All rights reserved. Apart from any reasonable quotations for the purposes of research criticism or review as permitted under the Copyright Act, no part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy and recording, without permission in writing from the publisher.

<b>Unit 5: Iteration control structures in Java .....</b>	<b>1</b>
<b>5.1 Using while loops .....</b>	<b>3</b>
5.1.1 Basic structure of a while loop .....	3
Activity 1: Repeat the same calculations.....	6
5.1.2 Preventing errors.....	7
Activity 2: Use a while loop to prevent errors .....	12
Activity 3: While loop terminologies .....	14
5.1.3 Creating simple games .....	15
Activity 4: Flow chart for guessing game .....	18
Activity 5: Small games .....	19
5.1.4 Letting the user control the number of repetitions .....	20
Activity 6: Repeat the same calculations a set number of times .....	22
Activity 7: Sentinel controlled repetitions .....	24
5.1.5 Calculating cumulative values .....	25
Activity 8: Calculate cumulative values .....	28
5.1.6 Finding the highest value from a list of values .....	28
Activity 9: Find the lowest value.....	30
Activity 10: Display the name with the lowest mark.....	30
5.1.7 Creating a menu .....	30
Activity 11: Improve Pizza toppings program .....	31
Activity 12: Use menus .....	32
<b>5.2 Using a do-while loop .....</b>	<b>33</b>
5.2.1 Basic structure of a do-while loop .....	33
5.2.2 Comparison between a while and a do-while loop .....	36
Activity 13: Problems using a do-while loop .....	37
<b>5.3 Using for loops .....</b>	<b>40</b>
5.3.1 Basic structure of a for loop .....	40

<i>Activity 14: Problems using a for loop</i> .....	45
5.3.2 Trace table for a for loop.....	47
<i>Activity 15: Create a trace table for a for loop</i> .....	50
5.3.3 Variations of the expression to change the loop control variable .....	52
<i>Activity 16: Convert kilogram to pounds</i> .....	53
<i>Activity 17: Display values in reverse order</i> .....	54
<i>Activity 18: Predict the number of iterations of a for loop</i> .....	55
<i>Activity 19: Predict the output of programs using a for loop</i> .....	55
<i>Activity 20: Solve problems using a for loop</i> .....	57
<b>5.4 Loop control</b> .....	<b>60</b>
5.4.1 Count-controlled loops.....	60
5.4.2 Sentinel-controlled loops .....	61
5.4.3 Result-controlled loops.....	62
<b>5.5 Using nested structures</b> .....	<b>63</b>
<b>5.6 More challenging activities implementing a while loop (not for assessment)</b> .....	<b>70</b>
5.6.1 To do mathematical calculations.....	70
<i>Activity 21: Find the largest factor of a number</i> .....	70
<i>Activity 22: Improve algorithm to determine prime number</i> .....	72
<i>Activity 23: Mathematical calculations using a while loop</i> .....	72
<b>Works referenced and/or consulted</b> .....	<b>73</b>
<b>Good alternative sources</b> .....	<b>74</b>

# Unit 5: Iteration control structures in Java

## Assessment criteria

The content covered in this unit relates to the following outcomes and assessment criteria:

**UNIT 5:** Decision making and repetition in programming.

**Outcome:** The student must be able to develop programming solutions involving decisions and repetition.

### Assessment Criteria

- An algorithm to solve a basic mathematical problem requiring counter controlled, pre-test and post-test repetition structures can be designed, presented in pseudo code, and implemented in a programming language to provide correct output.
- Output for algorithms using decision and repetition structures can be predicted by means of a trace table.
- Infinite loops can be identified, and solutions provided in a program.
- An algorithm to solve a basic mathematical problem requiring nested decision making and repetition structures can be designed, presented in pseudo code, and implemented in a programming language to provide correct output.

## Knowledge and skills needed

The knowledge and skills as described in the following Units and Outcomes are relevant and have to be applied in this Unit:

**Unit 1:** Problem solving.

**Outcome:** The student must be able to apply logic and knowledge to solve basic problems.

**Unit 2:** Basic programming principles.

**Outcome:** The student must be able to apply algorithmic problem solving to basic mathematical and programming problems.

In Unit 4 you learned how to write programs where decisions were made so that only certain program statements were executed. In this unit you are going to learn how to write programs where certain program statements have to be executed *repeatedly* (*iteratively*). In programming terminology, a *loop* is a structure that will make the program repeatedly execute certain statements. When you play a song, you can set it to loop, which means that when it reaches the end it starts over at the beginning. A loop in programming, also called iteration or repetition, is a way to repeat one or more statements.

In Scratch you used two loop structures. The [repeat until **<condition>**], and the [repeat **<number of times>**] control structures (program blocks). The C-shaped blocks of these loop structures clearly shows you which blocks should be repeated.

- Iterative
- Loop

These blocks are inside the loop and will be repeated until the user pushes the space bar on the keyboard.

These blocks are 'inside the loop' and will be repeated the number of times indicated by the user (the value stored in the variable iJumps).

The [say] block is 'outside the loop' and will only be executed once - when the loop has stopped.

---

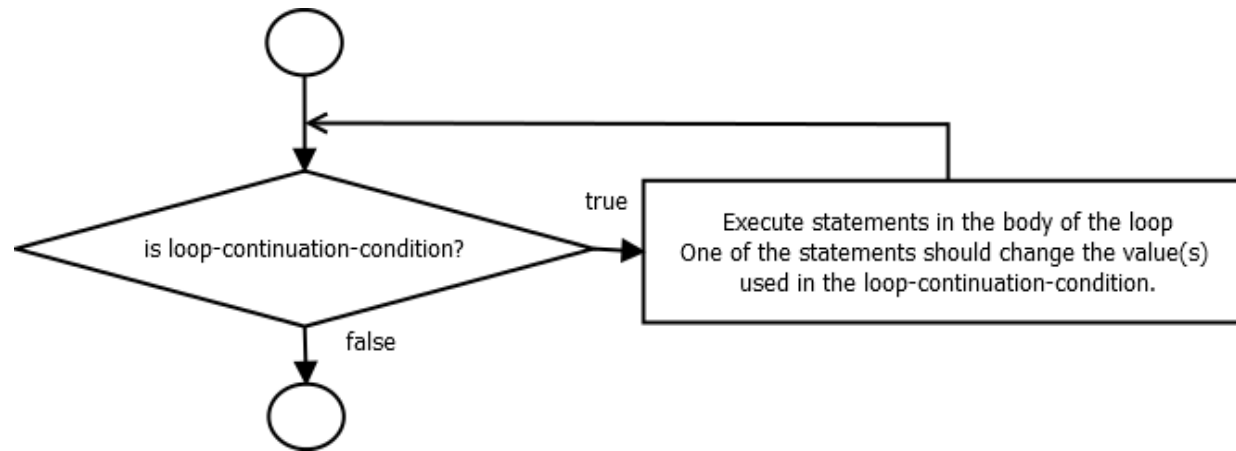
In Java there are three types of loop statements / iterative control structures. They are **while** loops, **do while** loops, and **for** loops.

---

## 5.1 Using while loops

### 5.1.1 Basic structure of a while loop

A while loop is used to repeatedly execute a number of statements, *while a certain condition is true*. In general, a flowchart for a while loop looks as follows:

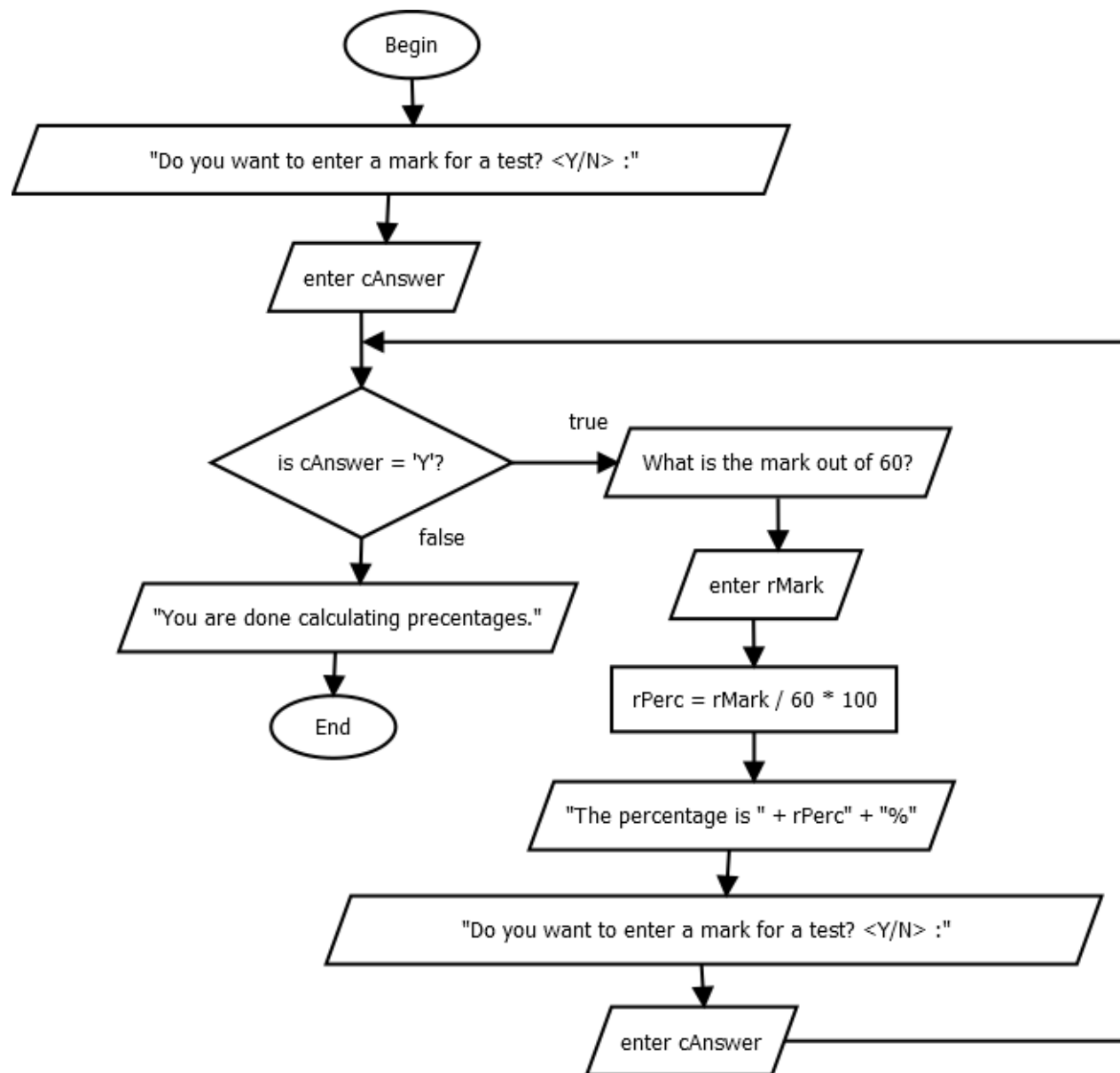


(HiClipart, n.d.)

Previously you wrote a Java program to read a mark for a test from the keyboard, and then calculate the percentage. In the following example, the teacher wants to calculate the percentage for many students – not just one. Instead of running the program for each student, we want the program to ask the teacher whether there is another set of marks and if the answer is YES, then repeat the same calculation within the same program.

The flowchart for this program can look as follows:

Note: There are other ways to write this program – later in the module we will use other repetition structures to perform the same task.



The Java code for this program can look as follows:

```
import java.util.Scanner;
public class AllStudCalcPerc
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        char cAnswer;
        double rMark;           //The mark the student scored
        final int TESTTOTAL = 60; //The total marks for the test
        double rPerc;           //The percentage the learner obtained
```

```
        System.out.print("Do you want to enter a mark for a test? Y/N: ");
        cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
```

```
        while (cAnswer == 'Y')
```

```
        { //begin while loop
```

```
            System.out.print("Enter the mark out of " + TESTTOTAL + ": ");
```

```
            rMark = keyboard.nextDouble();
```

```
            rPerc = Math.round(rMark / TESTTOTAL * 100);
```

```
            System.out.println("The percentage is : " + rPerc + "%");
```

```
            System.out.print("Do you want to enter a mark for a test? Y/N: ");
```

```
            cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
```

```
        } //end while loop
```

```
        System.out.print("You are done calculating percentages."); //The next statement outside the while loop.
```

```
    } //end main method
```

```
} //end class
```

```
Do you want to enter a mark for a test? <Y/N>: y
Enter the mark out of 60: 55
The percentage is : 92.0%
Do you want to enter a mark for a test? <Y/N>: y
Enter the mark out of 60: 49
The percentage is : 82.0%
Do you want to enter a mark for a test? <Y/N>: n
You are done calculating percentages.
```

If the variable `cAnswer` contains the value 'Y', the condition will be true, and the statements inside the loop will be executed (the flow of the program will enter the loop). Otherwise, the flow of the program will branch to the next statement outside the while loop, effectively "jumping" the loop.



### *Note the following*

- The values of some variables will be changed in the loop.
  - Each time the loop executes, the value of the variables rMark, rPerc and cAnswer will be overwritten.
  - Therefore, once you entered the mark for the second student, those of the first student will be overwritten, and you will not be able to retrieve the percentage of the first student, you will just see it printed on the screen.
- The prompt to ask the user whether another mark should be entered appears twice in the program (shaded in grey in the example).
  - Once before the loop.
  - Once before the end of the loop.
- When the value of the Boolean expression in a while loop is false the first time the expression is evaluated, the loop will not start, and the code in the body of the loop may not execute.

### Activity 1: Repeat the same calculations

Create the program in the previous explanation. Add code to the program to do the following [AllStudCalcPercWithCounter]:

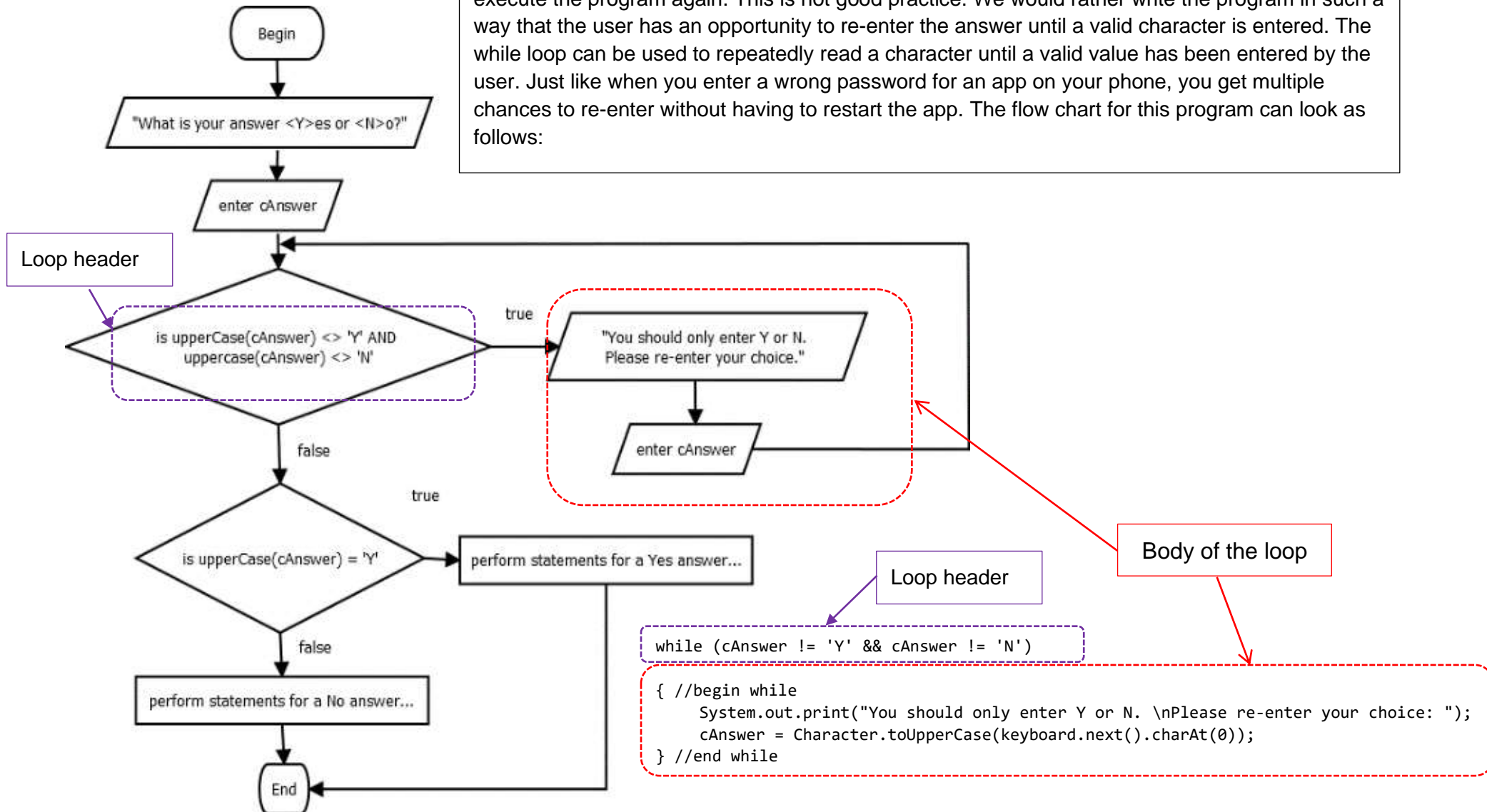
- Let the user enter the total for the test (do not use a constant (final) value).
- Add a counter so that the program displays the number of calculations that were performed.

```
What is the total for the test?: 60
Do you want to enter a mark for a test? <Y>/<N> (a mark is out of 60) y
What is the score? 55
The percentage is : 92.0%
Do you want to enter another mark for a test out of? Y/N (a mark is out of 60) y
What is the score? 34
The percentage is : 57.0%
Do you want to enter another mark for a test out of? Y/N (a mark is out of 60) y
What is the score? 25
The percentage is : 42.0%
Do you want to enter another mark for a test out of? Y/N (a mark is out of 60) n
You are done calculating percentages.
You processed 3 marks
```

There are various ways in which while loops can be used in any programming language. We are further going to demonstrate the way the while loop works by providing examples for different scenarios you may encounter in programming. Also note that for many of the scenarios we will use, the same results can be achieved using a do .. while, or a for loop. We will indicate how to rewrite certain solutions using a different loop when explaining the do .. while and for loops.

### 5.1.2 Preventing errors

In Unit 4 we wrote a program where a user had to answer a question by entering 'Y' or 'N'. We used a nested if .. then statement to display an error message when a user entered any value that was not 'Y' or 'N'. If an invalid character was entered, the program stopped, and the user had to execute the program again. This is not good practice. We would rather write the program in such a way that the user has an opportunity to re-enter the answer until a valid character is entered. The while loop can be used to repeatedly read a character until a valid value has been entered by the user. Just like when you enter a wrong password for an app on your phone, you get multiple chances to re-enter without having to restart the app. The flow chart for this program can look as follows:



The Java code for this scenario can look as follows:

```
import java.util.Scanner;
public class WhileYN
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        char cAnswer;
        System.out.print("What is your answer <Y>es or <N>o?: ");
        cAnswer = Character.toUpperCase(keyboard.next().charAt(0));

        while (cAnswer != 'Y' && cAnswer != 'N')
        { //begin while
            System.out.print("You should only enter Y or N. \nPlease re-enter your choice: ");
            cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
        } //end while

        if (cAnswer == 'Y')
            System.out.println("Your answer is Yes");
        else
            System.out.println("Your answer is No");

    } //end main method
} //end class
```

Note the different symbols to indicate 'not equal'.

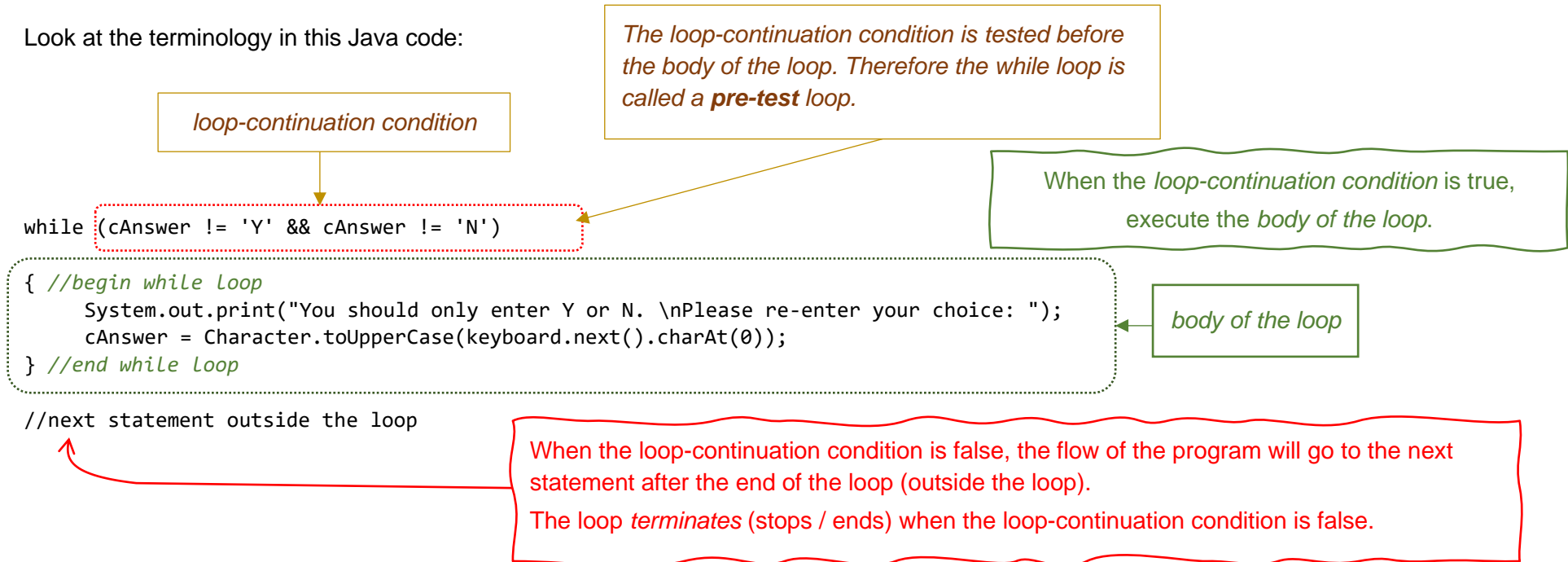
Mathematics	Pseudo code and flow charts	Java code
$\neq$	$\langle \rangle$	<code>!=</code>

The loop.

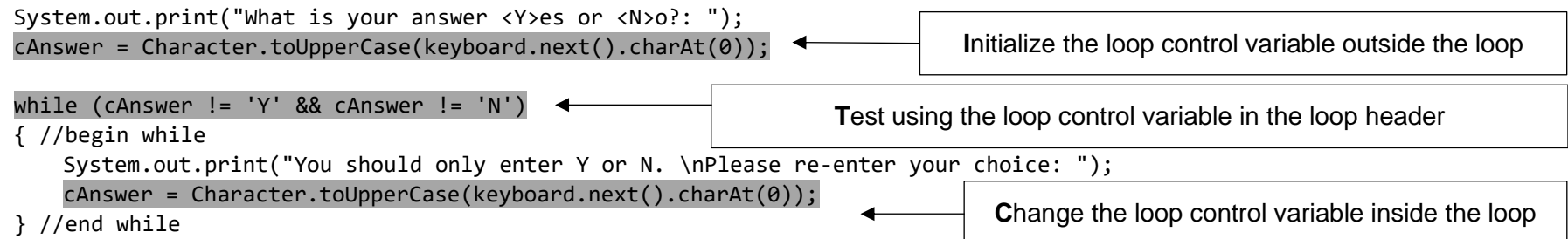
```
What is your answer <Y>es or <N>o?: t
You should only enter Y or N.
Please re-enter your choice: u
You should only enter Y or N.
Please re-enter your choice: y
Your answer is Yes
```

## Terminology relating to loops

Look at the terminology in this Java code:



- To write an efficient while-loop, you have to review your code and ensure that you applied the Initialize-Test-Change (ITC) principle. This means before the loop starts, you have to assign an initial value to the variable used in the loop-continuation-condition, then you test the condition, then the value of the variable has to change in the body of the loop.



- Pseudo code for a while loop can be very similar to the Java code itself. For example:

```
display "What is your answer <Y>es or <N>o?: "
enter cAnswer //as an uppercase value

while (cAnswer <> 'Y' AND cAnswer <> 'N')
  begin // while
    display ("You should only enter Y or N. Please re-enter your choice: ");
    enter cAnswer //as an uppercase value
  end // while
```

### Important

- To be able to continue with the rest of the program, the loop-continuation-condition must eventually become false so that the loop can terminate. Therefore, the value(s) used in the condition (the loop control variable(s)) must change somewhere in the loop. In the previous example, the variable cAnswer controls the loop. The user will enter a new value for cAnswer. So, when the user enters either 'Y' or 'N', the condition will become false, and the loop can end.
- Do not type a semi-colon (;) at the end of the loop-continuation-condition. A semi-colon indicates the end of a statement. Look at the following example:

```
while (cAnswer != 'Y' && cAnswer != 'N') ; // You are telling the computer to do nothing!
```



When the program reaches the semi-colon, it indicates the end of the statement, and program flow will loop back to testing the loop-continuation-condition. Since the user did not have the opportunity to enter a different character, the value of cAnswer will be the same. The user will never get the chance to enter a different character, and the loop will continue forever. This is called an *infinite loop*.

### A loop may be controlled by more than one loop control variables

In the following example two variables are used in the loop continuation condition.

The program is used in a Pizza shop. A pizza base costs R40.00. The cashier must indicate whether a customer wants a topping, and then has to enter the cost of the topping requested by the customer. The program prevents the cashier from adding more than 3 toppings.

- Infinite
- Terminate

To terminate an infinite loop, you can press <ctrl> + <c> at the same time to stop the program.

```

import java.util.Scanner;
import java.text.DecimalFormat;
public class PizzaTopsTwoVariables
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        DecimalFormat formatter = new DecimalFormat("R#,###.00");
        int iNumTops = 0;
        char cAnswer;
        double rTotal = 40.0, rCostTop;

        System.out.print("Does the customer want to add a topping <Y> / <N> ? ");
        cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
        while (cAnswer == 'Y' && iNumTops < 3)
        {
            iNumTops++;
            System.out.print("Cost of the topping the customer wants: R");
            rCostTop = keyboard.nextDouble();
            rTotal += rCostTop;
            System.out.print("Does the customer want to add a topping <Y> / <N> ? ");
            cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
        }

        if (iNumTops == 3 && cAnswer == 'Y')
            System.out.println("Remind the customer that the maximum number of toppings is 3.");

        System.out.println("Total cost is " + formatter.format(rTotal));

    } //end main method
} //end class

```

Two variables are used in the loop continuation condition:  
cAnswer and iNumTops.

## Activity 2: Use a while loop to prevent errors

a) The following program was used in an example in Unit 4.

This program is used by a courier company to calculate the cost of delivering a parcel. The cost is determined as follows:

- The first part of the cost (basic cost) is calculated as R5.50 per kg + transport cost per kilometre.
  - The transport cost is based on the mode of transport.
    - per road: 80¢ per km.
    - per train: 50¢ per km
    - by air: R1.50 per km
- If the customer wants to insure the parcel, the cost is increased by 11%.
- VAT of 15% must be added to the cost (after insurance has been added).

Use the following code, then adapt it as indicated in the instructions below the program.

```
package couriercost;
import java.util.Scanner;
import java.text.DecimalFormat;
public class CourierCost {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        DecimalFormat formatter = new DecimalFormat("R,##0.00");

        final int VAT = 15, INSURE_PERC = 11;
        final double PERKG = 5.5, ROAD = 0.8, TRAIN = 0.5, AIR = 1.5;
        double rMass, rKM, rTotal, rTransp;
        char cTransp, cInsure;

        //input
        System.out.print("Number of kg to transport: \t\t\t");
        rMass = keyboard.nextDouble();
        System.out.print("Number of km : \t\t\t\t\t");
        rKM = keyboard.nextDouble();
        System.out.print("Is transport by <R>oad, <T>rain or <A>ir \t");
        cTransp = keyboard.next().charAt(0);
        System.out.print("Insurance <Y>es or <N>o \t\t\t");
        cInsure = keyboard.next().charAt(0);
```

```

//set the transport cost based on the mode of travel
switch (cTransp)
{
    case 'R': rTransp = ROAD; break;
    case 'T': rTransp = TRAIN; break;
    case 'A': rTransp = AIR; break;
    default: rTransp = 0;
}
//calculate the cost based on the mass and the distance
rTotal = rMass * PERKG + rTransp * rKM;
//add insurance cost if chosen
if (cInsure == 'Y')
{
    rTotal = rTotal + INSURE_PERC/100.0 * rTotal;
    System.out.println("Adding insurance");
}
//Add VAT to the total
rTotal = rTotal + VAT/100 * rTotal;
//output
System.out.println("You need to pay: \t\t\t" + formatter.format(rTotal));
}
}

```

Adapt the program to meet the following requirements [CourierCostWhile]:

- A user must be able to enter lowercase characters such as r, y, n – the program should convert them to upper case.
- The program should ensure that a user enters one of the letters 'R', 'T', or 'A' for the transport mode. If another character is entered, an error message should be displayed, and the user must get a chance to re-enter a character.
- The program should ensure that a user enters only a 'Y' or 'N' when asked whether insurance should be added. If another character is entered, an error message should be displayed, and the user must get a chance to re-enter a character.

b) Create a Java program [Cashier] to do the following:

- Read the amount for all items a customer bought from the keyboard.
- Add VAT of 15% to the amount.
- Display the amount of money the customer should pay (amount due).
- Repeatedly read the amount the customer handed over to the cashier until the amount paid is bigger than the amount due.
- A message should be displayed when the amount of money handed over is not enough.
- Calculate and display the amount of change the customer should receive.



```
Amount for all items the customer bought: R365.75
Customer should pay R420.61
Amount customer handed to cashier: R400
It is not enough money - enter the amount again.
Amount customer handed to cashier: R420
It is not enough money - enter the amount again.
Amount customer handed to cashier: R430
Customer must receive R9.39 change.
```

### Activity 3: While loop terminologies

Write down the term for each of the following descriptions:

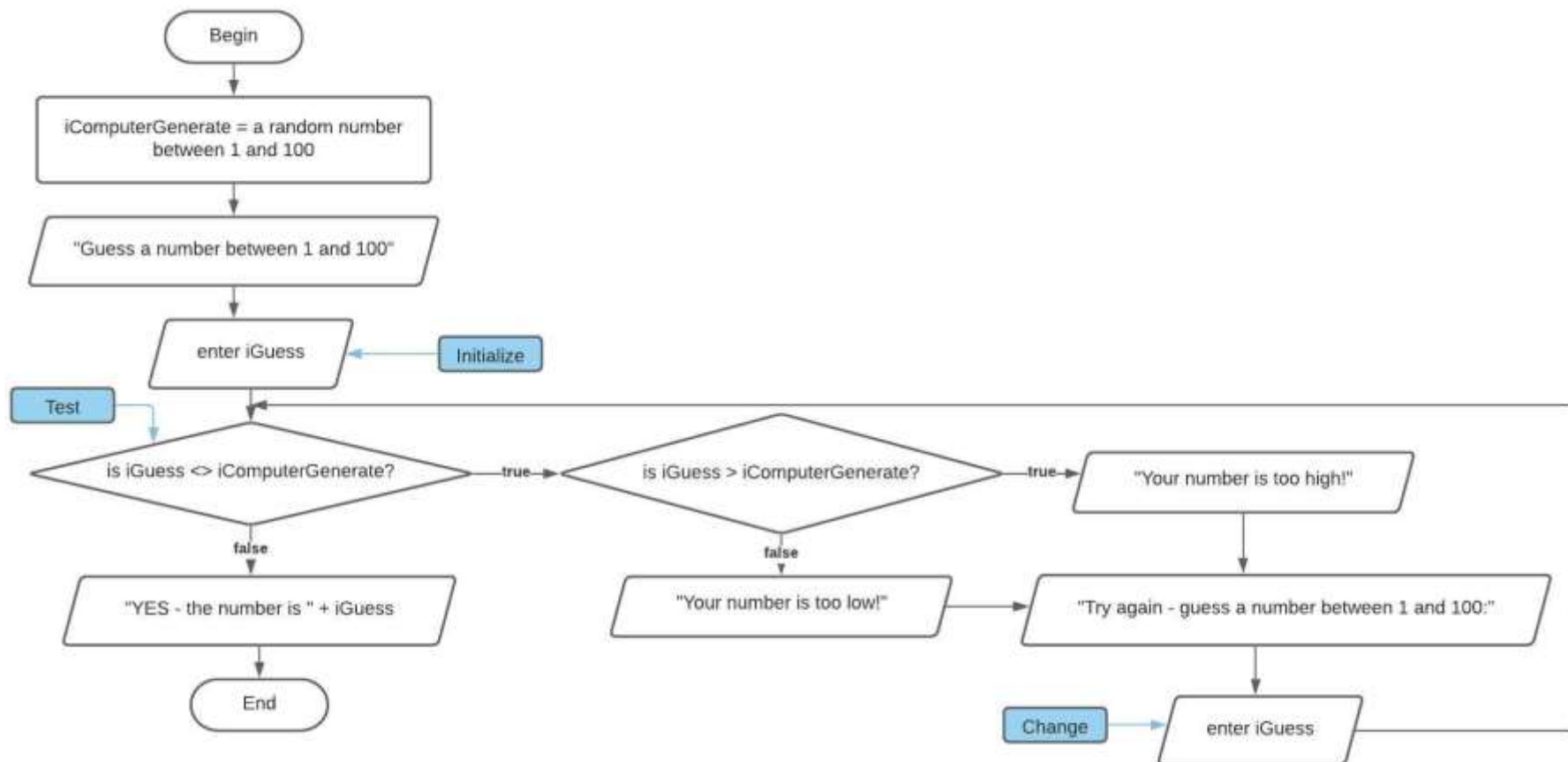
a)	The variable(s) used in the condition of a loop to determine whether the statements in the body of the loop should be executed.	
b)	The statements that will be executed when the loop continuation condition is true.	
c)	A loop that will not terminate because the loop continuation condition never becomes false.	
d)	The principle that should be applied to ensure that a while loop will execute and terminate correctly.	
e)	The combination of characters you need to press while running a program where a loop cannot terminate.	

### 5.1.3 Creating simple games

#### Guess the number

We can write a small game where the computer generates a random number between 1 and 100. The user must then guess which number the computer generated. Every time the user enters a number, the computer should tell the user if the number is too high, too low, or if it is the correct number. The user will therefore repeatedly provide an answer until the correct number is guessed - so we need a loop.

The flowchart can look as follows:



The code for the game can look as follows:

```
import java.util.Scanner;
public class GuessNumberGame
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iGuess, iComputerGenerate;
        iComputerGenerate = (int)(Math.random() * 100) + 1;
```

Remember the method random() of the class Math generates a value greater or equal to 0, and less than 1.

```
        System.out.print("Guess a number between 1 and 100: ");
        iGuess = keyboard.nextInt();
```

Initialize the loop control variable

```
        while (iGuess != iComputerGenerate)
        { //begin while
            if (iGuess > iComputerGenerate)
                System.out.println("Your number is too high!");
            else
                System.out.println("Your number is too low!");
```

Test the loop control variable

You can use any Java structure in the body of the loop. In this case an if .. else statement was needed.

```
            System.out.print("Try again - guess a number between 1 and 100: ");
            iGuess = keyboard.nextInt();
```

Change the loop control variable

```
        } //end while
```

```
        System.out.print("YES - the number is " + iGuess);
```

```
    } //end main method
} //end class
```

The number guessing game can be adapted so that it will count how many times the user guessed before getting the correct number. The output can look like this

```
Guess a number between 1 and 100: 50
Your number is too high!
Try again - guess a number between 1 and 100: 35
Your number is too high!
Try again - guess a number between 1 and 100: 20
Your number is too high!
Try again - guess a number between 1 and 100: 10
Your number is too low!
Try again - guess a number between 1 and 100: 15
Your number is too high!
Try again - guess a number between 1 and 100: 12
Your number is too high!
Try again - guess a number between 1 and 100: 11
YES - the number is 11
You guessed 6 times before getting it right!
```

An additional variable will be needed to keep track of the number of guesses. This variable has to be initialized to 0 (zero) and increased by one every time the user's guess is wrong. This is called a *counter*. It counts the number of times an action was repeated. The previous Java program can be adapted like this:

```
package guessnumbergamecount;
import java.util.Scanner;
public class GuessNumberGameCount {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int iGuess, iComputerGenerate;
        int iCountGuesses = 0;
        iComputerGenerate = (int)(Math.random() * 100) + 1;
        System.out.print("Guess a number between 1 and 100: ");
        iGuess = keyboard.nextInt();
        while (iGuess != iComputerGenerate)
        { //begin while

            if (iGuess > iComputerGenerate)
                System.out.println("Your number is too high!");
            else iCountGuesses ++;
                System.out.println("Your number is too low!");
            System.out.print("Try again - guess a number between 1 and 100: ");
            iGuess = keyboard.nextInt();
        } //end while
        System.out.println("YES - the number is " + iGuess);
        System.out.println("You guessed " + iCountGuesses + " times before getting it right!");
    }
}
```

Set the counter to zero **before (outside)** the loop.

Increase the counter **inside** the loop.

Report the final value of the counter **after (outside)** the loop.

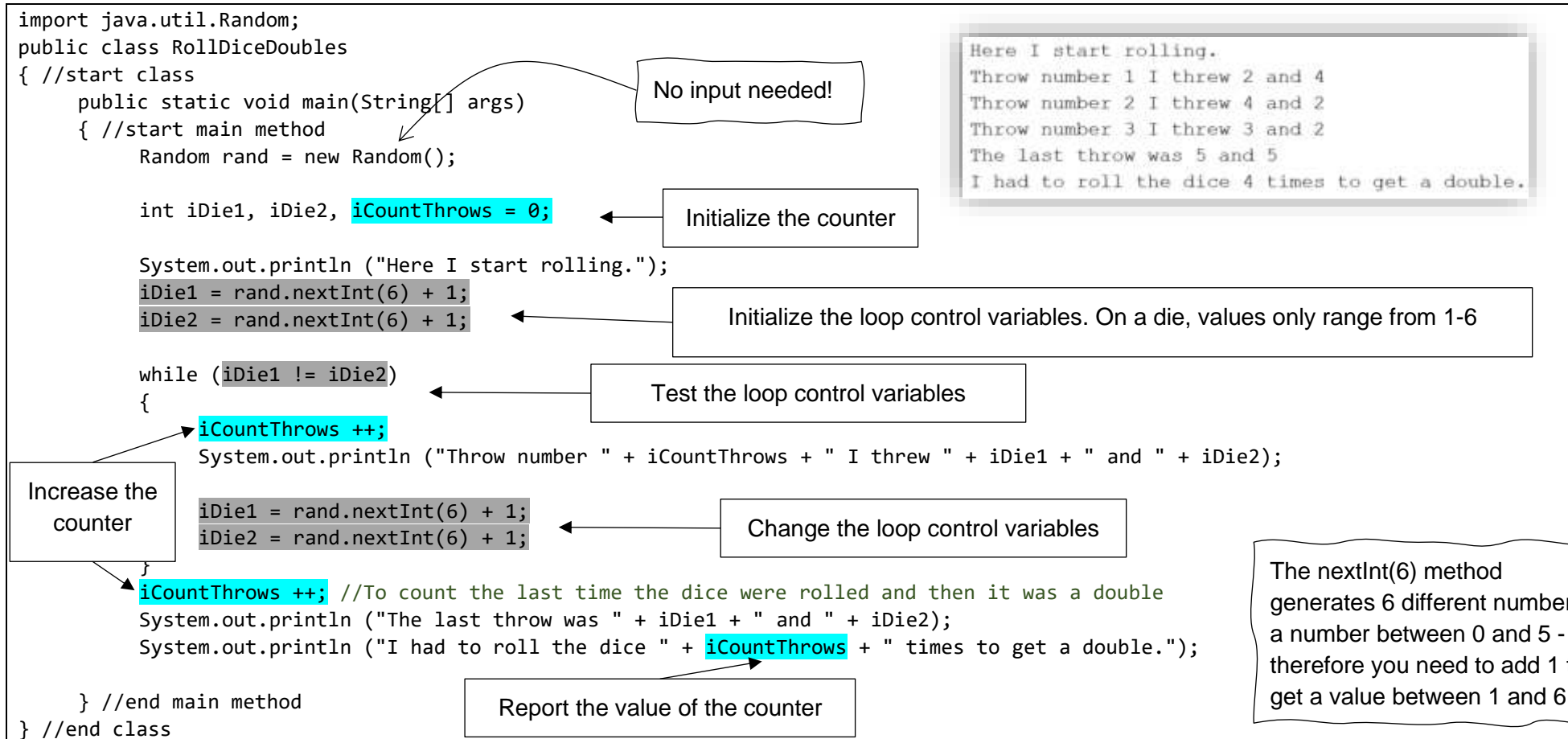
## Activity 4: Flow chart for guessing game

Draw a flow chart for the guessing game where the number of times the user guessed wrong is reported.

### Throw a double dice value

Many of the games generate random numbers to simulate playing a certain card or rolling a die. In the next program, the computer rolls two dice, and counts how many times the dice had to be rolled before a double was played (both dice had the same value).

- Die (*noun - singular*)
- Dice (*noun – multiple*)



## Activity 5: Small games

- a) Write a Java program [RollDiceSum7] to create a game where the computer generates two random numbers – this simulates the role of two dice. The program should 'role' the dice until the sum of the dice is 7. For each attempt, the program must report which numbers were thrown. When the sum of the dice is 7, the program should also report how many times the dice had to be rolled. Look at the example of output. (Activity adapted from (Reges & Stepp, 2011, p. 335))
- b) Write a Java program [RollDiceDouble6] to create a game where the computer generates two random numbers – this simulates the role of two dice. The program should 'role' the dice until both dice are a number 6. Look at the example output to design your program.

```
Here I start rolling.  
4 + 6 = 10  
2 + 4 = 6  
3 + 4 = 7  
All done - I had to roll the dice 3 times.
```

```
Here I start rolling.  
I threw 6 and 2  
I threw 6 and 3  
I threw 2 and 1  
I threw 1 and 6  
I threw 3 and 4  
I threw 2 and 4  
The last throw was 6 and 6  
I had to roll the dice 7 times to get two sixes.
```

### Summary

- While loops are controlled by a *loop-continuation condition*. If the result of the loop-continuation condition is true, the body of the loop is executed. If the result of the loop-continuation condition is false, the flow of the program branches to the first statement following the **end of the while loop**.
  - One or more variables may be used in the loop-continuation condition. These variable(s) are called the *loop control variable(s)*.
  - The loop control variable(s) are initialized before the start of the loop (outside the loop), they are tested in the loop-continuation condition, and changed in the body of the loop. This is referred to as the *ITC principle*.
  - The loop-continuation condition is tested *before* any of the statements in the body of the loop is executed. Therefore, the while loop is called a *pre-test* loop.
  - The result of the loop-continuation condition may be false and this will prevent the loop (or the repetition) from starting at all. . It is therefore possible that the body of a while loop may never be executed.
  - In the examples we reviewed so far, the result of the loop-continuation condition controlled the execution of the loop. This type of loop control is called *result control*.
- Pre-
  - Loop-continuation condition
  - Loop control variable(s)
  - ITC principle
  - Pre-test
  - Result controlled

## 5.1.4 Letting the user control the number of repetitions

### Specifying the number of times a loop should repeat

In the first example of a scenario where a while loop can be used, we asked the user to enter an answer (Y/N) to indicate whether a calculation should be repeated. We can also ask the user to enter the number of times a calculation should be repeated. The code will then look as follows:

```
import java.util.Scanner;
public class HowManyStudCalcPerc
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        double rScore;           //The mark the student scored
        final int TESTTOTAL = 60; //The total marks for the test
        double rPerc;            //The percentage the learner obtained
        int iNumStudents;        //Ask the user how many marks will be entered
        int iCounter = 0;
        System.out.print("How many students are there?: ");
        iNumStudents = keyboard.nextInt();

        while (iCounter < iNumStudents)
        { //begin while
            iCounter++;
            System.out.print("What is the score out of " + TESTTOTAL + " for student " + iCounter + "? ");
            rScore= keyboard.nextDouble();
            rPerc = Math.round(rScore / TESTTOTAL * 100);
            System.out.println("The percentage is for student " + iCounter + " is " + rPerc + "%");
        } //end while

        System.out.print("You are done calculating percentages.");

    } //end main method
} //end class
```

```
How many students are there?: 3
What is the score out of 60 for student 1? 45
The percentage is for student 1 is 75.0%
What is the score out of 60 for student 2? 38
The percentage is for student 2 is 63.0%
What is the score out of 60 for student 3? 55
The percentage is for student 3 is 92.0%
You are done calculating percentages.
```

While the counter has not reached the number of students yet, the loop should continue.

You can also say: while the counter is still smaller than the number of students, the loop should continue.

## Display five greetings

A while loop can also be used to execute certain statements a specific number of times. For example, let us write a program to display 5 greetings on the screen.

```
This is greeting number 1
This is greeting number 2
This is greeting number 3
This is greeting number 4
This is greeting number 5
Outside the loop the value of the counter is 6
Bye!
```

```
public class FiveGreetings
```

```
{ //start class
```

```
    public static void main(String[] args)
```

```
    { //start main method
```

```
        int iCounter = 1 ;
```

```
        while (iCounter <= 5)
```

```
        {
```

```
            System.out.println ("This is greeting number " + iCounter);
```

```
            iCounter ++;
```

```
        }
```

```
        System.out.println ("Outside the loop the value of the counter is " + iCounter);
```

```
        System.out.println ("Bye!");
```

```
    } //end main method
```

```
} //end class
```

No input needed!

Initialize the counter that controls the loop.

Test the counter that controls the loop. If the counter is 5 or less, the body of the loop will execute.

Use the value of the counter in the output.

Change the counter that controls the loop.

Notice that the value of counter will be 6 outside of the loop.



## Activity 6: Repeat the same calculations a set number of times

- a) A Java program [CalcTSA] is needed to do the following:
- The total surface area (TSA) of objects must be calculated. Each object consists of a number of rectangles. The program must:

- Read the number of rectangles an object has.
- For each rectangle, read the length and width of the rectangle.
- Calculate and display the TSA of the object.

```
How many rectangle surfaces does the object have?: 3
Enter length of rectangle 1: 5
Enter width of rectangle 1: 3
Enter length of rectangle 2: 5
Enter width of rectangle 2: 2
Enter length of rectangle 3: 5
Enter width of rectangle 3: 3
Total surface area is: 40.0
```

- b) You need to write a program to test whether a learner can do subtraction calculations correctly [SubtractQuiz]. The program should:
- Generate 4 questions. Each question should use two single-digit numbers randomly generated. This is an example of a question:

```
This is question number 1
What is 4 - 6?
```

- The learner must enter the answer to the question.
- The program should test whether the learner's answer is correct or not. A suitable message must be displayed in each case.
- After all 4 questions have been answered, the program must display the number of correct answers.

The program must make use of a while loop.

This is an example of the output of the program.

```
This is question number 1
What is 3 - 0? 3
Correct!
This is question number 2
What is 5 - 8? 3
No! 5 - 8 = -3
This is question number 3
What is 6 - 9? -3
Correct!
This is question number 4
What is 2 - 7? 5
No! 2 - 7 = -5
You had 2 correct answers.
```

## Entering a sentinel value

Another way to let the user control the number of times a loop should execute, is to ask the user to enter a *sentinel value*.

"In programming, a sentinel value is a special value that is used to terminate a loop. The sentinel value typically is chosen so as to not be a legitimate data value that the loop will encounter and attempt to work with. Also referred to as a *signal value*" (webopedia, 2020).

We can use the example of the program where the total amount of money donated had to be determined to demonstrate the use of a sentinel value.

```
import java.util.Scanner;
public class DonationSentinel
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);

        double rTotal = 0; //The total amount of money that was collected
        double rDonate;    //The amount of money one person donates
        int iCounter = 0 ; //Count the number of donations received

        System.out.print("Enter the donation amount. Enter 0 to stop: ");
        rDonate= keyboard.nextDouble();

        while (rDonate != 0)
        { //begin while
            iCounter++;
            rTotal= rTotal + rDonate;
            System.out.print("Enter the donation amount. Enter 0 to stop: ");
            rDonate= keyboard.nextDouble();
        } //end while

        System.out.println("You collected R" + rTotal);
        System.out.println("There were "+ iCounter + " donations");

    } //end main method
} //end class
```

A value of R0 is not a value you want to use in calculating the total money donated. So it is not a legitimate data value. Therefore, it can be used as a sentinel value (a signal to indicate that the end of all the data values was reached).

You could also use a negative value as a sentinel in this case.

## Activity 7: Sentinel controlled repetitions

A number of people joined a weight loss programme. Their weight was recorded when they joined, and again 6 months later. Write a Java program [WeightLoss] to do the following (adapted from (Erasmus & Pretorius, 2012)):

- Read the name and surname, initial weight, and end weight for a person.
- Determine the difference in weight and display a message to indicate if the person gained weight, lost weight, or stayed the same.
- The program should repeatedly read data for a person, until the name 'zzz' is entered.
- At the end of the program, the number of people who gained, lost and stayed the same should be displayed.

Look at the example output to plan your program.

- Remember how to ignore the case of letters in a string when comparing two strings. Unit 4, Topic 4.16.1

```
sUsernameEntered.equalsIgnoreCase(sUsernameStored)
```

- Also remember the problem with the `nextLine()` method when a string is read after a number was entered (discussed in Unit 3, Topic 3.10)

```
Enter name (zzz to stop): Jennifer Mabena

Start weight for Jennifer Mabena: 65
End weight for Jennifer Mabena: 60
Jennifer Mabena, you lost 5.0 kg.
Enter name (zzz to stop): Peter Mogolola

Start weight for Peter Mogolola: 88
End weight for Peter Mogolola: 92
Peter Mogolola, you gained 4.0 kg.
Enter name (zzz to stop): Jennifer Smith

Start weight for Jennifer Smith: 77
End weight for Jennifer Smith: 77
Jennifer Smith, you lost no weight.
Enter name (zzz to stop): zzz
1 lost weight.
1 gained weight.
1 stayed the same.
```

### 5.1.5 Calculating cumulative values

In the following program the user of the program wants to keep track of the total amount of money collected through donations. Every time a person gives the user some money, he enters the amount. The user wants the program to display the total amount of money that was collected, as well as the number of donations received.

```
import java.util.Scanner;
public class WhileAnotherDonation
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        char cAnswer;
1.      double rTotal = 0; //The total amount of money that was collected
        double rDonate; //The amount of money one person donates
2.      int iCounter = 0 ; //Count the number of donations received
3.      System.out.print("Do you have a donation? Y/N ");
4.      cAnswer = Character.toUpperCase(keyboard.next().charAt(0));

5.      while (cAnswer == 'Y') ←
        { //begin while
6.          iCounter++; //Increase the counter by 1
7.          System.out.print("How much money did the person donate? ");
8.          rDonate= keyboard.nextDouble();
9.          rTotal= rTotal + rDonate; //Increase the total by the amount of money (stored in variable rDonate)
10.         System.out.print("Do you have another donation? Y/N ");
11.         cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
        } //end while

12.     System.out.println("You collected R" + rTotal);
13.     System.out.println("There were "+ iCounter + " donations");
    } //end main method
} //end class
```

Some of the lines in the code are numbered. The use of a trace table will be explained next, and the trace table will include the number of the steps executed.

If the variable cAnswer contains the value 'Y', the condition will be true, and the statements inside the loop will be executed (lines 6 – 11). Otherwise, the flow of the program will branch to the first statement outside the while loop. (line 12).

```
Do you have a donation? Y/N y
How much money did the person donate? 300
Do you have another donation? Y/N y
How much money did the person donate? 250
Do you have another donation? Y/N y
How much money did the person donate? 100
Do you have another donation? Y/N n
You collected R650.0
There were 3 donations
```

### Using a trace table with a while loop

It is very important that you use a trace table to verify the outcome of a program when the program includes the use of loops. The following is a trace table for the class WhileAnotherDonation. The input values are the same as those in the screenshot. Work through the table while referring to the code so you can see how the program flows.

Step	Instruction	rTotal	rDonate	iCounter	cAnswer	Outcome of while loop continuation condition	Output
1.	initialize	0					
2.	initialize			0			
3.	output/display						Do you have a donation? Y/N
4.	input / enter				Y		
5.	while evaluation					true	
6.	calculation			1			
7.	output / display						How much money did the person donate?
8.	input / enter		300				
9.	calculation	300					
10.	output / display						Do you have another donation? Y/N
11.	input / enter				Y		
5.	while evaluation					true	
6.	calculation			2			
7.	output / display						How much money did the person donate?
8.	input / enter		250				

Step	Instruction	rTotal	rDonate	iCounter	cAnswer	Outcome of while loop continuation condition	Output
9.	calculation	550					
10.	output / display						Do you have another donation? Y/N
11.	input / enter				Y		
5.	while evaluation					true	
6.	calculation			3			
7.	output / display						How much money did the person donate?
8.	input / enter		100				
9.	calculation	650					
10.	output / display						Do you have another donation? Y/N
11.	input / enter				N		
5.	while evaluation					false	
12.	output / display						You collected R650.0
13	output / display						There were 3 donations

## Activity 8: Calculate cumulative values

- a) Write a program that will read the percentages of a number of students [CalcAverage]. The program should ask the user how many values will be entered. The program should calculate the average of all the percentages (scores) entered and display it as an integer value (round the average).
- b) Create a complete trace table to demonstrate the execution of statements and the output produced. Use the value 3 for the number of learners.

```
How many learners? 4
Enter percentage for learner 1: 55.8
Enter percentage for learner 2: 58.5
Enter percentage for learner 3: 88.6
Enter percentage for learner 4: 75.6
Average is 70%
```

```
How many learners? 3
Enter percentage for learner 1: 45.8
Enter percentage for learner 2: 66.5
Enter percentage for learner 3: 87.4
```

### 5.1.6 *Finding the highest value from a list of values*

In programming you often have to determine what the highest value is from a list of values. In the next example a teacher will enter the marks of all the learners in the class. When the last mark has been entered, the program must report which mark was the highest. The Java program will look as follows:

```
import java.util.Scanner;
public class FindHighest
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);

        double rHighest = 0;
        double rPerc;

        System.out.print("Enter the next mark. (Enter 0 to indicate all marks have been entered): ");
        rPerc = keyboard.nextDouble();

        while (rPerc != 0)
        { //begin while
            if (rPerc > rHighest)
                rHighest = rPerc;
            System.out.print("Enter the next mark. (Enter 0 to indicate all marks have been entered): ");
            rPerc = keyboard.nextDouble();
        } //end while

        System.out.println("The highest percentage was: " + rHighest);

    } //end main method
} //end class
```

Since all the values that will be entered will be greater than 0, the variable rHighest is initialized to 0 – the smallest value that may be entered).

If the value that was just entered is bigger than the highest value found so far, then let this value become the highest value. (The first time a value is entered, rHighest will be 0, so the first value will become the highest value.)



### Activity 9: Find the lowest value

Adjust the program in the example where the highest value in a list of values was found, so that it finds the lowest value from the list [FindLowest].

```
Enter the student mark. (Enter a number that is larger than 100 to indicate all marks have been entered): 88
Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 45
Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 68
Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 72
Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 999
The Lowest percentage is: 45.0
```

### Activity 10: Display the name with the lowest mark

Adjust the program in the previous activity so that the name of the learner is read with every mark. At the end of the program, the name of the student who scored the lowest mark should also be displayed [LowestAndName].

```
Enter the student mark. (Enter a number that is greater than 100 to indicate all marks have been entered): 88
Enter the name of the student who scored 88.0: Katlego

Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 75
Enter the name of the student who scored 75.0: Jennifer

Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 58
Enter the name of the student who scored 58.0: Tshepo

Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 84
Enter the name of the student who scored 84.0: James

Enter the student mark. (Enter a number that is more than 100 to indicate all marks have been entered): 999
TSHEPO obtained the lowest percentage!
The Lowest percentage is: 58.0
```

### 5.1.7 *Creating a menu*

A while loop can be used to create a menu option where the user can repeatedly choose between options. Look at the following screenshot of a program. In this program the user can choose to add different toppings to a pizza. More than one topping can be added, so we will use a while loop to repeatedly display the different toppings and allow a user to add a topping, until the user indicates that no more toppings will be needed.

```

import java.util.Scanner;
import java.text.DecimalFormat;
public class MenuPizzaTop
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        DecimalFormat formatter = new DecimalFormat ("R,##0.00");
        double rTotal = 0; //The total amount the customer should pay.
        int iOption; //The menu option the user chooses.
        final double MUSHROOM = 8.50;
        final double HAM = 16.00;
        final double CHEESE = 6.50;

        System.out.println("\nChoose an option. \n1. Mushroom \n2. Ham \n3. Cheese \n0. STOP ");
        iOption = keyboard.nextInt();
        while (iOption != 0)
        {
            switch (iOption)
            {
                case 1: rTotal= rTotal + MUSHROOM; break;
                case 2: rTotal= rTotal + HAM; break;
                case 3: rTotal= rTotal + CHEESE;
            } //end switch
            System.out.println("Your total so far is: " + formatter.format(rTotal));
            System.out.println("\nChoose an option. \n1. Mushroom \n2. Ham \n3. Cheese \n0. STOP");
            iOption = keyboard.nextInt();
        } //end while

        System.out.println("You should pay " + formatter.format(rTotal));
    } //end main method
}

```

The escape sequence `\n` in the `println()` method produce the menu on the screen. See Unit 3 Topic 3.21 to refresh your knowledge on the use of escape sequences.

```

Choose an option.
1. Mushroom
2. Ham
3. Cheese
0. STOP

```

## Activity 11: Improve Pizza toppings program

Adapt the pizza topping program that was provided as example to do the following [MenuPizzaTopImproved]:

- Display the price of each topping in the menu option (if the cost of a topping is changed, the menu must display the new values without you having to change the `println()` statement).

- b) Display the number of toppings for each type that was chosen by the user.

Look at the example output to help you to plan the program.

Notice the prices in the menu are all displayed below one another. *Hint:* Use the `\t` escape sequence.

```
Choose an option.
1. Mushroom      R8.50
2. Ham           R16.00
3. Cheese         R6.50
0. STOP
```

```
You ordered 2 mushroom topping(s)
You ordered 1 ham topping(s)
You ordered 0 cheese topping(s)
You should pay R33.00
```

## Activity 12: Use menus

- a) Write a Java program [HotelDiscount] where the original bill for staying at a hotel is entered. A discount is given to those who have done certain activities during their stay.
- The user should enter the number of activities they did.
  - The program should provide discount options based on the activities.
  - The user should be able to select all the activities they have done.
  - Finally, the program should calculate and display the total discount received and the bill after discount.

An example of the output is provided to guide you:

- b) Modify the program you wrote for activity 13 a) so that [HotelDiscountModified]:

```
How much is your original hotel bill: R5000

How many activities did you do?: 2

Activities      Discount
1. Zip-lining   10%
2. Horse riding 12.5%
3. Site seeing  0%
4. SpeedBoat    10%
Pick your activity No 1 < options 1 to 4>: 1
So far, your discount is: R500.00

Activities      Discount
1. Zip-lining   10%
2. Horse riding 12.5%
3. Site seeing  0%
4. SpeedBoat    10%
Pick your activity No 2 < options 1 to 4>: 4
So far, your discount is: R1,000.00

Your total discount is: R1,000.00
After discount, your bill will be R4,000.00
```

- The program will ask the user to re-enter the number of activities they did, if the number is less than 0 or more than 4.
- The list of activities and discount is only displayed once.

```

How many activities did you do?: 5
You should only enter an integer from 0 to 4
Please re-enter the number of activities you did: 2

Activities      Discount
1. Zip-lining   10%
2. Horse riding 12.5%
3. Site seeing  0%
4. SpeedBoat    10%
Pick your activity No 1 < options 1 to 4>: 5
You should only enter an integer from 0 to 4
Please re-enter your activity number: 2
So far, your discount is: R625.00
Pick your activity No 2 < options 1 to 4>: 4
So far, your discount is: R1,125.00

Your total discount is: R1,125.00
After discount, your bill will be R3,875.00

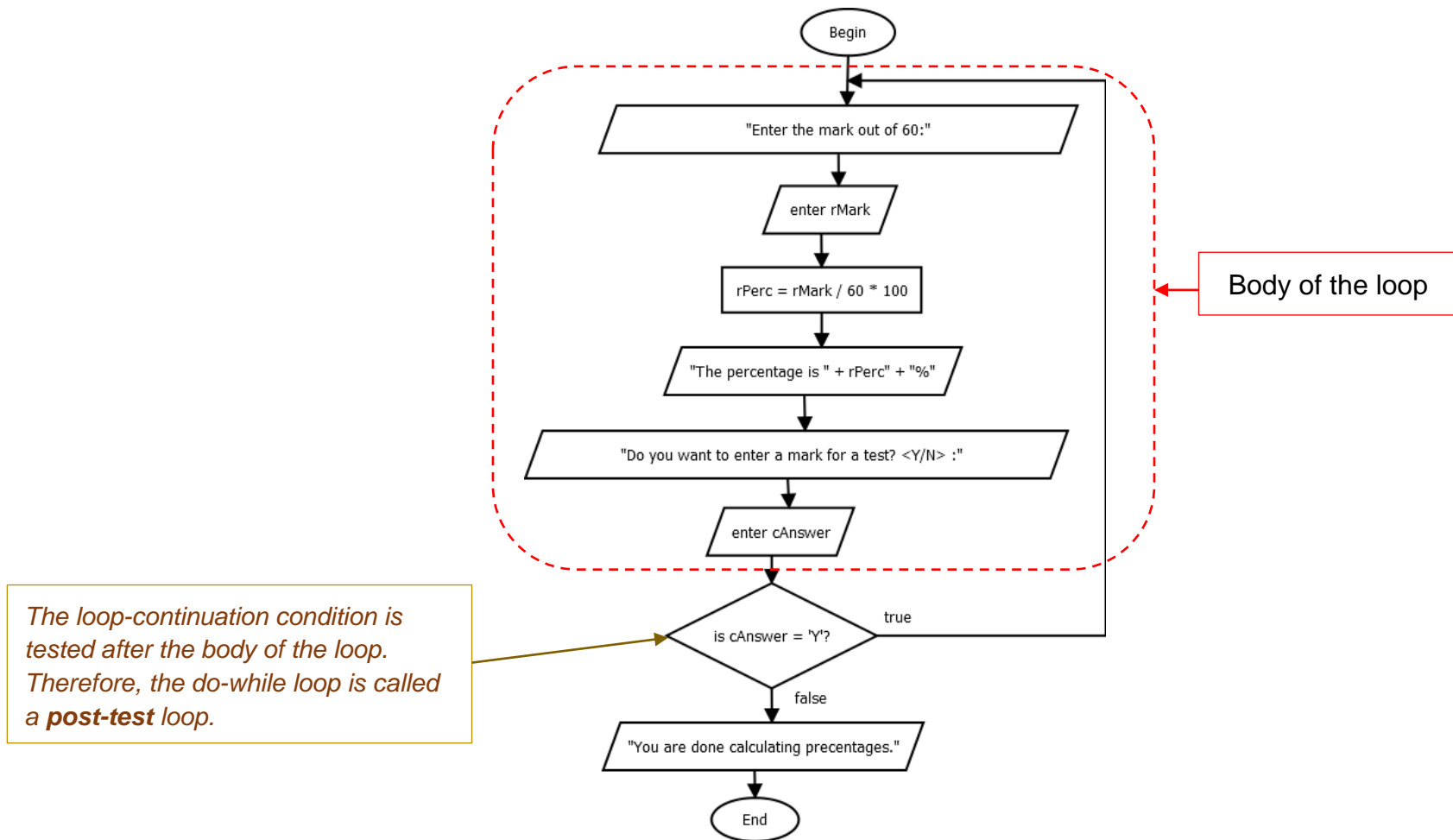
```

## 5.2 Using a do-while loop

### 5.2.1 *Basic structure of a do-while loop*

A do-while loop is another repetition structure that is available in Java. It works like a while loop, except that the condition is tested after the loop is executed. Therefore, the body of the do-while loop is always executed at least once.

The following flow chart and Java code is the same program as the first example in the while loop section – paragraph 5.1. You will see the first statement to be executed is the first line in the body of the loop.



```

import java.util.Scanner;
public class AllStudCalcPercDoWhile
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        char cAnswer;
        double rMark, rPerc;
        final int TESTTOTAL = 60;
        do
        { //begin loop
            System.out.print("Enter the mark out of " + TESTTOTAL + ": ");
            rMark= keyboard.nextDouble();
            rPerc = Math.round(rMark / TESTTOTAL * 100);
            System.out.println("The percentage is: " + rPerc + "%");
            System.out.print("Do you want to enter a mark for a test? Y/N ");
            cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
        } // end loop
        while (cAnswer == 'Y');

        System.out.println("You are done calculating percentages.");
    } //end main method
} //end class

```

The loop control variable is not initialized before the start of the do-while loop.

Test the loop control variable

Change the loop control variable

### Note the following

- Most programs where a while loop is used, can be re-written to use a do-while loop. The choice of loop to use is usually a programmer's personal preference.
- Be careful to place the end symbol } and the semi-colon ; in the correct places for the do-while loop. Look at the differences between the while and the do-while loops.

```

while (loop-continuation condition)
{
    //body of the loop
}

```

while loop

```

do
{
    //body of the loop
}
while (loop-continuation condition);

```

do-while loop

### 5.2.2 Comparison between a while and a do-while loop

Let us look at an example where a do-while loop is preferable to use.

A code consisting of three digits must be generated. A Java program should generate such a code, where all three digits must be in the range of 1 – 9, and none of the digits may be the same.

The solution using a do-while loop can be as follows:

```
package threedigits;
import java.util.Random;
public class ThreeDigits {
    public static void main(String[] args) {
        Random randomizer = new Random();
        int iDig1, iDig2, iDig3 ;
        do
        {
            iDig1 = randomizer.nextInt(9) + 1;
            iDig2 = randomizer.nextInt(9) + 1;
            iDig3 = randomizer.nextInt(9) + 1;
        }
        while ((iDig1 == iDig2) || (iDig1 == iDig3) || (iDig2 == iDig3));
        System.out.println("The three-digit code is: " + iDig1 + iDig2 + iDig3);
    }
}
```

Three variables are used in the loop-continuation condition. The digits are generated in the body of the loop only.

```
package threedigitswhile;
import java.util.Random;
public class ThreeDigitsWhile {
    public static void main(String[] args) {
        Random randomizer = new Random();
        int iDig1, iDig2, iDig3 ;
        iDig1 = randomizer.nextInt(9) + 1; iDig2 = randomizer.nextInt(9) + 1; iDig3 = randomizer.nextInt(9) + 1;
        while ((iDig1 == iDig2) || (iDig1 == iDig3) || (iDig2 == iDig3))
        {
            iDig1 = randomizer.nextInt(9) + 1; iDig2 = randomizer.nextInt(9) + 1; iDig3 = randomizer.nextInt(9) + 1;
        }
        System.out.println("The three-digit code is: " + iDig1 + iDig2 + iDig3);
    }
}
```

The solution using a while loop is not as elegant as using a do-while loop, since the statements to create the three digits have to be used twice. For the do-while loop solution the statements only appear once in the loop.

This scenario is an example where a programmer will most probably prefer to use a do-while loop instead of a while loop.

Here are some of the differences between a while and a do-while loop:

while	do-while
A pre-test loop	A post-test loop
Depending on the initial values of the loop control variable(s), the loop may terminate immediately – therefore it is possible that the statements in the body of the loop may never be executed.	The statements in the body of the loop will definitely be executed once.
The loop control variable(s) must be initialized before the loop header.	The loop control variable(s) are assigned values in the body of the loop only.

### Activity 13: Problems using a do-while loop

- a) Adapt the class RollDiceDoubles from Unit 5: Topic 5.1.3 (Throw a double dice value) to use a do-while loop instead of a while loop.

```
Here I start rolling.  
I threw 3 and 5  
I threw 5 and 4  
I threw 4 and 4  
I had to roll the dice 3 times to get a double.
```

- b) Create a Java program [Savings] that will read the following from the keyboard:
- Initial amount a person wants to invest, the target amount to be achieved, the interest rate per year.
  - The program should display the amount with interest after each year, until the amount is just larger than the target amount.

Use do-while loops to do the following data validation:

- The amount to be invested must be larger than 0.
- The target value must be larger than the amount to invest.
- The interest rate must be larger than 0.



Look at the example output to plan your program.

```
Initial amount to invest (a value > 0): R5000
Target you want to achieve (must be more than)5000.0: R8000
Interest rate (value greater than 0): 5
After year 1 you will have R5,250.00
After year 2 you will have R5,512.50
After year 3 you will have R5,788.12
After year 4 you will have R6,077.53
After year 5 you will have R6,381.41
After year 6 you will have R6,700.48
After year 7 you will have R7,035.50
After year 8 you will have R7,387.28
After year 9 you will have R7,756.64
After year 10 you will have R8,144.47
After 10 years you will have more than R8,000.00
```

- c) A ball is dropped from a specific height. The height must be provided in meters. Each time the ball bounces off the ground, it reaches a height half of the previous height. Write a Java program [DropBall] to calculate and display how many times it will bounce before the height is less than 5 centimetres. Look at the example output to plan your program. (Problem adapted from (Erasmus & Pretorius, 2012).)

```
Initial height of the ball in meters: 1.58
After bounce number 1 the height is 0.79
After bounce number 2 the height is 0.395
After bounce number 3 the height is 0.1975
After bounce number 4 the height is 0.09875
After bounce number 5 the height is 0.049375
After 5 bounces the height will be less than 5cm
```

- d) Write two Java programs, one implementing a while loop [WithdrawMoneyWhile] , the second one implementing a do-while loop [WithdrawMoneyDoWhile]. The program should simulate the withdrawal of funds from a bank account.

- Assume the balance for the account is R 5000.
- The user of the program should be allowed to make multiple withdrawals.
- If the withdrawal amount is bigger than the balance of the account, an error message should be displayed, and the user must have the opportunity to re-enter a withdrawal amount.
- When the user does not want to make any more withdrawals, the program should display how many valid withdrawals were made, how much money was

```
Enter the amount to withdraw: (0 to stop)200
The remaining balance is: R4,800.00
Enter the amount to withdraw: (0 to stop)500
The remaining balance is: R4,300.00
Enter the amount to withdraw: (0 to stop)0
Number of withdrawals: 2
Total amount withdrawn: R700.00
The remaining balance is: R4,300.00
```

withdrawn in total, and what the balance of the account is.  
See the examples of output to help you to plan the program.

```
Enter the amount to withdraw: (0 to stop)0
Number of withdrawals: 0
Total amount withdrawn: R0.00
The remaining balance is: R5,000.00
```

```
Enter the amount to withdraw: (0 to stop)500
The remaining balance is: R4,500.00
Enter the amount to withdraw: (0 to stop)6500
Insufficient funds. R6,500.00 cannot be withdrawn
Enter the amount to withdraw: (0 to stop)2000
The remaining balance is: R2,500.00
Enter the amount to withdraw: (0 to stop)0
Number of withdrawals: 2
Total amount withdrawn: R2,500.00
The remaining balance is: R2,500.00
```

## 5.3 Using for loops

### 5.3.1 Basic structure of a for loop

Another iteration control structure that can be used in Java to execute statements a specific number of times, is the *for* loop.

We used the following code to display five greeting on the screen using a while loop:

```
import java.util.Scanner;
public class FiveGreetings
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iCounter = 1 ;
        while (iCounter <= 5)
        {
            System.out.println ("This is greeting number " + iCounter);
            iCounter ++;
        }
        System.out.println ("Outside the loop the value of the counter is " + iCounter);
        System.out.println ("Bye!");
    } //end main method
} //end class
```

The diagram illustrates the three components of a while loop with arrows pointing to the corresponding code lines:

- Initialize the loop control variable (iCounter)**: Points to the line `int iCounter = 1 ;`.
- Test using the loop control variable. (Evaluate loop continuation condition)**: Points to the line `while (iCounter <= 5)`.
- Change the loop control variable.**: Points to the line `iCounter ++;`.

We can write a program delivering the same results using a for loop.

```

import java.util.Scanner;
public class FiveGreetings
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iCounter ;
        for (iCounter = 1; iCounter <= 5; iCounter++)
            System.out.println ("This is greeting number " + iCounter);
        System.out.println ("Outside the loop the value of the counter is " + iCounter);
        System.out.println ("Bye!");
    } //end main method
} //end class

```

```

This is greeting number 1
This is greeting number 2
This is greeting number 3
This is greeting number 4
This is greeting number 5
Outside the loop the value of the counter is 6
Bye!

```

This is the explanation of the structure and logic of the for loop:

Initialize the loop  
control variable  
(iCounter)

**for (iCounter = 1; iCounter <= 5; iCounter++)**  
System.out.println ("This is greeting number " + iCounter);

- The loop header consists of:
  - The Java keyword *for*.
  - Three instructions inside parentheses.
  - Semicolons separating the instructions.

Test using the loop control variable.

(Evaluate loop continuation condition)

The logic behind the condition is:

if (iCounter <=5) {do the loop again} else {jump to the end of the loop}

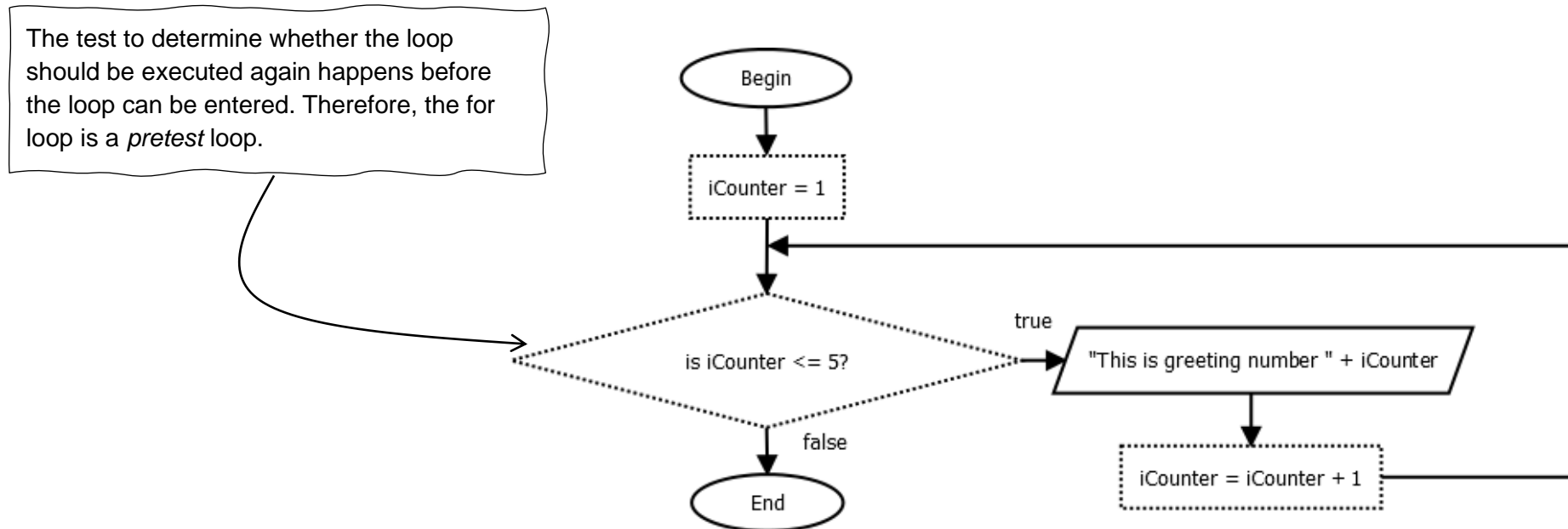
Change the loop control variable.

## Note

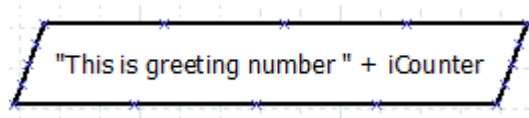
- If the body of the for loop contains one statement only, no braces are needed.
- If the body of the for loop contains more than one statement, braces should be used to indicate the beginning and end of the body of the loop. For example:

```
for (iCounter = 1; iCounter <= 5; iCounter++)  
{ //begin body of for loop  
    System.out.println ("This is greeting number " + iCounter);  
    System.out.println("*****");  
    System.out.println("");  
} //end of body of for loop
```

The for loop is also a *pretest* loop (like the while loop). This becomes clearer when the logic of the **for** loop is presented as a flowchart.



As the programmer you code the output statement as displayed in the parallelogram.

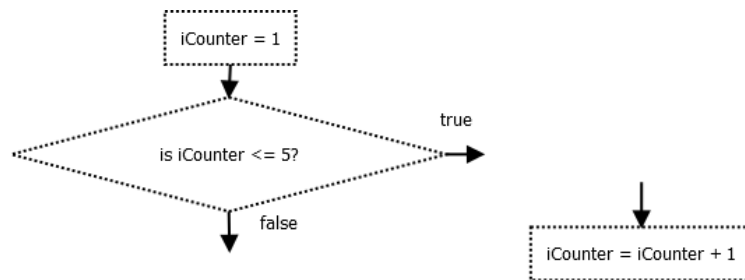


translates to



```
System.out.println ("This is greeting number " + iCounter);
```

The shapes in the dotted line in the flowchart represents all the actions executed by the for loop. The initialization, comparison (condition) and increasing of the loop control variable happens automatically and at different stages. As the programmer you write the loop header, and Java executes all the actions as indicated in the flowchart.



translates to



```
for (iCounter = 1; iCounter <= 5; iCounter++)
```

Many programs can be written with either a while, do-while or for loop. Look the the following example:

### Display a multiplication table

This program will ask the user which multiplication table must be displayed, as well as how many values of the table to display. The same output can be achieved using a while, do-while or for loop.

You can decide which solution you would prefer.

```
Which multiplication table do you want to see? : 7
How many values of the 7 X multiplication table do you want? : 8
1 times 7 = 7
2 times 7 = 14
3 times 7 = 21
4 times 7 = 28
5 times 7 = 35
6 times 7 = 42
7 times 7 = 49
8 times 7 = 56
```

```

package timeswhile;
import java.util.Scanner;
public class TimesWhile {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int iHowmany, iAnswer, iCounter = 1, iTimes;
        System.out.print("Which multiplication table do you want to see? : ");
        iTimes = keyboard.nextInt();
        System.out.print("How many values of the " + iTimes + " X multiplication table do you want? :");
        iHowmany = keyboard.nextInt();

        while (iCounter <= iHowmany)
        {
            iAnswer = iCounter * iTimes;
            System.out.println(iCounter + " times " + iTimes + " = " + iAnswer);
            iCounter ++;
        }
    }
}

```

```

package timesdowhile;
import java.util.Scanner;
public class TimesDowhile {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int iHowmany, iAnswer, iCounter = 1, iTimes;

        System.out.print("Which multiplication table do you want to see? : ");
        iTimes = keyboard.nextInt();
        System.out.print("How many values of the " + iTimes + " X multiplication table do you want? :");
        iHowmany = keyboard.nextInt();
        do
        {
            iAnswer = iCounter * iTimes;
            System.out.println(iCounter + " times " + iTimes + " = " + iAnswer);
            iCounter ++;
        }
        while (iCounter <= iHowmany);
    }
}

```

```

}
package timesfor;
import java.util.Scanner;
public class TimesFor {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int iHowmany, iAnswer, iCounter, iTimes;

        System.out.print("Which multiplication table do you want to see? : ");
        iTimes = keyboard.nextInt();
        System.out.print("How many values of the " + iTimes + " X multiplication table do you want? :");
        iHowmany = keyboard.nextInt();
        for (iCounter = 1; iCounter <= iHowmany; iCounter++)
        {
            iAnswer = iCounter * iTimes;
            System.out.println(iCounter + " times " + iTimes + " = " + iAnswer);
        }
    }
}

```

## Activity 14: Problems using a for loop

- a) Re-write Activity 6b) using a for loop.

```

This is question number 1
What is 5 - 4? 1
Correct!
This is question number 2
What is 3 - 1? 1
No! 3 - 1 = 2
This is question number 3
What is 3 - 2? 1
Correct!
This is question number 4
What is 6 - 4? 2
Correct!
You had 3 correct answers.

```

- b) Write a Java program [DispCharacters] that will ask the user to enter a string that contains at least 12 characters. The program should ask the user how many of the characters in the sentence they would like to display.
- If the sentence contains less than 12 characters, an error message should be displayed, and the user must re-enter the sentence until it contains 12 or more characters.



- If the user enters a number of characters to be displayed that is less than 1 or more than 12, an error message should be displayed, and the user will be prompted to enter a correct value. The program should keep on asking how many characters will be displayed until a correct value is entered.
- When the correct value is entered the program will then display the relevant characters the user would like to see.

Consider the following output examples to test your algorithm.

```
Enter a string containing at least 12 characters: Hallo how are you?
How many characters do you want to display? < pick any integer from 1 to 18>: 5
Good number!
I will display the first 5 character(s) of: Hallo how are you?
Character [0] is      H
Character [1] is      a
Character [2] is      l
Character [3] is      l
Character [4] is      o
```

```
Enter a string containing at least 12 characters: We empower
The string must contain at least 12 characters - please re-enter: We empower people.
How many characters do you want to display? < pick any integer from 1 to 18>: 20
Invalid number, Try Again
How many characters do you want to display? < pick any integer from 1 to 18>: 4
Good number!
I will display the first 4 character(s) of: WE EMPOWER PEOPLE.
Character [0] is      W
Character [1] is      E
Character [2] is
Character [3] is      E
```

### Important

You can also use a character as a counter in a for loop. For example, the following program displays the ASCII value for all lower case characters:

```
char c;
for (c = 'a'; c <= 'z'; c++)
    System.out.println("ASCII value of " + c + " = " + (int) c);
```

```
ASCII value of a = 97
ASCII value of b = 98
ASCII value of c = 99
ASCII value of d = 100
ASCII value of e = 101
ASCII value of f = 102
ASCII value of g = 103
```

### 5.3.2 Trace table for a for loop

Remember that a trace table is a tool to determine whether the order of statements, the calculations and the output deliver the required results. The following program is supposed to calculate the cube value of numbers in a certain range. The Java code looks as follows:

```
import java.util.Scanner;
public class DisplayCubes
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iCounter, iCube ;
1       System.out.println("I will display the cubed values of numbers in the range you choose.");
2       System.out.print("Enter first value: ");
3       int iFirst = keyboard.nextInt();
4       System.out.print("Enter last value: ");
5       int iLast= keyboard.nextInt();

6       for (iCounter = iFirst; iCounter <= iLast; iCounter++)
        {
7           iCube = (int) Math.pow(iCounter, 3);
8           System.out.println ("The cube of " + iCounter + " is " + iCube);
        }
    } //end main method
} //end class
```

```
I will display the cubed values of numbers in the range you choose.
Enter first value: 4
Enter last value: 9
The cube of 4 is 64
The cube of 5 is 125
The cube of 6 is 216
The cube of 7 is 343
The cube of 8 is 512
The cube of 9 is 729
```

Let us create a trace table where the value of iFirst is 2, and the value of iLast is 4. Some of the program statements (lines) have been numbered to help you to follow the flow of the program.

Line number	Type of statement	iFirst	iLast	iCounter	iCube	Outcome of condition (iCounter <= iLast)	Output (what appears on the screen)
1	output						I will display the cubed values of numbers in the range you choose.
2	output						Enter first value:
3	input	2					
4	output						Enter last value:
5	input		4				
6	loop initialize iCounter = iFirst			2			
6	loop test is iCounter <= iLast?					is 2 <= 4? true	
7	calculation				8		
8	output						The cube of 2 is 8
6	loop increase iCounter++			3			
6	loop test is iCounter <= iLast?					is 3 <= 4? true	
7	calculation				27		
8	output						The cube of 3 is 27
6	loop increase iCounter++			4			
6	loop test is iCounter <= iLast?					is 4 <= 4? true	
7	calculation				64		
8	output						The cube of 4 is 64
6	loop increase iCounter++			5			

6	loop test is iCounter <= iLast?					false	
---	------------------------------------	--	--	--	--	-------	--

When you create and run this program, you will see that it provides the output that is displayed in the output column of the trace table.

```
I will display the cubed values of numbers in the range you choose.
Enter first value: 2
Enter last value: 4
The cube of 2 is 8
The cube of 3 is 27
The cube of 4 is 64
```

## Activity 15: Create a trace table for a for loop

- a) Create a trace table for the following program, then describe in your own words what the program does. Use the following values as input:

iFirst = 13; iLast = 19.

```
import java.util.Scanner;
public class DisplayEvenNumbers
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iNumber, iSum = 0, iHowMany = 0 ;
        System.out.print("Enter first value: ");
        int iFirst = keyboard.nextInt();
        System.out.print("Enter last value: ");
        int iLast = keyboard.nextInt();

        for (iNumber = iFirst; iNumber <= iLast; iNumber++)
        {
            if (iNumber % 2 == 0)
            {
                iSum += iNumber;
                iHowMany ++ ;
                System.out.println("Even number " + iNumber + " found.");
            }
        } //end for
        System.out.println (iHowMany + " even numbers were found.");
        System.out.println ("The sum is: " + iSum);
    } //end main method
} //end class
```

b) Create a complete trace table for the following program to determine the output.

```
public class ForBrainGym
{ //start class
    public static void main(String[] args)
    { //start main method

        int k = 20, m = 2, g;

        for (g = 2; g <= 5; g++)
        {
            k = k + m;
            m = m + 5;
        } //end for
        System.out.println ("K = " + k);
        System.out.println ("M = " + m);
        System.out.println ("G = " + g);
    } //end main method
} //end class
```

### 5.3.3 Variations of the expression to change the loop control variable

The expression that is used to change (update) the loop control variable is not restricted to adding 1 (iCounter++). The following is an example of a program that displays even numbers from 2 to 100 by adding 2 to the loop control variable (Gaddis, 2010, p. 225).

```
for (iNumber = 2; iNumber <=100; iNumber += 2)
    System.out.println(iNumber);
```

The loop control variable may be a real (double) value. For example, the following program converts kilometres to miles starting at 1 km, in intervals of 2.5 km, for values below 15 km. 1 mile = 1.60934 kilometres.

```
public class ConvertKmToMiles
{ //start class
    public static void main(String[] args)
    { //start main method

        double rKm, rMiles;
        final double MILE_KM = 1.60934;

        for (rKm = 1.0; rKm <= 15.0; rKm += 2.5)
        {
            rMiles = Math.round(rKm / MILE_KM * 100) / 100.0;
            System.out.println (rKm + "\tkm =\t" + rMiles + "\tmiles");
        } //end for
    } //end main method
} //end class
```

1.0	km =	0.62	miles
3.5	km =	2.17	miles
6.0	km =	3.73	miles
8.5	km =	5.28	miles
11.0	km =	6.84	miles
13.5	km =	8.39	miles

## Activity 16: Convert kilogram to pounds

Write a Java program using a for loop that will display a table to convert kilograms to pounds. The kilograms must start at 0.5 kg, and be displayed in intervals of 0.5 kg, up to 6 kg. Round the pounds to two decimals. (1 kg = 2.20462 pounds)

0.5	kg =	1.1	pounds
1.0	kg =	2.2	pounds
1.5	kg =	3.31	pounds
2.0	kg =	4.41	pounds
2.5	kg =	5.51	pounds
3.0	kg =	6.61	pounds
3.5	kg =	7.72	pounds
4.0	kg =	8.82	pounds
4.5	kg =	9.92	pounds
5.0	kg =	11.02	pounds
5.5	kg =	12.13	pounds
6.0	kg =	13.23	pounds

The for loop header can also be written in such a way that the loop control variable starts at a certain value, and is then decreased – so the 'counter' counts backwards. For example, the following loop counts backwards from 10 to 1.

```
for (iCounter = 10; iCounter >= 1; iCounter --)
    System.out.println (iCounter);
```

The following program converts Celcius to Fahrenheit starting at 250 degrees Celcius, down to -250 degrees in steps of 100 degrees. The formula to convert Celcius to Fahrenheit is  $F = \left(\frac{9}{5}\right)C + 32$ .



```

public class CelciusToFahrenheit
{ //start class
    public static void main(String[] args)
    { //start main method

        int iCelcius;
        double rFahrenheit;

        for (iCelcius = 250; iCelcius >= -250; iCelcius -=100)
        {
            rFahrenheit = (9.0/5) * iCelcius + 32;
            System.out.println (iCelcius + " C\t=\t" + rFahrenheit + " F");
        } //end for
    } //end main method
} //end class

```

```

250 C    =    482.0 F
150 C    =    302.0 F
 50 C    =    122.0 F
-50 C    =    -58.0 F
-150 C   =    -238.0 F
-250 C   =    -418.0 F

```

### Activity 17: Display values in reverse order

- a) Write a Java program [ReverseSentence ] to read a sentence from the keyboard, then display the sentence in reverse order.

(Hint: You will need the methods length() and charAt() from the class String to write this program.)

```

Enter a sentence and I will reverse it: Hi, how are you?
?uoy era woh ,iH

```

- b) Write a Java program [ASCIIUpperCase ] to display the ASCII value of all the uppercase letters starting from 'Z' down to 'A'.

```

ASCII value of Z = 90
ASCII value of Y = 89
ASCII value of X = 88
ASCII value of W = 87
ASCII value of V = 86
ASCII value of U = 85
ASCII value of T = 84

```

### Important

- It is possible to write the header of a for loop so that the loop will not execute at all. The following loop will never execute:

```
for (iCount = 10; iCount <= 0, iCount ++ ) System.out.print(iCount);
```

The first time the loop continuation condition is tested, the condition will be: (10 <= 0), which is false. The control of the program will go to the first statement after the end of the loop.

### Activity 18: Predict the number of iterations of a for loop

Create trace tables to predict the number of iterations (times the loop will execute) for each of the following for loops. (Hint: if the line `System.out.println("*");` is part of the body of the loop, how many \* will be displayed. That is the number of times the loop will execute.)

- a) for (a = 12; a <= 19; a++)
- b) for (b = 6; b <= 17; b += 2)
- c) for (c = 1; c <= 10 ; c += 3)
- d) for (d = 9; d >= -2 ; d --)
- e) for (e = 0.25; e <= 5.5 ; e += 0.25)
- f) for (f = 10; f <= 16; f -= 2)

### Activity 19: Predict the output of programs using a for loop

Determine the exact output of each of the following programs.

*Activity 19 a)*

```
int w = 5, x = 2, y;  
for (y = 2; y <= 4; y ++ )  
{  
    w = w + x ;  
    System.out.println("w = " + w + " and y = " + y);  
}  
System.out.println("Outside the loop w = " + w + " and y = " + y);
```

*Activity 19 b)*

```
public class Unit5Act19b
{ //start class
    public static void main(String[] args)
    { //start main method

        int k = 4, m = 7, j;
        int n = m - k;
        for (j = 17; j >= 13; j -= 2)
        {
            if (m > n)
            {
                n = n + j;
                System.out.print(n);
            }
            else
            {
                k = m + n;
                System.out.println(k);
            }
        }
        System.out.println(j);

    } //end main method
} //end class
```

### Activity 19 c)

```
public class Unit5Act19c
{ //start class
    public static void main(String[] args)
    { //start main method

        int a = 4, b = 2, k, iSum;
        for (k = 1; k <= 3; k++)
        {
            a = a + 2;
            b = b + a;
            System.out.println("a = " + a + "b = " + b);
        }
        iSum = a + b;
        System.out.println("The sum of a and b is " + iSum);
    } //end main method
} //end class
```

### Activity 20: Solve problems using a for loop

- a) Write a Java program [Underline] to read a sentence from the keyboard. Display the sentence, with the same number of "\_" characters underneath it, so it appears as if the sentence is underlined. See the example output to show what should be displayed.

```
Enter a sentence and I will underline it: Hallo how are you?
Hallo how are you?
_____
```

- b) Viewers could vote for their favourite artist on a TV show. Two artists were nominated, and viewers could vote by sending the character A or B to a cell phone number. Write a Java program [VoteArtist] that will simulate this voting process. The program should first read the number of viewers that voted, and then read the code that every viewer sent for their favourite artist. Any code other than A or B should not be counted. The user should be allowed to enter upper- or lower-case characters. The number of votes for each artist should be displayed, as well as the code (A or B) of the winner, or a message if they have the same number of votes. Use the example output screenshots to help you to plan the program.

```
How many viewers voted?: 5
Viewer 1: A or B? a
Viewer 2: A or B? B
Viewer 3: A or B? C
Illegal code, the vote will not be counted.
Viewer 4: A or B? A
Viewer 5: A or B? A
Artist A has 3 votes.
Artist B has 1 votes.
A is the winner!
```

```
How many viewers voted?: 6
Viewer 1: A or B? a
Viewer 2: A or B? a
Viewer 3: A or B? b
Viewer 4: A or B? b
Viewer 5: A or B? a
Viewer 6: A or B? b
Artist A has 3 votes.
Artist B has 3 votes.
It is a draw!
```

- c) The factorial of an integer, e.g. 5, is the product of  $5*4*3*2*1$ . Calculate the factorial of an integer value entered by the user [Factorial].

```
I will calculate the factorial.
Enter a number: 5
5! = 120.0
```

```
I will calculate the factorial.
Enter a number: 100
100! = 9.33262154439441E157
```

- d) The waiters in a restaurant receive tips for their services rendered at the end of each day. Enter the number of waiters and the tips each waiter received for the day. Write a Java program [Waiters] to calculate and display the average amount received per waiter. Each waiter worked for 7.5 hours per day. Also calculate and display the average amount of tips per hour. Use constant values where possible.

```
How many waiters? 4
Tips for waiter 1 R350
Tips for waiter 2 R450.8
Tips for waiter 3 R500.80
Tips for waiter 4 R300
Average tips for the day is: R400.40
Average tips per hour is: R53.39
```

## 5.4 Loop control

As you have seen, there are different ways in which a loop can be controlled. (Information in this paragraph have been adapted from (Wassermann, et al., 2014)).

### 5.4.1 *Count-controlled loops*

A loop is count controlled if:

- the loop control variable is initialised, and the value of the variable is changed in the loop.
- a relational expression is used to test the value of the counter against a specific value to determine whether the loop should continue.

A count-controlled loop/repetition is also called a *definite* loop, because the number of repetitions / iterations is known before the loop begins executing.

The following code segments use *count-controlled* loops:

<p>This loop will execute 10 times.</p> <pre>for (iCounter = 1; iCounter &lt;= 10; iCounter ++)</pre>	<p>This loop will execute 26 times.</p> <pre>for (c = 'Z'; c &gt;= 'A'; c --)</pre>
---	---

<pre>iCounter = 1; while (iCounter &lt;= 12) {     iProduct = iCounter * 9;     System.out.println(iCounter + " * 9 = " + iProduct);     iCounter ++; }</pre>	<p>This loop will execute 12 times.</p>
---	---

## 5.4.2 Sentinel-controlled loops

A loop is sentinel controlled if:

- the loop control variable is tested against a special data value that indicates 'the end of data' (the sentinel value).
- the sentinel value is a value that will never be a valid data value – an out of range or impossible value.

A sentinel value can also be called a / signal / trailer / dummy value.

A sentinel-controlled loop is also referred to as an *indefinite* repetition since the number of times the loop will be executed cannot be determined before time.

The following code segments use *sentinel-controlled* loops:

```
System.out.print("Enter the donation amount. Enter 0 to stop: ");
rDonate= keyboard.nextDouble();

while (rDonate != 0)
{ //begin while
    iCounter++;
    rTotal= rTotal + rDonate;
    System.out.print("Enter the donation amount. Enter 0 to stop: ");
    rDonate= keyboard.nextDouble();
} //end while
```

The value 0 is the sentinel value. It will indicate to the loop that it should terminate.

```
String sName;
int iNumLearners = 0, iAvg;
double rTotal = 0, rPerc;
System.out.print("Enter student's name. <*> to stop: ");
sName = keyboard.nextLine();
while (sName.charAt(0) != '*')
{
    iNumLearners ++;
    System.out.print("Enter percentage for " + sName + ": ");
    rPerc = keyboard.nextDouble();
    rTotal += rPerc;
    keyboard.nextLine(); //remember to clear the keyboard buffer!
    System.out.print("Enter student's name. <*> to stop: ");
    sName = keyboard.nextLine();
}
```

The sentinel value is '\*'. It is impossible that a person's name will be \*.

Notice how you need to use the `charAt(0)` method to compare the first character in the string to the character '\*'.  
←



```
iAvg = (int) Math.round(rTotal / iNumLearners);
System.out.print("Average is " + iAvg + "%");
```

### 5.4.3 Result-controlled loops

A loop is result-controlled if

- it continues until a specific condition has been met (a result has been obtained).

A result-controlled loop is also regarded as an *indefinite* repetition.

Other names for a result-controlled loop are *free* loop and *general* loop.

The following code segments use result-controlled loops:

```
do
{
    iDie1 = rand.nextInt(6) + 1;
    iDie2 = rand.nextInt(6) + 1;
    iCountThrows ++;
    System.out.println ("I threw " + iDie1 + " and " + iDie2);
}
while (iDie1!= iDie2);
System.out.println ("I had to roll the dice " + iCountThrows + " times to get a double.");

System.out.print("What is your answer <Y>es or <N>o?: ");
cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
while (cAnswer != 'Y' && cAnswer != 'N')
{ //begin while
    System.out.print("You should only enter Y or N. \nPlease re-enter your choice: ");
    cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
} //end while
```

## 5.5 Using nested structures

It is possible to combine decision structures with repetition structures in any way necessary to solve a problem. For example, a for loop nested inside a while loop, a do-while inside an if or else block.

The following programs demonstrate some nested structures.

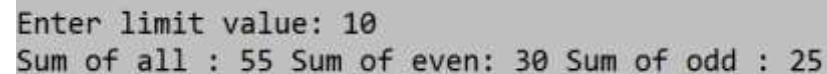
### Example 1 (Adapted from (Kjell, n.d.))

The following program adds up odd and even numbers from 1 to the limit specified by the user.

```
import java.util.Scanner;
public class AddUpIntegers
{
    public static void main (String[] args )
    {
        Scanner scan = new Scanner( System.in );
        int N, iSumAll = 0, iSumEven = 0, iSumOdd = 0;

        System.out.print( "Enter limit value: " );
        N = scan.nextInt();

        int iCount = 1;
        while (iCount <= N)
        {
            iSumAll += iCount ;
            if ( iCount % 2 == 0 ) iSumEven = iSumEven + iCount ;
            else iSumOdd = iSumOdd + iCount ;
            iCount++;
        }
        System.out.print ( "Sum of all : " + iSumAll );
        System.out.print ( "\tSum of even: " + iSumEven );
        System.out.println( "\tSum of odd : " + iSumOdd );
    }
}
```



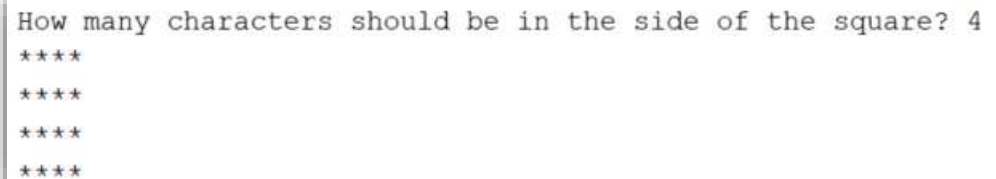
Enter limit value: 10  
Sum of all : 55 Sum of even: 30 Sum of odd : 25

An if .. else statement is nested inside a while loop.

### Example 2 (Obtained from (Buitendag))

A square made up of a certain number of characters can be drawn using a nested for-loop.

```
package drawsquare;
import java.util.Scanner;
public class DrawSquare {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int iSideSquare, iRow, iCol;
        System.out.print("How many characters should be in the side of the square? ");
        iSideSquare = keyboard.nextInt();
        for (iRow = 1; iRow <= iSideSquare; iRow++)
        {
            for (iCol = 1; iCol <= iSideSquare; iCol++)
            {
                System.out.print("*");
            } //end for iCol
            System.out.println(""); //go to the next line
        } //end for iRow
    } //end main
}
```



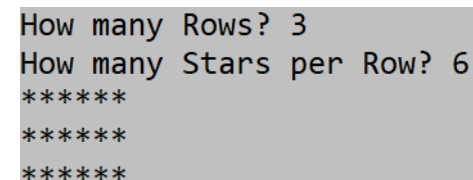
```
How many characters should be in the side of the square? 4
****
****
****
****
```

### Example 3 (Adapted from (Kjell, n.d.))

We want a program that writes out a certain number of rows of a certain number of stars, such as the following:

```
import java.util.Scanner;
public class StarBlock
{
    public static void main (String[] args )
    {
        Scanner scan = new Scanner( System.in );
        int iNumRows;      // the number of Rows
        int iNumStars;      // the number of stars per row
        int iRow;           // current row number
        int iStar;          // current star number

        // collect input data from user
```



```
How many Rows? 3
How many Stars per Row? 6
*****
*****
*****
```

```
System.out.print( "How many Rows? " );
iNumRows = scan.nextInt() ;
```

```
System.out.print( "How many Stars per Row? " );
iNumStars = scan.nextInt() ;
```

A for loop is nested inside another for loop.

```
for (iRow = 1; iRow <= iNumRows; iRow ++ ) //outer for loop
{
    for (iStar = 1; iStar <= iNumStars; iStar++ ) //inner for loop
    {
        System.out.print("*");
    }
    System.out.println();           // do this to end each line
} //end for
}
```

Notice how you the inner for loop will reset the variable iStar again each time the outer for loop is repeated.

#### Example 4 (Adapted from (Kjell, n.d.)

The following program uses several loops to draw a tree.

```
import java.util.Scanner;
public class PineTree
{
    public static void main (String[] args )
    {
        Scanner scan = new Scanner( System.in );
        int iRowCounter;      // the number of Rows
        int iStar;            // current star number
        int iBlankCounter;

        for (iRowCounter = 1; iRowCounter <= 8; iRowCounter ++ ) //draw 8 Lines
        {
            for (iBlankCounter = 1; iBlankCounter <= (8 - iRowCounter); iBlankCounter ++ )
                System.out.print(" "); //print a number of blanks
            for (iStar = 1; iStar <= (iRowCounter * 2) - 1; iStar++ ) //print the correct number of stars
            {
                System.out.print("*");
            }
            System.out.println(); //go to the next line
        }
    }
}
```

```

      *
     ***
    *****
   *********
  ***********
 *****
*****
      ***
     ***
    ***
```

```

    for (iRowCounter = 1; iRowCounter <= 3; iRowCounter ++ ) //draw the stem of the tree - 3 rows
    {
        for (iBlankCounter = 1; iBlankCounter <= 6; iBlankCounter ++ ) //each row five blanks
            System.out.print(" ");
            for (iStar = 1; iStar <= 3; iStar++ ) //each row 3 stars
                System.out.print("*");
            System.out.println(); //go to the next line
        }
    } //main
} //class

```

**Example 5** (Adapted from PPA115D Study pack 2020)

Lerato runs a small sphaza shop. She sells sweets and loaves of bread to the public. She contracts you as a programmer to create an application that will allow her to service her customers as they buy.

- A customer can order as many items as they want. The application must allow Lerato to enter the number of items each customer buys, and accumulate the amount due by the customer.
- She must be able to enter the amount paid by the customer, and the change must be calculated.
- For each customer a summary of the number of items bought and the amount paid must be printed.
- At the end of the day, a report displaying the total number of sweets, number of loaves of bread and the total amount of money collected must be displayed.

```

import java.util.Scanner;
import java.text.DecimalFormat;
public class LeratoShop
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        DecimalFormat formatter = new DecimalFormat ("R #,##0.00");
        //declare variables
        boolean bValidPay;
        char cMenuChoice, cAnotherCustomer;
        int iNumItems, iSweetsThisCustomer = 0, iLoavesThisCustomer = 0,
            iTotSweetsSold = 0, iTotLoavesSold = 0;
        double rPayment, rChange, rTotAmtMade = 0, rThisCustomerPay = 0, rItemCost;
    }
}

```

```

do
{
    rThisCustomerPay = 0 ;
    //display the menu
    System.out.print("Welcome to Lerato's Spaza Shop:" + "\n" +
        "Enter B/b -- to buy bread -- R12.00" + "\n" +
        "Enter S/s -- to buy sweets -- R0.50\n" +
        "Enter Z/z -- to exit" + "\n\n" +
        "Menu choice: ");
    cMenuChoice = Character.toUpperCase(sc.next().charAt(0));

    while (cMenuChoice != 'Z')
    {
        switch(cMenuChoice)
        {
            case 'B':
                System.out.print("Number of loaves of bread: ");
                iNumItems = sc.nextInt();
                rItemCost = 12.00 * iNumItems;
                iLoavesThisCustomer += iNumItems;
                iTotLoavesSold+=iNumItems;
                break;
            case 'S':
                System.out.print("Number of sweets: ");
                iNumItems = sc.nextInt();
                rItemCost = 0.50 * iNumItems;
                iSweetsThisCustomer += iNumItems;
                iTotSweetsSold += iNumItems;
                break;
            default:
                System.out.println(cMenuChoice + " is invalid. Please enter either <B/S/Z> ...");
                rItemCost = 0;
        } //end case B S or Z

        //update the payable amount
        rThisCustomerPay = rThisCustomerPay + rItemCost;
    }
}

```

```

        //display the menu
        System.out.print("Enter B/b -- to buy bread -- R12.00" + "\n" +
            "Enter S/s -- to buy sweets -- R0.50\n" +
            "Enter Z/z -- to exit" + "\n\n" +
            "Menu choice: ");
        cMenuChoice = Character.toUpperCase(sc.next().charAt(0));

    } // end while (cMenuChoice != 'Z')
    // this customer is done shopping - calculate payment

    //display the payable amount
    do
    {
        bValidPay = true;
        if (rThisCustomerPay > 0) //in case Lerato chooses 'Z' without entering any items for the customer
        {
            System.out.print("The payable amount is: " + formatter.format(rThisCustomerPay) +
                ". \nAmount customer pays R");
            rPayment = sc.nextDouble();
            if (rPayment >= rThisCustomerPay)
            {
                rChange = rPayment - rThisCustomerPay;
                System.out.println("The change is " + formatter.format(rChange));
                rTotAmtMade += rThisCustomerPay;
            }
            else
            {
                System.out.println("The payment is invalid.");
                bValidPay = false;
            } //end else
        } //end if

    } //end do
    while (!bValidPay); //ask again while the payment is invalid

    //display the items of the customer
    System.out.println("The customer bought the following items: " + "\n" +
        "Number of loaves: " + iLoavesThisCustomer + "\n" +

```

```

        "Number of sweets: " + iSweetsThisCustomer + "\n" +
        "Total amount paid: " + formatter.format(rThisCustomerPay) + "\n");

        //reset the values
        iLoavesThisCustomer = 0;
        iSweetsThisCustomer = 0;
        rThisCustomerPay = 0;

        System.out.print("Lerato, do you have another customer? <Y/N>: ");
        cAnotherCustomer = Character.toUpperCase(sc.next().charAt(0));
    } //end do
    while(cAnotherCustomer == 'Y');

    //display the summary report
    System.out.println("Lerato's Daily Summary Report:" + "\n" +
        "Number sweets sold : " + iTotSweetsSold + "\n" +
        "Number of loaves sold: " + iTotLoavesSold + "\n" +
        "Total amount made: " + formatter.format(rTotAmtMade));
} //end main
} //end class

```



## 5.6 More challenging activities implementing a while loop (not for assessment)

### 5.6.1 To do mathematical calculations

Various mathematical calculations require the use of a loop. We are going to provide a number of examples.

#### Find the smallest factor of a number

Program adapted from (Reges & Stepp, 2011, p. 327)

```
import java.util.Scanner;
public class SmallestFactor
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        System.out.print("For which number do you want the smallest factor? : ");
        int iNumber = keyboard.nextInt();
        int iFactor = 2;
        while (iNumber % iFactor != 0)
        {
            iFactor ++;
        }
        System.out.print("The smallest factor of " + iNumber + " is " + iFactor);
    } //end main method
} //end class
```

This is a result-controlled loop.

Since the loop contains only one statement, you can write it as follows.

```
while (iNumber % iFactor != 0) iFactor ++;
```

#### Activity 21: Find the largest factor of a number

Write a Java program to find the largest factor of a number entered by the user. For example, the largest factor of 100 is 50.

#### Test if a value is a prime number

In computational Mathematics you learned that:

- By definition, a value  $\leq 1$  is not a prime number.
- A prime number has only two factors – the value 1, and the number itself.
- An even number is not a prime number (except 2).

Programmers use different algorithms to determine whether a value is a prime number or not. Some algorithms are more effective than others. Here is a Java program that will correctly determine whether a number is prime or not.

```
import java.util.Scanner;
public class PrimeOrNotShort
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iNumber;
        int iTestValue = 2;
        boolean bFactorFound = false;
        System.out.print("I will test if a number is a prime number or not. \nEnter a number: ");
        iNumber = keyboard.nextInt();

        if (iNumber <= 1)
            System.out.println("A value <= 1 is not prime by definition.");
        else
        {
            while (iTestValue < iNumber-1)
            {
                if (iNumber % iTestValue == 0)
                    bFactorFound = true;
                iTestValue += 1;
            } //end while

            if (bFactorFound)
                System.out.println(iNumber + " is NOT a prime number");
            else
                System.out.println(iNumber + " IS a prime number");
        } //end else
    } //end main method
} //end class
```

iTestValue was initialized to have the value 2.  
All values up to one less than the value itself will be tested to determine if the test value is a factor of iNumber.

## Activity 22: Improve algorithm to determine prime number

The algorithm provided is not an efficient algorithm. Improve this program by adding code to achieve the following:

- The while loop should stop executing when a factor of iNumber is found – it should not continue testing for more factors.
- It is not necessary to test even numbers (other than 2).
- All unique divisors of a number are values less than or equal to the square root of the number, so we need not search past that.

*Hint:* The following pseudocode can be used for the while loop

```
while (iTestValue <= iNumber^(1/2)) AND (bFactorFound = false))
```

Here are test values you can enter with the results your program should give.

```
I will test if a number is a prime number or not.  
Enter a number: 0  
A number smaller or equal to one is not a prime number
```

```
I will test if a number is a prime number or not.  
Enter a number: 3  
3 IS a prime number
```

```
I will test if a number is a prime number or not.  
Enter a number: 2  
2 IS a prime number
```

```
I will test if a number is a prime number or not.  
Enter a number: 6  
6 is even - therefore NOT a prime number.
```

## Activity 23: Mathematical calculations using a while loop

- e) Write a Java program to calculate and display the sum of a certain number of the first even numbers. For example, the sum of the first 3 even numbers is  $2 + 4 + 6 = 12$ . The user should enter the number of even numbers that should be added.
- f) Adapt the program to display the output as follows:

```
How many even numbers must be added? 5  
2 + 4 + 6 + 8 + 10 = 30
```

```
How many even numbers must be added? 5  
The sum of the first 5 even numbers is: 30
```

# Works referenced and/or consulted

- Baeldung. (2019, August 2). *Lossy Conversion in Java*. Retrieved March 07, 2020, from Baeldung: <https://www.baeldung.com/java-lossy-conversion>
- Deitel, P., & Deitel, H. (2015). *Java How to program : Early Objects*. Essex: Pearson education Limited.
- edSurge. (2015, June 28). Retrieved from Learn to Code, Code to Learn: <https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn>
- Erasmus, H. G., & Pretorius, C. M. (2012). *Basic Programming Principles* (2nd ed.). Cape Town, South Africa: Pearson Education South Africa (Pty) Ltd.
- Farrel, J. (2002). *Programming Logic and Design, Introductory* (2nd ed.). Boston: Course Technology.
- Gaddis, T. (2010). *Starting out with Java: From control structures to objects* (4th ed.). Boston, MA 02116: Pearson Education, Inc.
- HiClipart. (n.d.). Retrieved September 2, 2020, from HiClipart: <https://www.hiclipart.com/free-transparent-background-png-clipart-ifxaw>
- Jenkov, J. (2015, March 15). *Java Math Operators and Math Class*. Retrieved from Jenkov.com Tech and Media Labs: <http://tutorials.jenkov.com/java/math-operators-and-math-class.html#:~:text=The%20Java%20Math%20Class,-The%20Java%20Math&text=The%20Math%20class%20contains%20methods,and%20not%20in%20the%20java>.
- Johari, A. (2019, November 26). *What are the different Applications of Java?* Retrieved from edureka!: <https://www.edureka.co/blog/applications-of-java/>
- Kjell, B. (n.d.). *Part4; Chapter 17*. (Central Connecticut State University) Retrieved February 02, 2021, from Introduction to Computer Science using Java: [https://chortle.ccsu.edu/java5/Notes/chap17/ch17\\_7.html](https://chortle.ccsu.edu/java5/Notes/chap17/ch17_7.html)
- Lang, Y. D. (2013). *Introduction to Java programming: Comprehensive Version* (9th ed.). London: Pearson.
- Murach, J. (2017). *Murach's Java programming* (5th ed.). United States of America: Mike Murach & Associates, Inc.
- Net-informations.com. (n.d.). *What's the meaning of System.out.println in Java?* Retrieved January 15, 2020, from Net-informations.com: <http://net-informations.com/java/cjava/out.htm>
- Oracle. (2019). *The Java Tutorials*. Retrieved August 18, 2020, from Oracle Java Documentation: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html>
- Reges, S., & Stepp, M. (2011). *Building Java Programs: A back to basics approach* (2nd ed.). Boston: Addison-Wesley.
- Richard, P. (2018, January 22). *Java fall through switch statements*. Retrieved August 18, 2020, from Tutorials Point: <https://www.tutorialspoint.com/Java-fall-through-switch-statements>
- The University of Texas at San Antonio. (2019). *Introduction to Object Technology*. Retrieved January 14, 2020, from UTSA CS 3443 Application Programming: Course Notes: <http://www.cs.utsa.edu/~cs3443/ch01.html>
- Tshwane University of Technology. (2019). FPIDS01:Practical Study Guide. (U. Wassermann, Compiler) Pretoria, Gauteng: Tshwane University of Technology.

tutorialspoint. (n.d.). *LEARN JAVA PROGRAMMING*. Retrieved January 14, 2020, from Java - Object and Classes:  
[https://www.tutorialspoint.com/java/java\\_object\\_classes.htm](https://www.tutorialspoint.com/java/java_object_classes.htm)

Wassermann, U., Gentle, E., Gibson, K., Noomé, C., Van Zyl, S., & Zeeman, M. (2014). *IT is gr8! @ Grade 11: Delphi*. Pretoria, South Africa: Study Opportunities.

Wassermann, U., Noomé, C., Gentle, E., Gibson, K., Macmillan, P., & Zeeman, M. (2011). *IT is gr8! @ Grade 10*. Pretoria, South Africa: Study Opportunities.

webopedia. (2020). *sentinel value*. Retrieved October 15, 2020, from webopedia:  
[https://www.webopedia.com/TERM/S/sentinel\\_value.html#:~:text=\(sen%C2%B4t%26%2Dn%26l%20val,and%20attempt%20to%20perform%20with.&text=Also%20referred%20to%20as%20a%20flag%20value%20or%20a%20signal%20value.](https://www.webopedia.com/TERM/S/sentinel_value.html#:~:text=(sen%C2%B4t%26%2Dn%26l%20val,and%20attempt%20to%20perform%20with.&text=Also%20referred%20to%20as%20a%20flag%20value%20or%20a%20signal%20value.)

Wikipedia. (2020, July 15). *Switch statement*. Retrieved August 19, 2020, from Wikipedia: [https://en.wikipedia.org/wiki/Switch\\_statement](https://en.wikipedia.org/wiki/Switch_statement)

## Good alternative sources

<http://orion.towson.edu/~izimand/237/LectureNotes/236-Lecture-Loops1.htm>  
<https://chortle.ccsu.edu/Java5/>

Web page containing information regarding the methods of the Character class.  
<https://www.geeksforgeeks.org/character-class-java/>