Tshwane University of Technology

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Principles of Programming A (Extended) (PPAF05D)

Introduction to Programming 115R (TROF05D)

Unit 3

Basic programs in Java

DEVELOPED: 2019-2021

Unit 3: Basic programs in Java

Assessment criteria

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

UNIT 2: Basic programming principles **Outcome:** The student must be able to apply algorithmic problem solving to basic mathematical and programming problems.
Assessment Criteria
• Input and output can be identified for a given programming problem. • The variables needed as part of an algorithm can be identified by completing an input-processing-output (IPO) table. • Valid variable names can be provided following the best practice standards of Java. • Constants to be used in a program can be identified by writing it in the correct section of an algorithm. • Valid data types for variables can be suggested by indicating the data type in an algorithm. • The data type of a variable can be converted to another type using the type casting facilities of Java. • The rules of the order of processing can be applied in writing pseudo code for a mathematical calculation. • A solution can be planned using an input-output-processing (IPO) table. • An algorithm to solve a basic mathematical problem requiring sequential steps can be designed, presented in pseudo code, and implemented in Java providing correct output. • Comments can be added in a program to clarify the flow of the program. • A solution to solve a basic mathematical problem can be created using methods of built-in classes of Java. • The output for a given algorithm using sequential steps and basic mathematical calculations can be predicted by means of a trace table. • The output of a program can be formatted to display text and numbers in a certain format. • Syntax errors can be identified and corrected in a given or a self-written program. • The difference between syntax errors, run-time and logical errors can be explained by providing examples of each, and suggesting ways to prevent, identify and solve these errors in a program.

Knowledge and skills needed

Before you attempt to work on this unit, you need to know and be able to:

- Accurately type text in a document at a reasonable speed.
- Type special characters such as {, ;, \ on the keyboard.
- Type uppercase characters using the <Shift> key.
- Identify and use the arrow keys on the keyboard.
- Use the following keys in combination:
    - <ctrl> + <c> to copy text
    - <ctrl> + <v> to paste text
    - <ctrl> + <s> to save changes in a file
- Create folders and subfolders, and store files in relevant folders.
- Identify and use the *Windows* icon ⊞ to open specific applications.
- Switch between different applications by clicking on the relevant icon in the Taskbar.
- Move files from one folder to another.
- Provide definitions/explanations for the following terms
    - The type of a file
    - An executable file
    - Machine code
    - Source code
    - Hard drive of a computer

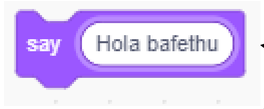
It is extremely important that you are comfortable using the keyboard, as well as File Explorer to manage files and folders, otherwise you will not be able to execute instructions quickly and will claim that Java is difficult.

Contents

3.1 Scratch vs. Java

Java is an object-oriented programming (OOP) language. It can be used to create programs/applications for various environments such as Mobile applications; Desktop GUI applications,Web-based applications and many more (Johari, 2019). Some of the differences between Scratch and Java are the following:

Programming in Scratch	Programming in Java

A program is created by dragging code blocks onto the script area. 	A program is created by typing all the programming instructions into a document.
	`public class Hola`
	`{`
	`    public static void main(String[] args)`
	`    {`
	`        System.out.println("Hola bafethu");`
	`    }`
	`}`

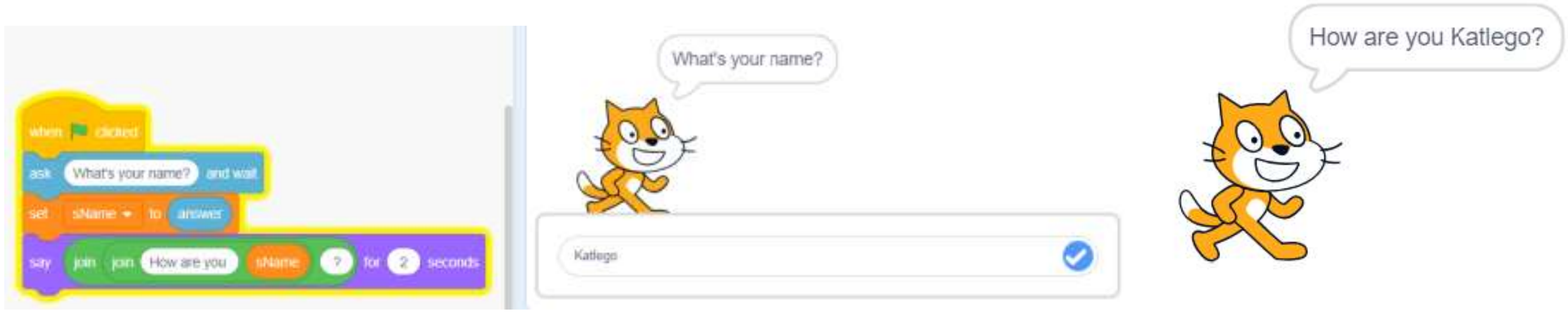
A programmer does not have to type much code, since the instructions are available as existing code blocks. A program can be created in a short time even if the programmer does not have a high level of typing skills.	No code is generated automatically. All instructions must be typed. A programmer needs to be efficient and accurate in typing.
To 'make something happen', you need to add code blocks in a script.	To 'make something happen', you need to call a *method*, for example, *println()* is a method that will display text on the screen.
A program (project) consists of one or more scripts for one or more sprites. No additional instructions are needed.	• A basic program consists of a *main()* method that is stored in a *class*. All instructions to create the class must be typed by the programmer. The name of the class can be chosen by the programmer, but the name of the method must be *main()*. • The class can contain more methods, but it must contain a main() method, otherwise it cannot perform any tasks. • The programmer must write the code to create the class and the main() method.

To run a program, you need to click on a block or a script, or you need to click on the green flag icon. Examples of other OOP languages arePython, Ruby and Smalltalk.	A programmer needs to first *compile* a program, then *run* (execute) the program in a second step. • A *compiler* is a special program that translates a program written in one language (such as Java) to a language the computer can understand (machine code). • When a program is compiled, the compiler cannot continue if there are any *syntax errors* (mistakes in the programming code that does not follow the rules of the programming language).

3.2 Writing and running a Java program

3.2.1 *Compare the Scratch and Java environments*

In Scratch you write programming code in the Script Area, and the output is executed and displayed on the Stage. Input is entered in an edit box at the bottom of the stage.



```
when 🏁 clicked
ask What's your name? and wait
set sName ▾ to answer
say join join How are you sName ? for 2 seconds
```

What's your name?

Katlego

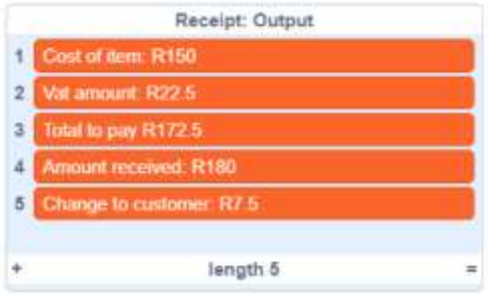
How are you Katlego?

Program code (source code) in the Script Area

Input in an edit box at the bottom of the Stage

Output in a speech bubble next to the sprite.

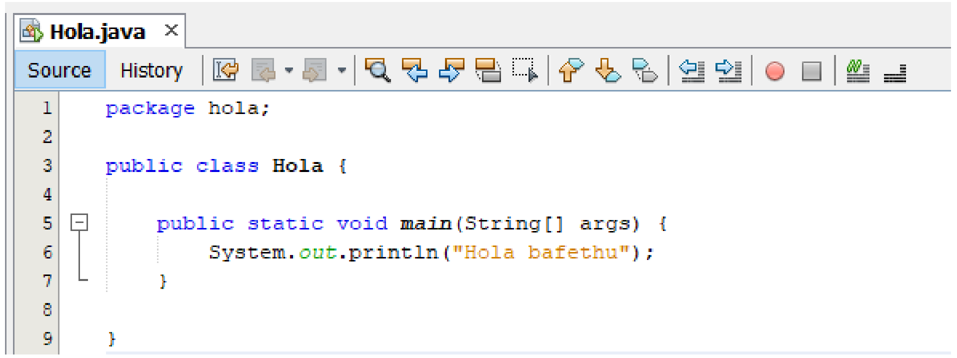
You also used the List component to display all lines of output on the Stage.



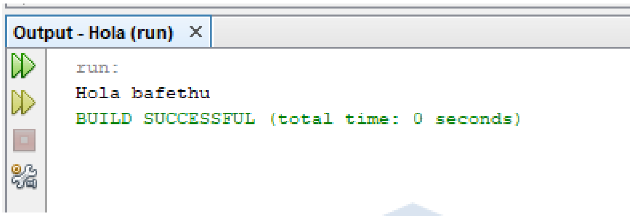
```
Receipt: Output
1 Cost of item: R150
2 Vat amount: R22.5
3 Total to pay R172.5
4 Amount received: R180
5 Change to customer: R7.5
length 5
```

In Java, all program code (the source code) must be typed in a file (we are going to use NetBeans 8.2). There are no blocks to drag and drop.

This is what you will see when you write and run a Java program.



```java
package hola;

public class Hola {

    public static void main(String[] args) {
        System.out.println("Hola bafethu");
    }

}
```



```
run:
Hola bafethu
BUILD SUCCESSFUL (total time: 0 seconds)
```

Program code (source code) typed in the source code editor of NetBeans.

An output window is displayed below the source code editor. In this window you will:

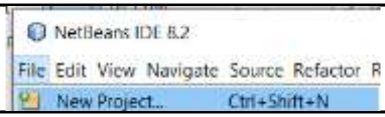
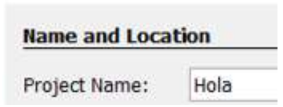
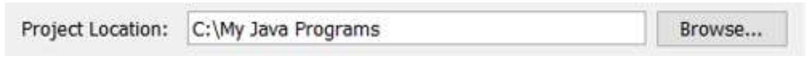
- See the prompt (the question displayed to the user).
- Enter input values.
- See the final output message.

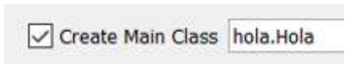
Terminology

- In Scratch programming code was stored in a file with an extension .sb3.
- Methods were called from classes to 'make something happen' to the sprite. For example, the say method was called from the Looks class to make a speech bubble with words appear next to the sprite on the stage.
- Java provides many classes that contain methods. For example, the print() method from the System class will display words on the screen.
- To create a program, the programmer must create a class that contains one method – the main() method.
- The programmer can choose the name of the class – just like you could choose the name of a Scratch program.
- This class will be stored in a file with extension .java.

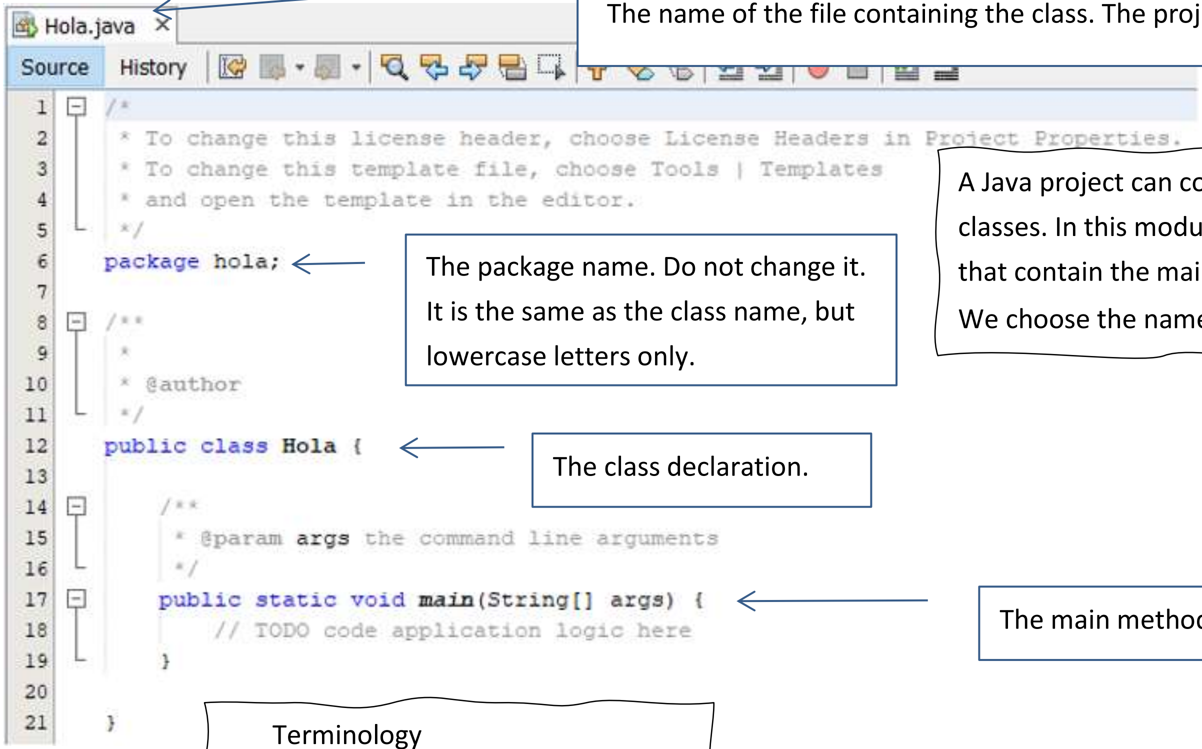
3.2.2 *First Java program*

Here are the steps to follow to create a Java program that will display a greeting on the screen. The program will be stored as a class called Hola.

1. Open NetBeans	
2. Click on File> New Project ….	 NetBeans IDE 8.2 File Edit View Navigate Source Refactor R New Project... Ctrl+Shift+N
3. Click on Next	
4. Enter a Project Name	 Name and Location Project Name: Hola The programmer chooses the project name. The project name will always be the same as the name of the class that contains the main method.
5. Choose the folder where you are going to store all your Java programs.	 Project Location: C:\My Java Programs You only need to do this once. Click on the Browse… button to select the folder. When you open NetBeans again, it will remember this location.

6. Make sure the checkbox is ticked.	 ☑ Create Main Class hola.Hola
7. Click on Finish	

You will now see the editor where you need to enter programming code. A number of lines of code is automatically created, and you should not change it.



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hola;

/**
 *
 * @author
 */
public class Hola {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

The name of the file containing the class. The project name, the class name and the file name is the same.

A Java project can contain many packages, and a package can contain many classes. In this module we will only write code for one class – the 'main' class that contain the main() method.
We choose the name of the project, NetBeans automatically creates a package

The package name. Do not change it. It is the same as the class name, but lowercase letters only.

The class declaration.

The main method declaration.

Terminology

- { } braces
- ( ) parentheses
- [ ] brackets

Where must my code be entered?

Pseudo code

```
begin
    say "Hola bafethu"
end
```

Java code

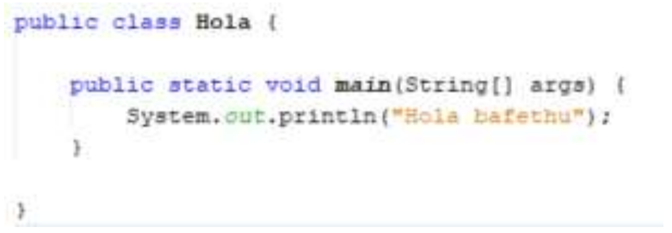
```
public class Hola {
    public static void main(String[] args) {
        System.out.println("Hola bafethu");
    }
}
```

Your programming code goes between the two braces.

```
{

}
```

Activity 1: First Java program (class name Hola)

Create the program to display a greeting in the output window.



```
public class Hola {

    public static void main(String[] args) {
        System.out.println("Hola bafethu");
    }

}
```

- Java is case-sensitive - that means it does not see, for example, the letter *a*, and the letter *A* as the same letter. Therefore, you must type each character and letter as in the example program. If you type, for example, system.Out instead of System.out, your program will not compile.
- You must type all the letters and characters exactly as they are. The only choices you have, is the name of the class, and the words you want to have displayed. You can, for example, use "Dumela Mogwera", or "Hi, how are you!".
- Take note of the indentation.
- See that braces, brackets and parentheses are always in pairs.
- The end of every programming instruction is indicated by a semi-colon.

Note

- It is a Java convention to let the name of a class start with a capital letter, for example:

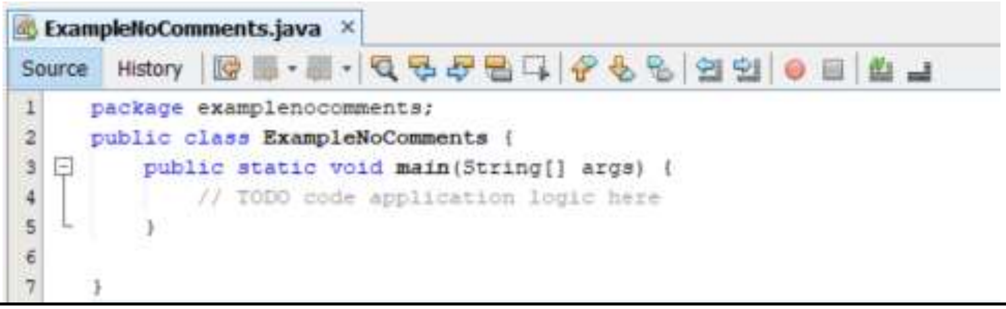
```
public class Hola
public class CalcArea
public class InstallmentPerMonth
```

- In Scratch you could attach comments to a code block. In Java you can add comments in a program in different ways.

  Option 1: Use the characters //. This is used to type *one line* of comments.

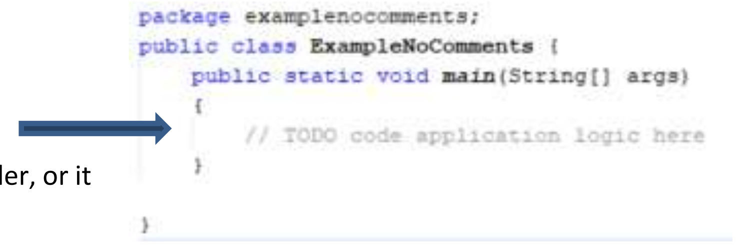
  Option 2: Use the /*   */. This is used to type a *block comment* that can run over more than one line.

- You may delete the commented lines in the NetBeans editor that is automatically added. Therefore your code may look like this before you start to add code in the main() method.



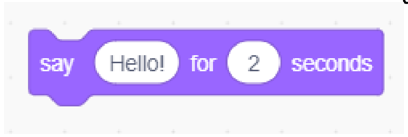
```
package examplenocomments;
public class ExampleNoComments {
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

- You will learn the meaning of the various words and characters in the header of the main() method later. What is important now, is that you may not change anything in the header.
- The left brace { indicates the start of the main() method.
- The right brace } indicates the end of the main() method.
- Your code must be entered between the braces.
- The brace indicating the start of the main method may be on the same line as the header, or it may be on the next line.



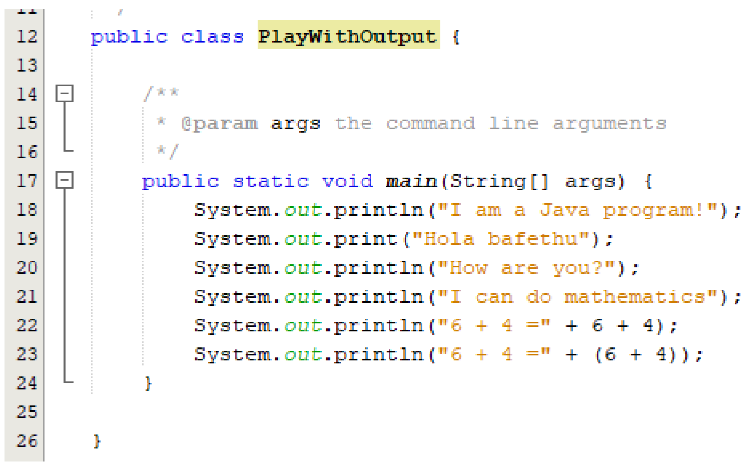
```
package examplenocomments;
public class ExampleNoComments {
    public static void main(String[] args)
    {
        // TODO code application logic here
    }

}
```

3.3 Displaying output in Java

Output in Scratch	**Output in Java**
The [say] block (method) is used to display output on the screen. 	• Two methods are available to display output on the screen.They are *print()* and *println().* (We say *'print line'* for the println() method. `System.out.println("Hallo!")` `System.out.print("How are you?")` • In Java you call a method by entering the name of the programming structure where a method is stored, and then enter the method name. • The print() and println() methods can be found in the System class in a field called out. (You will learn about fields and classes later. For now, you just need to know the methods are available in a programming structure called System.out, and you must enter the name of the structure before the methods can be called.)

Activity 2: Difference between print() and println() (class name PlayWithOutput)

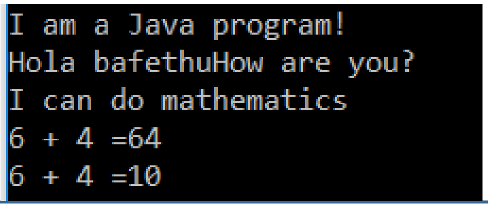
Create the following program. Compile and run it. Then answer the questions that follow.



```
public class PlayWithOutput {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("I am a Java program!");
        System.out.print("Hola bafethu");
        System.out.println("How are you?");
        System.out.println("I can do mathematics");
        System.out.println("6 + 4 =" + 6 + 4);
        System.out.println("6 + 4 =" + (6 + 4));
    }

}
```

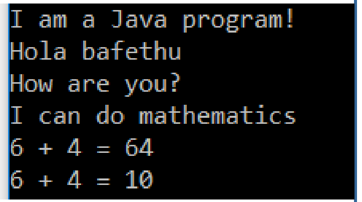
a) Describe to a fellow student what the difference is between the print() and the println() methods.

b) The output looks like this:



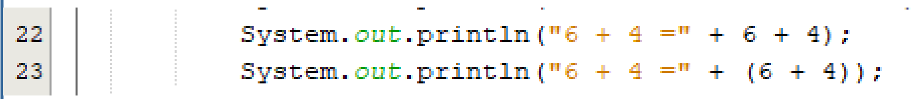
```
I am a Java program!
Hola bafethuHow are you?
I can do mathematics
6 + 4 =64
6 + 4 =10
```

Adapt the program code so that the output looks like this:



```
I am a Java program!
Hola bafethu
How are you?
I can do mathematics
6 + 4 = 64
6 + 4 = 10
```

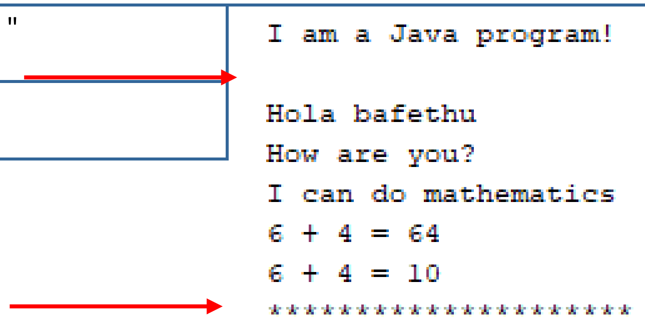
c) Why does line 22 and line 23 (of the Java code) produce different output?



```
22            System.out.println("6 + 4 =" + 6 + 4);
23            System.out.println("6 + 4 =" + (6 + 4));
```

d) Change the program so that an open line is displayed after the line "I am a Java program! "

e) Change the program so that a line of asterisks (*) is displayed at the end.



```
I am a Java program!

Hola bafethu
How are you?
I can do mathematics
6 + 4 = 64
6 + 4 = 10
*********************
```

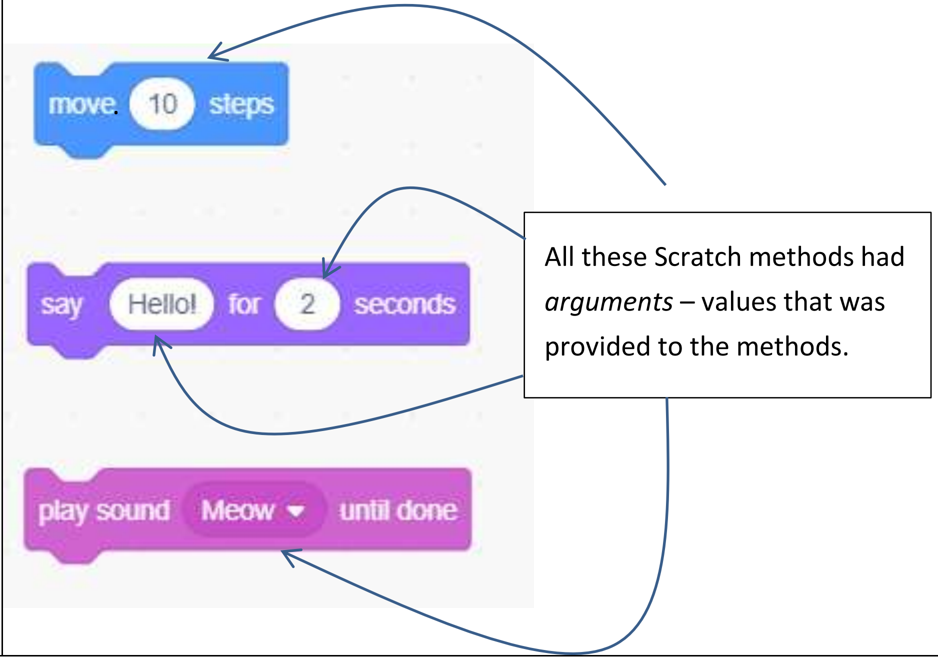
OOP Terminologies

There are important terminologies that will be referred to in the rest of this Unit. Refer to these descriptions while you work through the unit.

Methods and arguments

- A method can be seen as a set of instructions that was developed by another programmer (e.g. a Java developer). A method commands the computer to perform an action. For example, to print a line on the console (screen) we use a print() or println() method.
- The println() method needs an argument – the string that must be displayed on the console. An argument is the data that must be supplied to a method for it to perform a specific task. Arguments are always presented inside a pair of parentheses ( ).
- String (text) values are always presented in a pair of double quotes. "   "
- The syntax rules of Java require that even methods that do not need any arguments need to have parentheses.

These are examples of Scratch methods. They were called code blocks (note that the 'names' of the methods are verbs).



All these Scratch methods had *arguments* – values that was provided to the methods.

Objects

- As the name suggests, object-oriented programming (OOP) languages make use of *objects* to provide tools to build programs.
- In real life we are surrounded by objects. Look around you, you may see a desk, chair, television set, cell phone (nouns). All objects have:

characteristics	and	*behaviour*
• It describes what the object looks like. • It is also called attributes or states. • It is the object's data / *fields*. • For example, a cell phone has a certain colour, width, length, height.		• It determines the actions an object can perform – what the object can do. • It is the *methods* of an object. • For example, a cell phone can make a call, take photos, and send text messages.

- In Scratch, you used objects such as sprites and the stage (without really knowing that they were called objects).

characteristics (fields) of a sprite object	and	*behaviour (methods) of a sprite object*
• Some examples are – the sprite has a certain colour; size, costume, x and y position on the stage and a direction in which the sprite is pointing.		• Some examples are - the sprite can move, turn, glide and play a sound.

- A software object is a programming structure that can be used in combination with other objects to build a program (application).

Classes

- All real-life objects have been created using a recipe or blueprint or plan. For example, a cake is an object, but you need a recipe to describe how to bake it. A house is an object, but you need plans to build it. In the same way a software object needs to be described/defined, and the plan for a software object is called a class.
- Therefore, a class describes what fields (data / attributes) an object contains, and what methods the object has (the tasks it can perform, or its behaviour). You can say: 'A class is the recipe for an object'.
- In Scratch, you do not need to give any instructions regarding classes. Every time you choose a sprite, an object is automatically created for you,

based on the class (the recipe) for a sprite. In OOP language, we say:

- The object (sprite) was *instantiated* (created),
- A sprite is an instantiation of the class Sprite (using the recipe for a sprite).

Java classes, objects, and methods you have used so far

- You created your own class, containing one method – the main() method – it describes the behaviour of your class.
- Your class did not have any fields / data yet.
- The System class (a recipe) contains data and methods that can be called to display output in the console window.
- One of the fields of the System class is an object called *out*.
- The out object contains the print() and println() methods that you called.

`System.`	`out.`	`print`	`("Hola bafehtu!");`
Class name	field name	method name	argument – a string
Class names start with a capital letter.	Object names start with lowercase letters.	Methods names start with lowercase letters.	A method always has parentheses, even if it does not need an argument value. A string is always enclosed in double quotes " ".

Activity 3: Knowledge checklist

Use the following checklist and ensure you can correctly define or explain the following concepts:

Argument of a method	
Behaviour / actions / method of an object	
Braces, brackets and parentheses	
Class	
Compiler	
Fields /data / attributes of an object	

main() method	
Math class	
Method	
Object	
OOP	
Run a program	
Syntax errors	

Activity 4: Skills checklist

Use the checklist to ensure you can do the following:

Create a new Java project in NetBeans.	
Type code in the main() method of a class.	
Run a Java program.	
Display output in Java using the print() and println() methods.	

3.4 Obtaining values from the user

We now want to obtain values from the keyboard (the standard input device). Complete the following activity, then we will discuss the purpose of each of the additional statements (the ones highlighted in colours).

Activity 5: Read a name from the keyboard (class name GreetMe)

```
import java.util.Scanner; //import the Scanner class form the java.util package
public class GreetMe
{ //start class
     public static void main(String[] args)
     { //start main method
          Scanner keyboard = new Scanner(System.in); //create an object to read data from the keyboard
          String sName; //declare a variable
          System.out.print("What is your name?");
          sName = keyboard.nextLine(); //call a method to read data from the keyboard
          System.out.println("Dumela " + sName);
     } //end main method
} //end class
```

How will the output change if you use a *println()* method to display "What is your name? ", and a *print*() method to display the greeting?

Change the code and observe the change. Never be afraid to change code to 'see what happens'. That is how you learn to code. You do not have to wait for someone else to provide the answer.

Did you notice?

The print() method prints what was provided as an argument, then the cursor waits on the same line.

The println() method prints what was provided as an argument, then the cursor jumps to the next line.

Now we will discuss the highlighted code. The purple highlighted code shows how to declare a variable in Java.

3.4.1 *Declaring variables*

In Scratch you only need to *create* (make*)* a variable:

- Give it a name
- Indicate the scope (For all sprites, or For this sprite only).

In Java you need to *declare* a variable – the declaration process involves two steps:

- Give the variable a name that is descriptive and follows rules and conventions of the programming language.
- Indicate the *type of data* that will be stored in the variable – for example, integer, real, string, char. For the first few programs you are going to write, you will use the following data types:

  int (integer for whole numbers); double (floating point / real numbers) and String.

In Java, the scope of a variable is determined by the place in the program where the variable is declared.

To start with, we will only write programs with a main method, and all variables will 'belong' to the main method. We can say the variables have *method scope*.

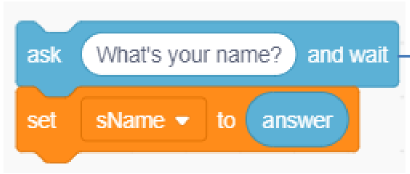
```
int       iNumStudents;
double    rAverage; //we keep the prefix as r
String    sName; //capital letter S for a String
```

So, generally, we can indicate the declaration of a variable as follows:

The data type of a string value is indicated with the capital letter S. This because Java considers a string value as a class. In the beginning you will use a String as a single value containing alphanumeric values, for example, "Grade 6", "Katlego", "0846675422". The use of methods of the String class will be discussed later.

3.4.2 *Obtaining input from the keyboard*

In Scratch you use an *ask* method to display a prompt on the stage, and a *set* method to transfer the input on the keyboard to a specific variable.



In Java you use a print() or println() method to display a prompt in the console window.

```
System.out.print("What is your name?");
```

To transfer the input from the keyboard to a specific variable in Java, you need to add additional code. The code is explained in the table below:

1.	Add code to specify that the class containing input methods will be needed. We say the Scanner class from the java.util package must be imported (added to the program).	`import java.util.Scanner;`

All the classes available in Java are organized into packages. So, a *package* is a *group of related classes*. The Scanner class is part of the java.util package. You can think of a package as a library where classes are stored on shelves. In a library each shelve contains related books (computer science, chemistry, life sciences.)

2.	Create an object called keyboard based on the Scanner class.	`Scanner keyboard = new Scanner(System.in);`

Some objects are created automatically by Java – for example, the *out* object containing the print() and println() methods.
Some methods can only be accessed once the programmer created an object in the program where the methods must be called.
The Scanner class contains methods to read values from the keyboard (the standard input device).
The programmer needs to instantiate an object based on the Scanner class before the methods to read values from the keyboard can be called.
Examples of these methods are:

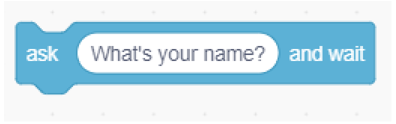
Reads real values	`nextDouble()`	`rAmount = keyboard.nextDouble();`	*NOTE* Different methods are called to read different data types from the keyboard.
Reads integers	`nextInt()`	`iAge = keyboard.nextInt();`	
Reads more than one word	`nextLine()`	`sNameSurn = keyboard.nextLine();`	
Reads a single word	`next()`	`sOneWord = keyboard.next();`	

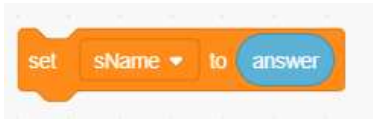
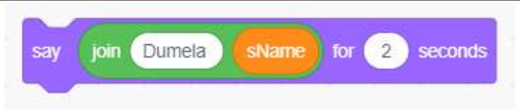
3.	Call the nextLine() method from the keyboard object to transfer the value entered on the keyboard to the variable with the name sName.	`sName = keyboard.nextLine();`

3.4.3 *Pseudo code vs Scratch vs Java*

Let us look at the way pseudo code, Scratch code and Java code are related when values are read from the keyboard and displayed on a screen. You need to realize that the pseudo code you wrote for Scratch can be translated to Java. Therefore, if you have pseudo code for the activities for the Scratch programs in Unit 2, you do not need to write pseudo code again if you want to write the same program in Java. You just translate the pseudo code to meet the Java syntax requirements.

Pseudo code	Scratch code	Java code

`ask "What's your name? "`
`display "What's your name? "`
		`System.out.print("What is your name? ");`

In Java we are going to use the word *display* in pseudocode instead of *ask*, since the methods for a prompt for input, and for displaying output is the same.		
`enter sName`		`sName = keyboard.nextLine();`
`display "Dumela " + sName`		`System.out.println("Dumela " + sName );`

Creating a Scanner object

This is a breakdown of the way an object based on the Scanner class is created in a Java program. Creating other objects follow the same pattern.

The Scanner class is stored in the java.util package. The programmer needs to import the Scanner class from the java.util package before an object based on the Scanner class can be created.

```
import java.util.Scanner;
```

`Scanner`	`keyboard`	`= new`	`Scanner`	`(System.in);`
The name (type) of the class the object should be based on. (You can also say: 'The recipe to follow.')	The name of the object. (The name you want to use in your program to refer to the keyboard). The name must start with a lower-case character. You can use any name you want, for example: kb or input	A Java keyword	A call to the Scanner method in the Scanner class. This method constructs (creates)the object called keyboard. This method is called the *constructor* method.	The data needed by the object, in this case the name of the standard input device. This is the argument of the Scanner method (the constructor).

All constructors have the same name as the class. That is why the name of the method starts with a capital letter, whereas the convention stipulates a method name should start with a lower-case letter.

Creating an object can be done in two separate steps, for example:

```
    Scanner keyboard; //declaring the object name similar to declaring a variable name
    keyboard = new Scanner (System.in); // Creating the keyboard object by calling the constructor method on the Scanner class.
```

Terminologies

When an object has been created, we say the object is an *instance of the class*.

- keyboard is an instance of the Scanner class.
- The keyboard object has been *instantiated* by calling the constructor method called Scanner.

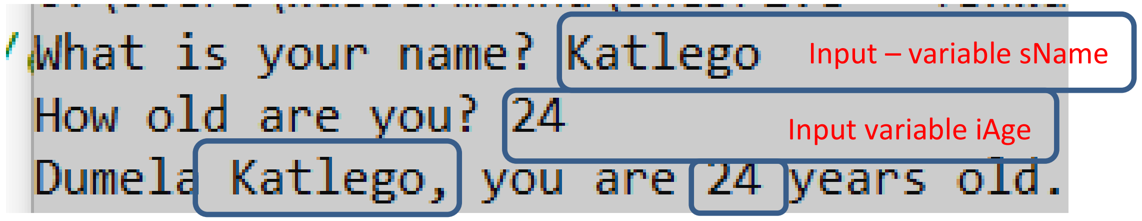
The System class

- A standard library of pre-written classes is available to all Java programs. These classes are part of the Java Application Programmer Interface (API).
- The System class is part of the API.
- The System class is stored in the java.lang package, but the package does not have to be imported.
- The System class contains an object (a field) called *out*.
- The out object is based on the PrintStream class from the java.io package. (It is an instantiation of the PrintStream class).

Activity 6: Read and display name and age (class name NameAndAge)

a) Write a Java program to read a name from the keyboard, and display a greeting using the name entered.

b) Add code to the Java program to also read the age of a person, then display a message such as the following:



```
What is your name? Katlego
How old are you? 24
Dumela Katlego, you are 24 years old.
```

Complete Java programs

Two complete examples of programs you wrote in Scratch will now be provided in Java, so that you can see how the logic involved is the same, but the syntax of the two programming languages is different.

Example 1: Calculate percentage

In Scratch you wrote a program to calculate the percentage for a test if the score and the test total are provided. Compare the Scratch code to the Java code.

The Java code will look as follows:

```
import java.util.Scanner; // Scanner is stored in the java.util package
public class CalcPercentage //Create new class CalcPercentage
{ //start class
      public static void main(String[] args)
      { //start main method
            //Create an object called keyboard, based on the Scanner class, to read data from the keyboard
            Scanner keyboard = new Scanner(System.in);
```

Classes are grouped together and stored in *packages*. Therefore, a package is a container of several classes. The java.util package includes various miscellaneous utility classes for example, classes to read data from the keyboard and to generate random numbers.

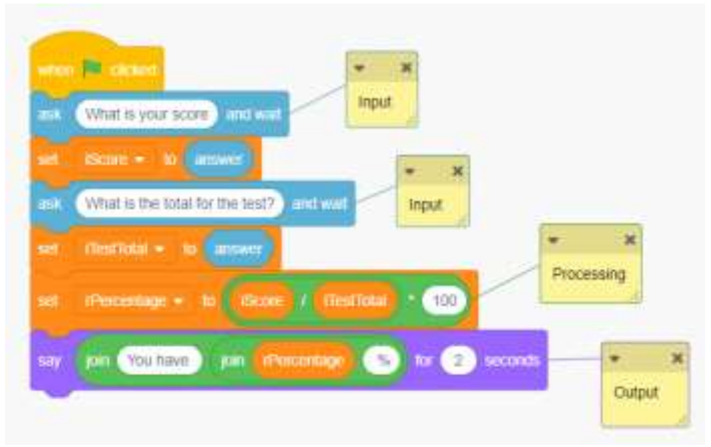
1. Create an object to read input from the keyboard using the Scanner class.

```
            //Declare variables

            int iTestTotal;
            double rScore, rPercentage;
```

2. Specify the names of all the variables you want to use in your program and indicate what *type* of data you are going to store in each variable.

```
            //Prompt the user to enter the score obtained in the test
            System.out.print("Test score:");
            //Read score from the keyboard and store it in the variable called rScore
            rScore = keyboard.nextDouble();
            //Prompt the user to enter the total for the test
            System.out.print("Test total:");
            //Read score from the keyboard and store it in the variable called iTestTotal
            iTestTotal = keyboard.nextInt();
            //Calculate the percentage
            rPercentage = rScore/iTestTotal * 100;
            //Display the percentage on the screen
            System.out.println("The percentage is: " + rPercentage + " %");
      } //end main method
} //end class
```

3. Call methods from the keyboard object to read values from the keyboard.



```
when 🏁 clicked
ask (What is your score) and wait
set [rScore ▾] to (answer)
ask (What is the total for the test?) and wait
set [iTestTotal ▾] to (answer)
set [rPercentage ▾] to (((rScore) / (iTestTotal)) * (100))
say (join (You have) (join (rPercentage) (%))) for (2) seconds
```

Type the calculation in Java similar to the way you write the calculation in pseudo code.

In Java you can write the pseudo code for this solution as follows:

```
CalcPercentage // the class name
begin //start of the class
    main method
    begin //main method

        //declare variables
        int iTestTotal
        double rScore, rPercentage

        //Prompt the user to enter the score obtained in the test
        display "Test score: "
        //Read score from the keyboard and store it in the variable called rScore
        enter rScore

        //Prompt the user to enter the total for the test
        display "Test total: "
        //Read score from the keyboard and store it in the variable called iScore
        enter iTestTotal

        //Calculate the percentage
        rPercentage = rScore/iTestTotal * 100

        //Display the percentage on the screen
        display "The percentage is: " + rPercentage + " %"

    end //main method
end //end of the class
```

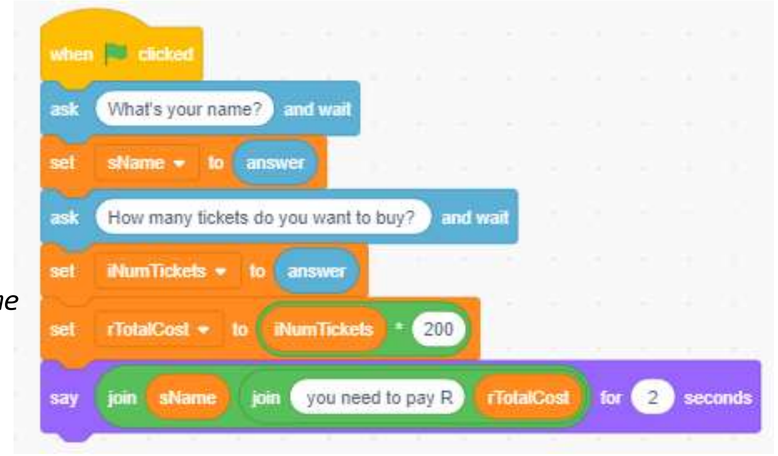
Since the data type of a variable is very important in Java, we can also indicate it in the pseudo code.

Example 2: Cost of tickets

In Scratch you wrote a program that will read a person's name and the number of tickets sold, then calculate and display the total money the person owes. The Java code can look as follows:

The code in grey forms part of the framework of code you should use for every program.

```
import java.util.Scanner; // Scanner is in the java.util package
public class CostTickets //A new class called CostTickets is created.
{ //start class
    public static void main(String[] args)
    { //start main method
        //Create an object based on the Scanner class to read input from the keyboard
        Scanner keyboard = new Scanner(System.in);
        //Declare variables
        int iNumTickets;
        double rCost;
        String sName;
        //Prompt the user to enter a name
        System.out.print("What is your name?:");
        //Read name from the keyboard store it in the variable called sName
        sName = keyboard.nextLine();
        //Prompt the user to enter the number of tickets bought
        System.out.print("How many tickets do you want to buy?:");
        //Read number of tickets from the keyboard and store it in the variable called iNumTickets
        iNumTickets = keyboard.nextInt();
        //Calculate the cost
        rCost = iNumTickets * 200;
        //Display the cost on the screen
        System.out.println(sName + " you need to pay R" + rCost);

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



```
when [flag] clicked
ask (What's your name?) and wait
set sName to (answer)
ask (How many tickets do you want to buy?) and wait
set iNumTickets to (answer)
set rTotalCost to ((iNumTickets) * (200))
say (join (sName) (join (you need to pay R) (rTotalCost))) for (2) seconds
```

Java has two ways in which methods can be accessed (called):

1. Call methods from a class (static / anonymous / class methods)

```
ClassName.methodName(arguments)
```

Call a method directly from a class. The following code shows how the *ceil()* method is called from the Math class:

```
double rNumBoxes = iLitrePaint / iInContainer;

int iNumBoxes = Math.ceil(rNumBoxes);
```

The ceil() method is called a *static / anonymous/ class* method – it can be accessed directly from a class without creating an object.

The object/class name is always followed by a dot (.), and then the method name is provided.

This is called the dot-notation.
You say *Math dot ceil*(), or
keyboard dot nextLine.

2. Call methods from an object (instance methods)

```
objectName.methodName(arguments)
```

Use a class as a 'recipe' or 'blueprint', and create an object based on the class. Then call methods of the object. The following code shows how methods are called from the keyboard object based on the Scanner class.

```
Scanner keyboard = new Scanner(System.in);
String sName = keyboard.nextLine();
int iAge = keyboard.nextInt();
```

Methods such as nextLine() and nextInt() are called *instance* methods. The methods are part of the Scanner class, but they require an object based on the Scanner class to be created before they can be called.

Keyboard is an object created by the programmer, therefore the programmer can choose the name. For example:

```
Scanner reader = new Scanner(System.in);
sName = reader.nextLine();
```

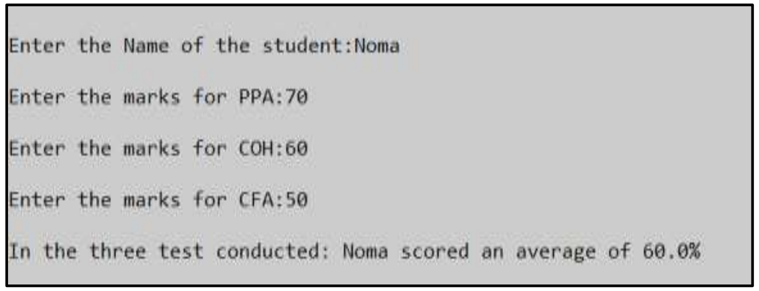
3.4.4 *Concepts and terminology regarding methods*

There are two ways in which methods perform tasks	
1. Value-returning methods	**2. Void methods**

These are examples of methods that return values when they are called:
 	
`rScore = keyboard.nextDouble();`
 	
The nextDouble() method returns a double value.
It receives no arguments.
 	
`iNumBoxes = Math.ceil(rNumBoxes);`
 	
The ceil() method returns an integer value.
It has an argument that should be a floating-point number.
 	
Value-returning methods are called by treating it as a value in an assignment statement.
	These are examples of methods that just perform tasks:
	
	`System.out.print("Hi how are you!");`
	
	The print() method displays the data that is passed to it through the argument. It does not return a value. It is called a *void* method.

Activity 7: Convert pseudo code to Java code

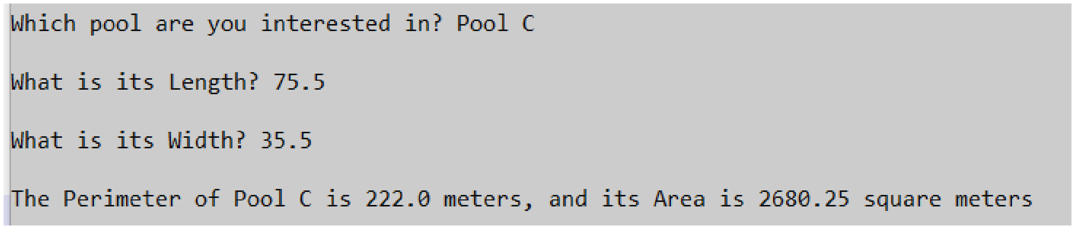
a) The pseudo code that follows aims to calculate the average percentage of a student in 3 subjects identified as PPA, COH and CFA. Translate this into a Java program, compile and execute your program. Include all the comments on your code. The code should produce results as shown in the screen shot.



```
Enter the Name of the student:Noma

Enter the marks for PPA:70

Enter the marks for COH:60

Enter the marks for CFA:50

In the three test conducted: Noma scored an average of 60.0%
```

```
CalculateAverage // the class name
begin //start of the class
     main method
     begin //main method
          //declare variables
          String sStudentName
          int iPPA,iCOH,iCFA
          double rAverage
           //Prompt the user to enter the Name of the student whose Average is to be calculated
          display "Enter the name of the student: "
          //Read the name from the keyboard and store it in variable sStudentName
          enter sStudentName
          //Prompt the user to enter the marks obtained in PPA
          display "Enter the marks for PPA: "
          //Read the marks from the keyboard and store it in variable iPPA
          enter iPPA
          //Prompt the user to enter the marks obtained in COH
          display "Enter the marks for COH: "
          //Read the marks from the keyboard and store it in variable iCOH
          enter iCOH
          //Prompt the user to enter the marks obtained in CFA
          display "Enter the marks for CFA: "
          //Read the marks from the keyboard and store it in variable iCFA
          enter iCFA
          //Calculate the average score
          rAverage = (iPPA + iCOH + iCFA )/3
          //Display the Average on the screen
         display "In the three tests conducted: " + sStudentName + " scored an average of " + rAverage + "%"
     end //main method
end //end of the class
```

b) Thandy is an Olympic swimmer with a full year membership at a sports centre. She has 3 different pools, poolA, poolB and poolC to practice at in every visit. You are expected to enter the details of a pool (name, length and width) in meters and then calculate and display the perimeter and the area of the pool. Include all the comments. Result should be as shown in the screenshot.

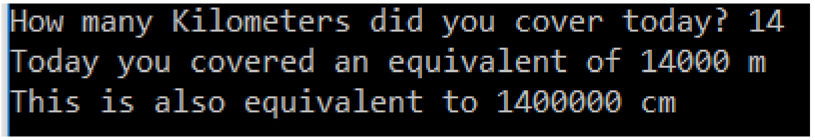


```
Which pool are you interested in? Pool C

What is its Length? 75.5

What is its Width? 35.5

The Perimeter of Pool C is 222.0 meters, and its Area is 2680.25 square meters
```

```
PoolDimentions // the class name
begin //start of the class
   main method
   begin //main method
       //declare variables
       String sPoolName
       double rLength, rWidth, rPerimeter, rArea
       //Prompt the user to provide the Name of the pool used
       display "Which pool are you interested in? "
       //Read the name from the keyboard and store it in variable sPoolName
       enter sPoolName
       //Prompt the user to enter the length of the pool
       display "What is its Length? "
       //Read the length from the keyboard and store it in variable rLength
       enter rLength
       //Prompt the user to enter the Width of the pool
       display "What is its Width? "
       //Read the length from the keyboard and store it in variable rWidth
       enter rWidth
       //Perform the calculations
       rPerimeter = 2 * (rLength + rWidth)
       rArea = rLength * rWidth
       //Display the Answers
       display "The Perimeter of " + sPoolName + " is " + rPerimeter + " meters, and its Area is " + rArea +
               " square meters."
   end //main method
end //end of the class
```

Activity 8: Convert Java to pseudo code

a) On your way to and from school, you cover a certain distance in kilometres. Enter the distance you cover each day and then convert this distance to meters and centimetres. The Java code provided below already performs this calculation. Translate it to back to an algorithm in pseudo code. Add comments to your algorithm. Provided is a screenshot of the output:

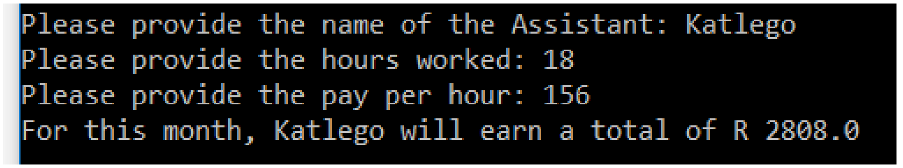


```
How many Kilometers did you cover today? 14
Today you covered an equivalent of 14000 m
This is also equivalent to 1400000 cm
```

```
import java.util.Scanner;
public class ConvertDistance
{ //start class
      public static void main(String[] args)
      { //start main method
            Scanner keyboard = new Scanner(System.in);
            //declare variables
            int  iKilometer;
            int iMeter;
            int iCentimeter;
            //Input
            System.out.print("How many Kilometers did you cover today? ");
            iKilometer = keyboard.nextInt();
            //Processing
            iMeter = iKilometer * 1000;
            iCentimeter = iMeter * 100;
            //Output
            System.out.println("Today you covered an equivalent of " + iMeter + " m");
            System.out.println("This is also equivalent to " + iCentimeter + " cm");
      } //end main method
} //end of the class
```

b) Student assistants in all TUT libraries are paid according to the number of hours they work in a month. As a finance official, you are expected to read in the assistant's name, hours they worked as well as their pay rate; then calculate and display their salary. The Java code provided below already performsthis calculation. Translate it to back to an algorithm in pseudo code; add comments to your algorithm. Provided is a screenshot of the console window:

```
import java.util.Scanner; // Import the java.util package
public class CalculateSalary      { //Start class
   public static void main(String[] args) { //start main method
      Scanner keyboard = new Scanner(System.in);
      //Declare variables
      String sAssistantName;
      double rSalary, rRate;
      int iHourWorked;
      //Input
      System.out.print("Please provide the name of the Assistant: ");
      sAssistantName = keyboard.nextLine();
      System.out.print("Please provide the hours worked: ");
      iHourWorked = keyboard.nextInt();
      System.out.print("Please provide the pay per hour: ");
      rRate = keyboard.nextDouble();
      //Calculate the salary
      rSalary = iHourWorked * rRate;
      //Display the Salary
      System.out.println("For this month + " + sAssistantName + " will earn a total of R " + rSalary);
   } //end main method
} //end of the class
```



```
Please provide the name of the Assistant: Katlego
Please provide the hours worked: 18
Please provide the pay per hour: 156
For this month, Katlego will earn a total of R 2808.0
```

Activity 9: Knowledge checklist

Use the following checklist and ensure you can correctly define or explain the following concepts:

	✓ / ?		✓ / ?
Constructor method.		Object attributes.	
Dot-notation		Object Oriented Programming (OOP)	
Import a class		Package	
Instance method		Scanner class	
Instance of a class		Static / anonymous / class method	
Instantiation		System object	
Method scope		Value-returning methods	
Object		void methods	

Activity 10: Skills checklist

Use the following checklist to ensure you are able to do the following:

	✓ / ?
Import the Java scanner class from its relevant package	
Create an object based on the Scanner class to read values from the keyboard.	
Call the relevant methods from an object based on the Scanner class to read variables of specific data types from the keyboard.	
Declare a variable of a specific data type.	
Prompt users for input and perform appropriate calculations based on the inputs provided by a user	
Convert pseudo code statements into executable Java code	
Convert Java code to correct and understandable pseudo code	

3.5 Primitive data types

You have already declared variables and indicated data types for them, and you only declared variables of type int, double and String. The table below contains more detail about the purpose of each of the 8 *primitive data types* available in Java.

- "A primitive data type specifies the size and type of variable values, and it has no additional methods" (w2schools.com, 2020).
- "A primitive type is predefined by the language and is named by a reserved keyword" (Oracle, 2019). As a programmer one of your tasks is to determine
    - what *type* of values will be stored in each variable, and
    - what the *range* of the values will be
    - what *precision* will be needed.

The range of the values will help you to determine which data type will use the least amount of memory space. For example, if you have avariable that will store the grade of a learner (a value between 1 and 12), there is no need to declare it as a data type long that will use 8 bytes in memory. Rather use a data type byte that will only require 1 byte (8 bits).

String

Type	**Number of bytes**		**Use**
double	8 (64 bits)	Floating-point numbers	Double-precision, floating-point numbers range from -1.7E308 to 1.7E308 with up to 16 significant digits. (A double-precision number uses twice as many bits as a single-precision number. Therefore,it is called *double*-precision.)
float	4 (32 bits)		Single-precision, floating-point numbers from -3.4E38 to 3.4E38 with up to 7 significant digits. (This is the regular/standard floating-point number. Therefore, it is called *single* precision.)
long	8	Whole numbers	Long integers from -9 223 372 036 854 775 808 to 9 223 372 036 854 775 807
int	4		Integers from -2 147 483 648 to 2 147 483 647
short	2		Short integers from -32 768 to 32767
byte	1		Very short integers from -128 to 127
char	2 (16 bits)		A single Unicode character that is stored in two bytes. An example of a character is the letter 'A', or the symbol '*'. (Note: When you assign a character value to a variable, you use single quotes.)
boolean			A *true* or *false* value.

Note the following

- Significant digits:

  Numbers in real life applications are mostly a measure of something, e.g. km between home and campus, GB memory-storage on your phone, characters in a tweet, weight of an atom, etc. The digits of a number that express a magnitude to a specified degree of accuracy are very important. They are sometimes also called *significant* figures (or significant digits or decimal places). Usually the number is rounded off. In the process of rounding numbers, you are required to round to an appropriate number of significant digits.

  When a lotto advert said the estimated jackpot for Saturday (April 4th) was R6 million, this number has been rounded to one significant figure (National Lottery, 2020). The actual value however was R5 996 533.86, which has nine significant figures. Which value is more accurate?

  Definition: The significant figures of a number are digits that carry meaning contributing to how precise and accurate the measurement is.

  The following table contains rules for significant figures (Columbia University, n.d.)

1. All non-zero numbers **ARE** significant.	The number 33.2 has **THREE** significant figures because all the digits present are non-zero.
2. Zeros between two non-zero digits **ARE** significant.	2051 has **FOUR** significant figures. The zero is between a 2 and a 5.
3. Leading zeros are **NOT** significant.	They are nothing more than place holders. The number 0.54 has only **TWO** significant figures. 0.0032 also has **TWO** significant figures. All the zeros are leading.
4. Trailing zeros to the right of the decimal **ARE** significant.	There are **FOUR** significant figures in 92.00. The value 92.00 is different from 92: a scientist who measures 92.00 milliliters knows his value to the nearest 1/100th milliliter; meanwhile his colleague who measured 92 milliliters only knows his value to the nearest 1 milliliter. It is important to understand that "zero" does not mean "nothing." Zero denotes actual information, just like any other number. You cannot tag on zeros that are not certain to belong there.
5. Trailing zeros in a whole number with the decimal shown **ARE** significant.	Placing a decimal at the end of a number is usually not done. By convention, however, this decimal indicates a significant zero. For example, 540. indicates that the trailing zero IS significant; there are THREE significant figures in this value.

6. Trailing zeros in a whole number with no decimal shown are **NOT** significant.	Writing just 540 indicates that the zero is NOT significant, and there are only TWO significant figures in this value.
7. Exact numbers have an **INFINITE** number of significant figures.	This rule applies to numbers that are definitions. For example, 1 meter = 1.00 meters = 1.0000 meters = 1.0000000000000000000 meters, etc.
8. For a number in scientific notation: **N** x**10**^{y}, all digits comprising **N ARE** significantby the first 6 rules; **10** and **y** are **NOT** significant.	5.02 x 10^4 has **THREE** significant figures: 5.02. 10 and 4 are not significant.

Rule 8 provides the opportunity to change the number of significant figures in a value by manipulating its form. For example, let us try writing 1100 with THREE significant figures. Before you read further – what is your answer?

By rule 6, 1100 has TWO significant figures; its two trailing zeros are not significant. If we add a decimal to the end, we have 1100., with FOUR significant figures (by rule 5.) But by writing it in scientific notation: 1.10 x 10^3, we create a THREE-significant- figure value.

- A *floating-point* number is a number with a fractional part, containing decimal values. (Note, a value such as 45.0 is also a floating-point number, even though it does not have a value after the decimal point. In computer programming, the value 45.0 is stored in a different way than the value 45.) The term *float* is used because there is no fixed number of digits before and after the decimal point – the decimal point can 'float'. For example, 123.456; 1.23456; 12345.6 are all floating-point numbers.
- The term *real* value can also be used instead of floating-point number.
- When you write a program, you can use scientific notation to express the value of a floating-point number (you learned how to write real (floating-point) values in scientific notation in Computational Mathematics, Chapter 1).

  For example, the following values are all equivalent.

Mathematical value	Scientific notation	Value in Java code
238.2	2.382 x 10^2	2.382E2
0.000456	4.56 x 10^{-4}	4.56E-4

- String is not a primitive data type. It is a class. In this module you will not notice the difference between a primitive data type, and a class, since we will use String values as single values only. We will discuss how methods from the String class can be used to manipulate values later.

Activity 11: Determine the best data type

Which data type is the most appropriate for the following variables values?

Variables' descriptions	Data types						
	byte	short	int	long	float	double	char
The interest rate is 13.50							
5000 registered students							
The number of meters is 1274367851							
The number of tickets sold is 5							
The change is R45.50							
The answer is the square root of 500							
The class group is H							
The weight is 10							

3.6 Naming rules and conventions

When programmers develop a program, or a programming language, there are many things that need to have names. For example, classes, variables, methods, objects. These names are called identifiers. Some rules for identifiers are specific to the syntax of programming languages. There are also conventions programmer follow (common practice) – rules they agree to follow to make programs easier to read and interpret.

3.6.1 *Rules for identifier names*

- Identifier may not start with a digit.
- It may only contain letters, digits (0 to 9), and the underscore character (_).
- An identifier name may not be a Java keyword. For example, class is not a valid identifier.

3.6.2 *Conventions for identifier names*

- Start a variable name with a lowercase character and let every new word in the name start with an uppercase letter. For example, topSpeed; numberOfLearners. Note – in this module we follow the additional convention to let the first letter be an indication of the data type stored in a variable. For example, rTopSpeed; iNumberOfLearners; sName.
- Start the name of a class with an uppercase letter.
- Use meaningful variable names that are easy to remember.
- Remember that Java is case-sensitive. Do not use variable names such as iNUM, and iNum in the same program. They represent different values and may make the program confusing to follow.
- Let the name of a constant consist of only uppercase letters e.g. TAX_PERC.

Activity 12: Valid / suitable identifiers

In the following activity, you need to indicate whether an identifier is valid, and whether it is suitable. If your answer is no, you need to motivate why, and suggest a better option. The first two variables are examples of what your answers should look like.

Type of identifier	Name	Valid ? ✓/✗	motivation & suggestion	Suitable ✓ /✗	motivation & suggestion
class	costTickets	✓		✗	The convention is to start the name of a class with a capital letter.
variable	1_Price	✗	May not start with a digit. Valid name is rPrice_1	If an identifier is not valid, the suitability is not relevant. You can just write N/A in both columns.	N/A
variable	iNum&Litres				
variable	iHours				
Class	Hours				
Class	Class				
variable	price 2				
variable	payRate				
methods	print Ln				
methods	nextInt				
class	rands				

3.7 Declare and initialize variables

You can declare and assign initial values to variables in different ways.

Use two statements to declare and initialize a variable			

`type variableName; //declare`
`variableName = value; //initialize`			

	`int iCounter;`
	`iCounter = 1;`
		`double rTotal;`
		`rTotal = 5.6;`
			`String sName;`
			`sName = "James";`

Use one statement to declare and initialize a variable at the same time			
`type variableName = value;`			

	`int iCounter = 1;`
	`double rTotal = 5.6;`
	`String sName = "James";`
	`int iAge = keyboard.nextInt();`		

These are more examples of the way values can be assigned to variables of different types.

```
int iCounter = 10; double rTotal = 25.99;

double rCost = 5.0;

float rFinal = 9.345F;          //You must add the letter F to indicate it is data type float

long iNumBytes = 20000L;        //You must add the letter L to indicate it is data type long

double rDistance = 2.5E+8;      //Scientific notation

char cClass = 'D';              //To assign a value to a character, you use single quotes

char cRegisterClass = 68;       //This is valid -the Unicode value of the letter D is 68

boolean bValid = false;         //false is a keyword

int a = 8, b = 9;               //You can initialize two values in one statement
String sPassWord = "James123*";
```

Enrichment: The following web sites provide information about assigning hexadecimal, octal and binary literal values to variables
https://data-flair.training/blogs/literals-in-java/
https://www.geeksforgeeks.org/literals-in-java/
https://riptutorial.com/java/example/6654/hexadecimal--octal-and-binary-literals

Activity 13: Declare and initialize variables

1. Translate the following lines into Java code:
   a) Declare an integer variable called iDistance with an initial value 100.
   b) Declare and initialize a double variable called rValue to 1.406.
   c) Declare a Boolean variable bAnswer and set the value of the variable to true.
2. Which of the following is an *in*valid assignment statement?
   a) iTotal=9;
   b) 75.00=price;
   c) cAns='N';
   d) iSum=150;
3. Fix all the errors in the following code:

```
public class Test
{
pablic statics void main(string[] args)
{
   iAge = 52 ;
   int iGift = iAge * 5
   System.Out.printline("Your gift is R" + Gift);
}
}
```

3.8 Numeric operators

The following is an assignment statement (a programming instruction).

```
double rAverage = (67 + rTest2 + 58) / 3;
```

- A value will be *assigned* to the *variable* rAverage.
- The *data type* of the variable rAverage is double, and only one statement is used to declare the variable and assign a value to it.
- The = sign is an assignment operator.
- (67 + rTest2 + 58) / 3 is an *arithmetic expression*.
- The *arithmetic operators* used in this expression is + and /.
- The *operands* in this expression are 67; rTest2; 48 and 3 – so there are four operands in the expression.
- Some of the operands are *literals* (a value that appears directly in a program) such as 67; 58 and 3.
- One of the operands is a *variable* (rTest2).
- Operators such as + and – work on two operands. Such operators are called *arithmetic binary operators*. For example, the expression 6 + 4 has two operands, so the plus operator works on two operands.

Operators for numeric data types include standard arithmetic operators:

- Addition +
- Subtraction -
- Multiplication *
- Division /
- Remainder or Modulus %

You wrote several programs in Scratch. You will now write many of the same activities in Java. These activities use the +; - ; * and / arithmetic binary operators.

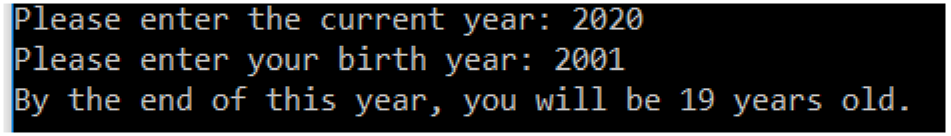
Activity 14: Calculate age this year (Refer to activity 16 in Unit 2) (Class name CalculateAge)

Ask the user for the current year and the user's year of birth, and then calculate and tell the user how old he or she will be in the current year

Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	**Variable name**	**What value will be stored in thevariable**	**Is it an input/intermediate /output variable**

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.



```
Please enter the current year: 2020
Please enter your birth year: 2001
By the end of this year, you will be 19 years old.
```

3.9 Constant values (final)

In Scratch you wrote programs where certain values will stay constant in the program. For example, VAT will stay 15% throughout a program, or the cost of one pen will be R20.00.

In Java, the syntax is as follows:

```
final type CONSTANT_NAME = value;
```

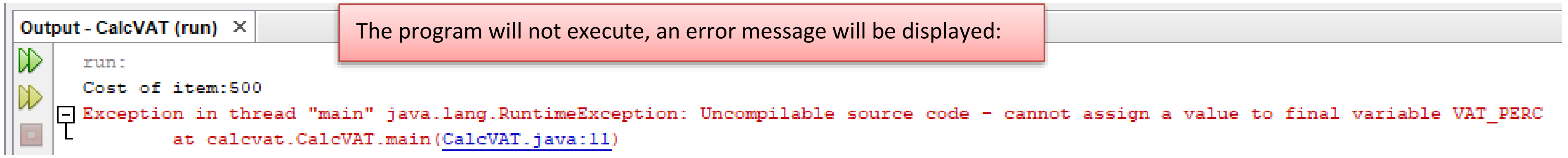
```
final double VAT = 0.15;
final String PASSWORD_ADMIN = "HiBaby123*";
final double DISTANCE = 3.65e+9;
final int PERC_DISC = 10;
```

Information about constant values

- Use all capital letters for constant values, and separate words using an underscore character.
- The value of a constant cannot be changed later in the program.
- Advantages of using constant values:
    - You do not have to repeatedly type the same value if it is used multiple times.
    - If the value needs to change (e.g. VAT changes to 16%), you only have to change it once at the beginning of the program.
- A descriptive name makes a program easier to read.
- In Scratch, a constant value was just a variable, a value was assigned to the variable at the beginning of the program, and we used capital letters to indicate it is a constant value. In Java, the program will not execute if you declare a constant value, and then add code to change the value of the constant.

Look at the following Java code:

```java
import java.util.Scanner;
public class CalcVAT {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        final double VAT_PERC = 0.15;
        double rCost, rVatAmt, rFinal;
        System.out.print("Cost of item:");
        rCost = input.nextDouble();
        VAT_PERC = 0.16;
        rVatAmt = VAT_PERC * rCost;
        rFinal = rCost + rVatAmt;
        System.out.println("The amount of VAT to be paid is R" + rVatAmt);
        System.out.println("The final amoount is R" + rFinal);
    }
```


Output - CalcVAT (run)

The program will not execute, an error message will be displayed:

```
run:
Cost of item:500
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - cannot assign a value to final variable VAT_PERC
        at calcvat.CalcVAT.main(CalcVAT.java:11)
```

In pseudo code a constant value can be used as follows:

```
CashRegister // the class name
  begin //start of the class
  main method
  begin //main method
    //declare a constant value
    PERC_DISC = 10
    //declare variables
    int: iNumItems
    double: rPriceOne, rCost, rDiscount, rTotalAmount
    // Input
    display "Number of items: "
    enter iNumItems
    display "Cost of one item: R"
    enter rPriceOne
    // Processing
    rCost = iNumItems * rPriceOne
    rDiscount = PERC_DISC / 100 * rCost
    rTotalAmount = rcost - rDiscount
    // Output
    display "The discount amount is: R" + rDiscount
    display "The  total amount to pay is: R" + rTotalAmount
  end //main method
end //end of the class
```

The Java code looks as follows:

```
import java.util.Scanner;
public class CashRegister
{ // begin class
  public static void main(String[] args)
  { // begin main method
    Scanner kb = new Scanner(System.in);
    //declare a constant value
    final double PERC_DISC = 10;
    //declare variables
    int iNumItems;
    double rPriceOne, rCost, rDiscount, rTotalAmount;
    // Input
    System.out.print("Number of items: ");
    iNumItems = kb.nextInt();
    System.out.print("Cost of one item: R ");
    rPriceOne = kb.nextDouble();
   //Processing
    rCost = iNumItems * rPriceOne;
    rDiscount = PERC_DISC / 100 * rCost;
    rTotalAmount = rCost - rDiscount;
   //Output
    System.out.println("The discount amount is: R" + rDiscount);
    System.out.println("The total amount to pay is: R" + rTotalAmount);
  } // end main method
}  //end class
```

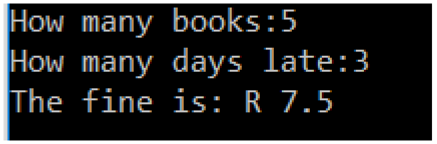
Activity 15: Library fine (Refer to activity 21 in Unit 2) (Class name LibraryFine)

A library charges a fine of 50c per day per book if a book is returned late. Write pseudo code using valid variable names to read the number of books and the number of days they have been returned late, then calculate and display the fine in Rand.

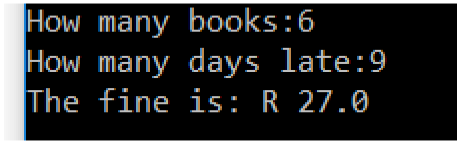
Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	Variable name	What value will be stored in the variable	Is it an input/intermediate /output variable

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.



```
How many books:5
How many days late:3
The fine is: R 7.5
```



```
How many books:6
How many days late:9
The fine is: R 27.0
```

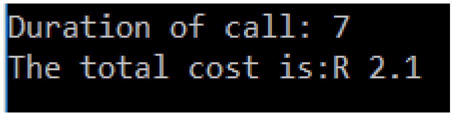
Activity 16: Calculate phone call cost (Refer to activity 24 in Unit 2) (Class name PhoneCallCost)

The first three minutes of a phone call costs 50c per minute. The remaining minutes (everything more than 3 minutes), costs 15c per minute. Write pseudo code using valid variable names to calculate the total cost of a phone call when the duration of the call is provided.Assume the user will never enter a value less than 3.

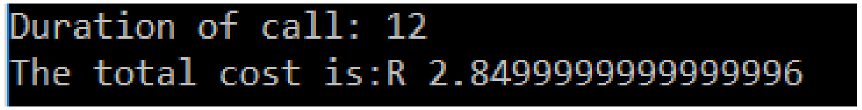
Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	Variable name	What value will be stored in thevariable	Is it an input/intermediate /output variable

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.



```
Duration of call: 7
The total cost is:R 2.1
```



```
Duration of call: 12
The total cost is:R 2.8499999999999996
```

You will learn how to display only two decimal values later.

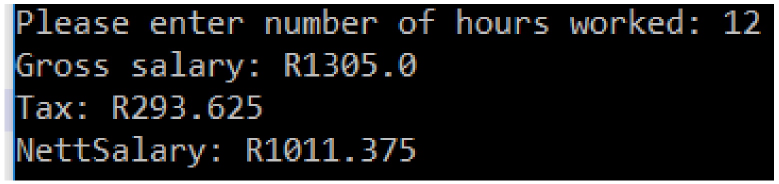
Activity 17: Calculate wages (Refer to Unit 2 activity 26) (class name Calculate Wages)

William is a temporary worker and receives a wage of R100 per hour for the first 5 hours worked but for any additional hours the hourly wage is increased by 15%. He must pay tax at a rate of 22.5% of his gross income. The hours worked by him must be entered (e.g. 6.8). Assume he will work more than 5 hours. Calculate his gross income, tax to be paid, and net income. See the example output for the valuesyou need to display. Use as many constant values as possible.

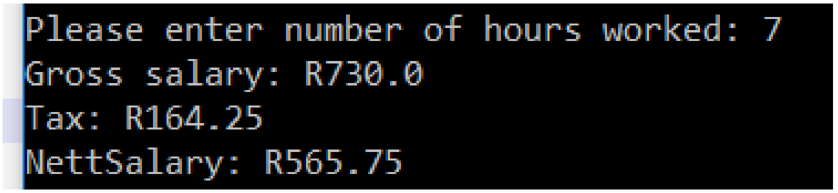
Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	Variable name	What value will be stored in the variable	Is it an input/intermediate / output variable or a constant

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.



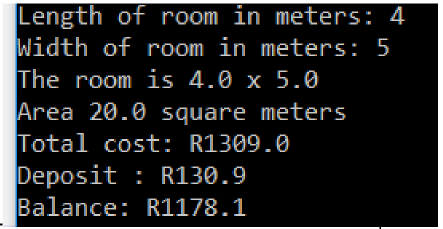
```
Please enter number of hours worked: 12
Gross salary: R1305.0
Tax: R293.625
NettSalary: R1011.375
```



```
Please enter number of hours worked: 7
Gross salary: R730.0
Tax: R164.25
NettSalary: R565.75
```

Activity 18: Contractor fees (Refer to Unit 2 Activity 27) (class name ContractorFees)

A contractor lays floor tiles at a tariff of R65.45 per square meter (m^2). Calculate the total cost to tile a room of a certain length and widthin meter (as specified by the user). The contractor requires a deposit of 10% of the cost prior to commencing, and the balance on completion of the work. Calculate the deposit as well as the balance. Display the area, cost, deposit and the balance. Use constant values where relevant.



```
Length of room in meters: 4
Width of room in meters: 5
The room is 4.0 x 5.0
Area 20.0 square meters
Total cost: R1309.0
Deposit : R130.9
Balance: R1178.1
```

Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	**Variable name**	**What value will be stored in thevariable**	**Is it an input/intermediate / output variable or a constant**

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.

Activity 19: Food cost (class name EatALot)

(This activity was adapted from (Pretorius & Erasmus, 2021, p. 45)

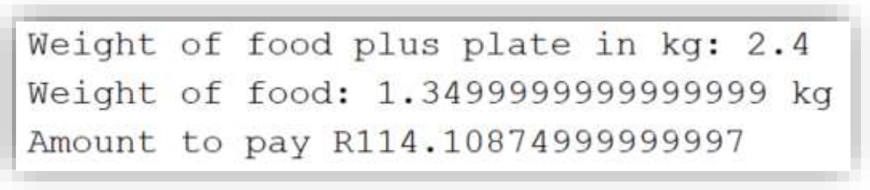
At the Eat-a-Lot restaurant, customers dish up their ownfood and have the plate weighed (in kg). The total weight of the plate with the food in it must be entered. The plate weighs 1.05 kg, which must be subtracted from the total weight to determine the weight of the food only. The customer has to pay R7.35 per 100g of food, and 15% VAT must be added to the total amount.

Plan and write a Java program to determine the amount of money the customer should pay. The program should display the weight of the food, as well as the amount due.

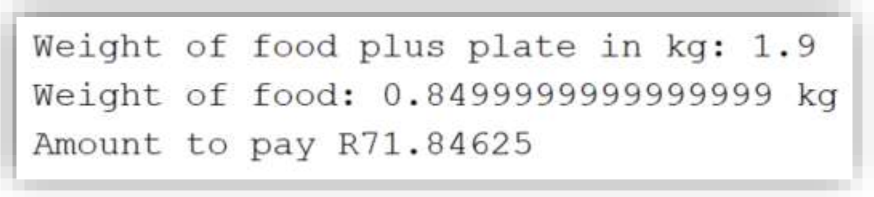
Use the following constant values:

```
    final double PLATE = 1.05; //plate weighs 1.05 kg
    final double FOOD_UNIT = 100.0; //cost is calculated per 100g food
    final double COST_PER_UNIT = 7.35; //R7.35 per 100 g
    final double VAT_PERC = 0.15; //VAT % in South Africa in 2021
```

These are examples of output:



```
Weight of food plus plate in kg: 2.4
Weight of food: 1.3499999999999999 kg
Amount to pay R114.10874999999997
```



```
Weight of food plus plate in kg: 1.9
Weight of food: 0.8499999999999999 kg
Amount to pay R71.84625
```

Because of the way decial values are stored internally, many decimal numbers are displayed. Later in this unit you will learn how to use methods from the Math class to to round values to a certain number of decimal digits.

Activity 20: Knowledge checklist

Use the following checklist and ensure you can correctly define or explain the following concepts:

	✓ / ?		✓ / ?
Assign		Intermediate variable	
Assignment operator		Intermediate variable	
Assignment statement		Literals	
Binary operator		Numeric operators	
Constant value		Operands	
Convention		Precision	
Data type		Primitive data type	
Declare		Range	
Double		Real	
Floating-point		Scientific notation	
Identifiers		Significant digit	
Initialize		Unicode	

Activity 21: Skills checklist

Use the checklist to ensure you can do the following:

	✓ / ?
Determine/choose the best data type for a variable	
Name variables	
Name classes	

Declare variables	
Assign values to variables	
Initialize variables	
Use an IPO table to plan a program	
Write mathematical solutions using actual values, then translate it to an algorithm / pseudo code	

3.10 More about coding calculations

Programmers often have to write code to do arithmetic calculations. In the examples and activities you did so far, the calculations were relatively simple, for example:

```
rPercentage = rScore/iTestTotal * 100;

rCost = iNumTickets * 200;

iAge = iThisYear – iBirthYear;

rFine = iNumBooks * iNumDays * 0.5;
```

You noticed that all calculations should be typed on one line. Therefore, a mathematical calculation such as

$rPercentage = \frac{rScore}{iTestTotal} * 100$ had to be written as `rPercentage = rScore/iTestTotal * 100;`

The following are examples of mathematical equations converted to Java assignment statements. Note that mathematical equations use single letters for variables, where in programming we usually use descriptive variable names that follow certain rules.

Note:

- We know ab in algebra means a * b. However, when we write programming code, we must supply all operators.
- Calculations in parentheses are processed first.

Java does not have an integer division operator.(In Computational Mathematics we used the symbol $\backslash$ or $\lfloor \quad \rfloor$).

Java has only one division operator **/**

If both operands are integer, then Java does integer division. For example, 5/9 is interpreted as 5\9. Therefore, 5/9 = 0. However, $5.0/9 = 0.5555555556$

Similarly, $1/2 = 0$, but $1.0/2 = 0.5$

Mathematical equation	Java assignment statement
$x = a(b + c)$	`iResult = iA * (iB + iC);`
$C = \frac{5}{9}(F - 32)$	`rCelcius = 5.0/9 * (rFahr-32);`

$A = \frac{1}{2}bh$	`rAreaTriang = 1.0/2 * rBase * rHeight;`
	`rAreaTriang = 0.5 * rBase * rHeight;`

$A = \frac{h}{2}(s1 + s2)$	`rAreaTrapez = rHeight/2 * (rSide1 + rSide2);`
$y = mx + c$	`rYCoord= rGradient * rXvalue + rConstant;`
$x = \frac{a + b}{c + d} - 2ab$	`rX = (rA + rB)/(rC + rD) - 2 * rA * rB;` Note: you must add the parentheses in the Java code.

Remember: Whenever you see a division with more than one term as numerator and/or more than one term as denominator, it actually means the terms above the division line are in brackets. Therefore, you can write the previous equation as follows:

$$x = \frac{(a + b)}{(c + d)} - 2ab$$

Then it is easier to realize how to write the Java statement

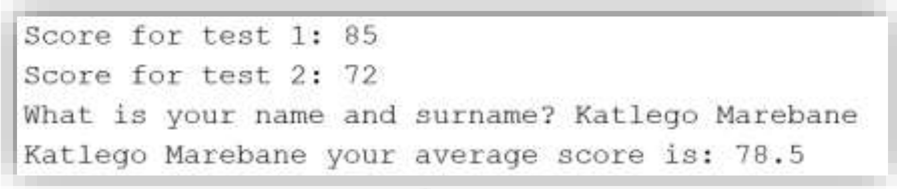
Activity 22: Translate Math to Java and *vice versa*

	Mathematical equation	Java assignment statement
a)	$\text{perIncrease} = \frac{\text{new}-\text{old}}{\text{old}} \times 100$	
b)		`rResult = 2*(rX + 3*rY) / (2*rX - 3*rY);`
c)	$\text{result} = 2a - \frac{ab}{a+b} \div \frac{b}{b-1}$	
d)	$\text{ageA} = 3(\text{AgeB} - 10)$	
e)	$\text{weeklyWage} = \text{dailyWage} \times \text{noDays}$	
f)		`rZ = 2*rA * (rA * rB - rA / (rB - 2));`
g)		`rZ = rX / (2 * rX - 3 * rY) + 6 * rY;`
h)	$z = 3xy + \frac{xy}{y-2y} - 2y$	

Problem with the nextLine() method

Most of the input values we read in activities and examples so far were numeric values (integer and floating-point). When we did read a String value – such as the user's name – it was the first input value we read. However, when you use the nextLine() method after having read number values, you will experience a problem. Look at the following code:

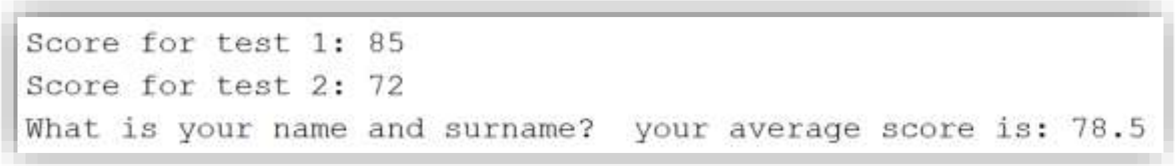
```
package calcaveragenl;
import java.util.Scanner;
public class CalcAverageNL {
        public static void main(String[] args){
           Scanner keyboard = new Scanner(System.in);
           int iTest1, iTest2;
           double rAvg;
           String sNameSurn;
           System.out.print("Score for test 1: ");
           iTest1 = keyboard.nextInt();
           System.out.print("Score for test 2: ");
           iTest2 = keyboard.nextInt();
           System.out.print("What is your name and surname? ");
           sNameSurn = keyboard.nextLine();
           rAvg = (iTest1 + iTest2) / 2.0;
           System.out.println(sNameSurn + " your average score is: " + rAvg);
        } //end main method
} //end class
```



```
Score for test 1: 85
Score for test 2: 72
What is your name and surname? Katlego Marebane
Katlego Marebane your average score is: 78.5
```

This is the output you would expect.

This is the output you get.



```
Score for test 1: 85
Score for test 2: 72
What is your name and surname?  your average score is: 78.5
```

What happened? Why did the program not give the user the opportunity to enter a name?

Here is the explanation:

When a user pushes the <Enter> key, a new-line character is stored in an area of the computer's memory called the *keyboard buffer*.
When the value for iTest1 was entered by the user, the value 85 as well as a new-line character was stored in the keyboard buffer.

85 <new-line>

The nextInt() method copies the value 85 from the keyboard buffer to the variable iTest1, but the new-line character was left behind in the buffer.

<new-line>

The next input was the value 72. The keyboard buffer now looked like this:

<new-line> 72 <new-line>

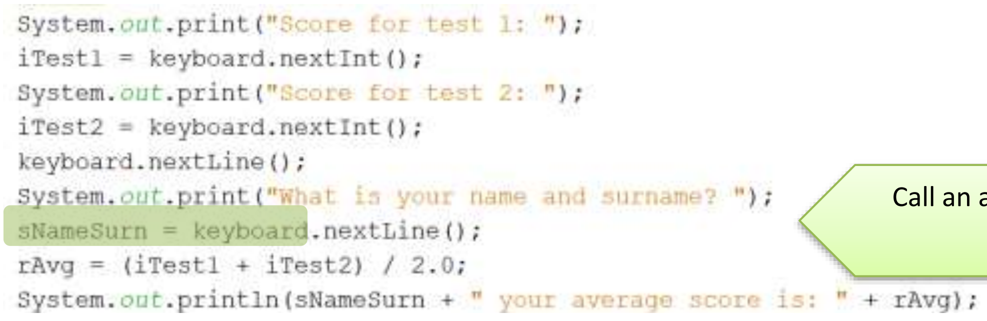
The nextInt() method knows it should ignore the new-line character, it only accepts characters that can be part of an integer. So it ignores the first new-line character, and copies the value 72 from the buffer to the variable iTest2. The new-line character following the 72 stays in the buffer.

<new-line>

When the nextLine() method is called, it checks the keyboard buffer and finds a new-line character. The nextLine() method has been programmed to copy all the characters in front of the new-line character in the buffer and transfer it to a variable (in this case sName). However, there are no characters in front of the new-line character, so an empty string is transferred to the variable sName, and the user does not get the opportunity to enter a name and surname.

Here is the solution:

You need to add a statement after the second integer value was read to remove the additional new-line character. The code should therefore look like this:



```
System.out.print("Score for test 1: ");
iTest1 = keyboard.nextInt();
System.out.print("Score for test 2: ");
iTest2 = keyboard.nextInt();
keyboard.nextLine();
System.out.print("What is your name and surname? ");
sNameSurn = keyboard.nextLine();
rAvg = (iTest1 + iTest2) / 2.0;
System.out.println(sNameSurn + " your average score is: " + rAvg);
```

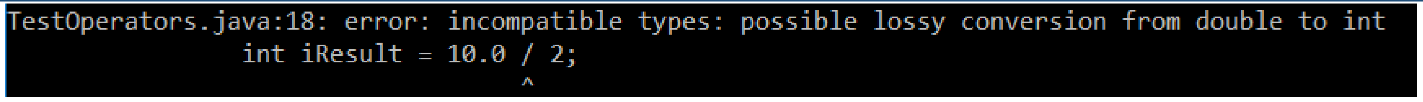
Call an additional nextLine() method after the last nextInt() method to remove the <new-line> character.

3.11 The division operator in Java

The division binary operator in Java works as follows:

Operator	Name	Description	Example

/	Division	• Divides the right operand into the left operand.
		• If both operands are integer, the result will be integer.
		• If one or both operands are a floating-point value, the result will be a floating-point value.
			`int iResult = 12 / 6;          //iResult = 2`
			
			`int iResult = 12 / 5;          // iResult = 2`
			`double rResult = 12.0 / 6;     // rResult = 2.0`
			`double rResult = 12 / 5.0;     // rResult = 2.4`
			`int iResult = 10.0 / 2;        //Error message`



```
TestOperators.java:18: error: incompatible types: possible lossy conversion from double to int
                int iResult = 10.0 / 2;
                                   ^
```

Terminology

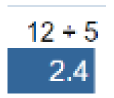
Lossy conversion is simply the loss of information while handling data.

In Java, it corresponds to the possibility of losing the value or precision of a variable while converting one type to another.

When we try to assign a variable of large-sized type to a smaller sized type, Java will generate an error, *incompatible types: possible lossy conversion,* while compiling the code. (Baeldung, 2019)

In the paragraph *Converting data types* you will learn how to prevent these types of errors.

3.12 Doing integer division in Java

Mathematics notation: Integer division	
When you want to indicate that an integer value is required as answer when doing division, the following notation can be used: $12 \backslash 5$ or $\left\lfloor \frac{12}{5} \right\rfloor$	$12 \backslash 5 = 2$ $\lfloor 12/5 \rfloor = 2$
Calculating the answer of integer division	
To determine the result of integer division, first do the calculation 12/5 on your pocket calculator. The result of the division will be 2.4. The result of *integer division* is the number before the decimal point. Ignore the fraction.	 12 ÷ 5 2.4
Pseudo code notation: Integer division	
When planning a program, you can use the following notation to indicate that integer division is required: `iAnswer = rValue1 \ iValue2` or $\text{iAnswer} = \left\lfloor \frac{\text{rValue1}}{\text{iValue2}} \right\rfloor$	

Java code: Integer division when both operands are integer values	
When both operands are integer, a Java division operator will provide an integer answer, disregarding any decimal value. For example: `int iAnswer = iValue1 / iValue2;` if iValue1 = 12 and iValue2 = 5, iAnswer will contain the value 2 after the division operation.	
Java code: Integer division when one or both operands are floating-point values	
Additional Java code is needed to remove the decimal part or to obtain an integer answer after division with floating-point values.	
Method 1: Disregard the decimal part of the result after division	**Method 2: Obtain an integer value after division**

Use the floor() method from the Math class
`double rAnswer;`
`double rValue1 = 12.5, rValue2 = 4.3;`
`rAnswer = Math.floor(rValue1 / rValue2);`
`rAnswer = 2.0`
	Use *explicit casting* to indicate that an integer value should be provided.
	`int iAnswer;`
	`double rValue1 = 12.5, rValue2 = 4.3;`
	`iAnswer = (int) (rValue1 / rValue2);`
	`iAnswer = 2`

Note Methods from the Math class as well as explicit casting will be discussed in more detail later in this unit.	

3.13 The remainder operator in Java

Operator	Name	Description	Example

%	Remainder (MOD)	Returns the value that is left over after dividing the right operand into the left operand.	`int iResult = 13 % 5;          // iResult = 3`
			`double rResult = 13 % 5.0;     // result = 3.0`

The % operator is also called the *modulus* or *MOD* operator.			`int iResult = 5 % 7;           // iResult = 5`
			`int iResult = 14 % 2;          // iResult = 0`
			`double rResult = 8 % 2.5;      // rResult = 0.5`
			`double rResult = 2 % 5.5;      // rResult = 2.0`
			`double rResult = 10 % 2.5;     // rResult = 0.0`

Mathematics notation: Remainder (MOD)

When you want to indicate that the modulus (remainder after a division operation) should be calculated, the following notation can be used:

12 MOD 7 or 12 % 7

For example:

12 MOD 7 = 5

12 % 7 = 5

In these notes the symbol % will be used in pseudo code and planning, since it is the symbol used in Java.

Pseudo code notation: MOD

When planning a program, you can use the following notation to indicate that the remainder after a division operation should be calculated:

```
remainder = value1 % value2
```

Calculating the answer when using the remainder (MOD) operator

	Example: Calculate 12 % 7 (the answer is 5) Step 1: Start with integer division `12 \ 7 = 1`	
Method 1	Then write the value 12 as follows: `12 = 1 * 7 + 5`	If you multiply 1 by 7, you need to add 5 to get a total of 12, so 5 is the remainder.
Method 2	In primary school you learned to do write the result of division as follows: $12 \div 7 = 1\ rem\ 5$	If you write the division operation in this way, it is clear that 5 is the remainder.
Method 3	Looking at the values, you can also calculate the remainder as follows: `remainder = 12 - 1 * 7`	

You say:

- 12 mod 7 is 5.
- 12 divided by 7 is 1 remaining 5.
- 7 goes into 12 once, and 5 remains.

Now you can write an algorithm to calculate the answer to the % operator. Let us say you need to calculate the following:

`firstValue % secondValue`

Algorithm in pseudo code	`iIntDivAns = firstValue \ secondValue`
	`remainder = firstValue - iIntDivAns * secondValue`

	Using the algorithm you just developed, you can calculate the remainder after division for any values. The values can be any combination of floating-point numbers, integer values, positive or negative.

	`34 % 6`
	`iIntDivAns = 34 \ 6 = 5`
	`remainder = 34 – 5 * 6 = 34 – 30 = 4`

	`4 % 9`
	`iIntDivAns = 4 \ 9 = 0`
	`remainder = 4 – 0 * 9 = 4 – 0 = 4`

	`12 % 4`
	`iIntDivAns = 12 \ 4 = 3`
	`remainder = 12 – 3 * 4 = 12 – 12 = 0`

	`19 % (-5)`
	`iIntDivAns = 19 \ (-5) = -3`
	`remainder = 19 – (-3) * (-5) = 19 – 15 = 4`

	`8 % 2.5`
	`iIntDivAns = 8 \ 2.5 = 3`
	`remainder = 8 – 3 * 2.5 = 8 – 7.5 = 0.5`

	`15.8 % 3.3`
	`iIntDivAns = 15.8 \ 3.3 = 4`
	`remainder = 15.8 – 4 * 3.3 = 15.8 – 13.2 = 2.6`

	`0 % 5`
	`iIntDivAns = 0 \ 5 = 0`
	`remainder = 0 – 0 * 5 = 0 – 0 = 0`

Java code: MOD

Here are examples of Java code to determine the remainder after division:
`double rRem = rVal1 % rVal2;`
`int iRem = iVal1 % iVal2;`

Activity 23: Predict results of \ and MOD operators

1. Fill in the blanks:

`int result = 10 / 3;`	`// result will be`	`int iNumSweets = 35;`
		`int iNumChildren = 4;`
		`int iNumEach = iNumSweets / iNumchildren;`
		`int iNumLeft = iNumSweets % iNumchildren;`
		
		`iNumEach =`
		
		`iNumLeft =`

`int result = 10 % 3;`	`// result will be`	
`int result = -10 / 3;`	`// result will be`	
`int result = 3 % 10;`	`// result will be`	

`int result = 11 % 3 % 2;`	`// result will be`	`int iMinutes = 112;`
		`int iHours = iMinutes / 60;`
		`int iRemMinutes = iMinutes % 60;`
		
		`iHours =`
		
		`iRemMinutes =`

`double result = 10.0 / 5 / 4;`	`// result will be`	
`double result = 1.5 / 0.5;`	`// result will be`	
`double result = 12 % 2.5;`	`// result will be`	

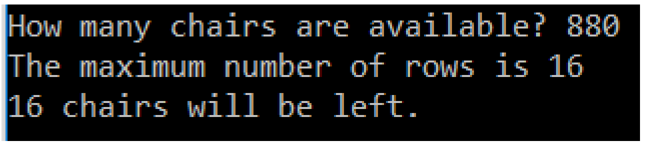
2. Predict the exact output of the following program:

```
public class CapeToCairo { //start class
  public static void main(String[] args) { //start main method
    int iTime = 588224;                   // time in seconds, to drive from Cape Town to Cairo
    int iSeconds = iTime % 60;            // iSeconds will be_______________seconds
    iTime = iTime / 60;                   // iTime will be____________minutes
    int iMinutes = iTime % 60;            // iMinutes will be________minutes
    iTime = iTime / 60;                   // iTime will be__________hours
    int iHours = iTime % 60;              // iHours will be__________hours
    int iDays = iTime / 24;               // iDays will be___________days

    System.out.println("Driving from Cape Town to Cairo would take " + iDays + "days " + iHours + "hours " +
                                        iMinutes + "min and " + iSeconds + "sec.");
  } //end main method
} //end class
```

The following two activities use integer division and the remainder (mod) operator.

Activity 24: Chairs in student centre (Refer to Unit 2 Activity 32) (class ChairsStudentCentre)

TUT has a certain number of chairs available to place in the student centre for a concert. Write an algorithm to calculate the maximumnumber of rows of 54 chairs each that can be filled with chairs. Calculate how many chairs will be left. (The number of available chairs must be read from the keyboard.) Use constant values where possible.



```
How many chairs are available? 880
The maximum number of rows is 16
16 chairs will be left.
```

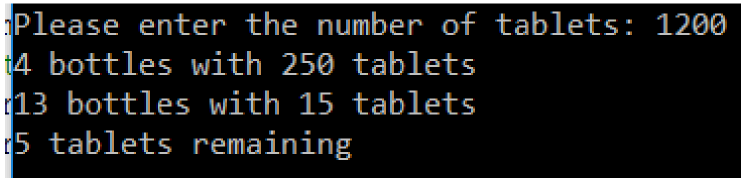
Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	Variable name	What value will be stored in the variable	Is it an input/intermediate / output variable or a constant

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.

Activity 25: Tablets in bottles (Refer to Unit 2 Activity 31) (class name TabletsInBottle)

A medicine factory has a certain number of tablets (read the number of tablets from the keyboard). They have two types of bottles. The one can hold 250 tablets, and the other 15 tablets. They want to fill all the bottles that can contain 250 tablets first, and then place the remaining tablets in the bottles that can hold 15 tablets. Calculate and display how many bottles of each size can be packed, and how many tablets will remain. Use constant values where possible.



```
Please enter the number of tablets: 1200
4 bottles with 250 tablets
13 bottles with 15 tablets
5 tablets remaining
```

Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	Variable name	What value will be stored in the variable	Is it an input/intermediate / output variable or a constant

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.

Activity 26: Knowledge checklist

Use the following checklist and ensure you can correctly define or explain the following concepts:

	✓ / ?
Lossy conversion	
Incompatible types	
Remainder	

Activity 27: Skills checklist

Use the checklist to ensure you can do the following:

	✓ / ?
Predict results of division calculations with integer operands	
Predict results of MOD calculations for integer and real values	
Design programming solutions that require floating-point division, MOD and integer division.	
Translate / convert mathematical equations to Java assignment statements and vice versa	
Correctly read values using the nextLine() method after having read numerical values	
Declare and initialize a constant according to acceptable convention	

3.14 The char data type

3.14.1 *Storing characters – ASCII and Unicode*

Devices such as smartphones, tablets, and computers store data in digital formats that can be handled by electronic circuitry. Everything on a computer is represented as streams of binary numbers, 1s and 0s. Audio, images, and characters are all translated into machine code (simple binary codes that hardware understands).

Characters – such as the letters 'A', 'B', and special characters such as '*', '?' - are internally represented as numbers in computers. Several different encoding schemes have been invented to associate printable as well as many non-printable characters with numbers.

One such encoding scheme is called ASCII (American Standard Code for Information Interchange) – it is pronounced "AS-kee". ASCII code makes provision for 256 different characters to be stored.

When 8 bits are grouped together it is called a *byte*.

- A byte can represent any number between 0 (represented in bits as 00000000) and 255 (represented in bits as 11111111).
- Therefore, there are 256 different values that can be stored in a byte. (The number of different values can be calculated by using the formula 2^N, where N is the number of bits, and N = 8 gives $2^8 = 256$.)

Most ASCII characters are printable characters of the alphabet such as *abc, ABC, 123, ?&!,* etc. The others are control characters such as *carriage return*, *escape*, *tab*, etc., as shown in the ASCII table below.

Character encoding means the conversion of a symbol into a binary number and using a "character map" to read the binary numberas a type of letter. (Ishida, 2015)

Since 256 characters is not enough to provide for the needs of the wide variety of languages spoken by humans. Therefore, the ASCII coding system (that uses 8 bits to store a character), has been extended to make provision for more characters.

This extended standard for encoding characters is called *Unicode* (Universal Code Character Set). It is a single standard that provides a unique number for every character, no matter what platform, device, application, or language.

Unicode was created to unify all the encoding schemes so that confusion between computers could be limited as much as possible. Unicode uses a variable bit encoding program called Unicode Transformation (UTF), where you can choose between UTF-8, UTF-16, and UTF-32-bit encodings. This means it can accommodate a huge number of characters, and currently contains most written languages and still has room for even more. This includes even right-to-left scripts like Arabic, Chinese, Egyptian-hieroglyphics, Emoji and the many other variants are also represented within Unicode. To maintain compatibility with ASCII, Unicode was designed in such a way that the first seven bits matched that of the ASCII chart. The Unicode standard was initially designed using 16 bits to encode characters, but writing sixteen 1s and 0s can be tedious, so hexadecimal was used. Therefore lowercase 'h' character with decimal value of 104, is written as

U+0068 (or 0x0068). Java provides support for Unicode, hence the size of the character data type (char) in Java is 2 bytes (= 16 bits), and range is 0 to 65535 (2^{16}).

This is important to know

- Java stored characters internally as 16-bit integers.
- The encoding scheme called Unicode determines which integer value represents each character.
- ASCII can be seen as a subset of Unicode. You should be able to look up the ASCII value for a specific character in the ASCII table.

ASCII characters and associated decimal value (ASCII-Table, 2020)

Decimal	Symbol		Decimal	Symbol		Decimal	Symbol		Decimal	Symbol	
0	NUL	Null	32	space	Space	64	@	At sign	96	`	Grave / accent
1	SOH	Start of Header	33	!	Exclamation mark	65	A	Capital A	97	a	Small a
2	STX	Start of Text	34	"	Double quote	66	B	Capital B	98	b	Small b
3	ETX	End of Text	35	#	Number	67	C	Capital C	99	c	Small c

Decimal	Symbol		Decimal	Symbol		Decimal	Symbol		Decimal	Symbol	
4	EOT	End of Transmission	36	$	Dollar sign	68	D	Capital D	100	d	Small d
5	ENQ	Enquiry	37	%	Percent	69	E	Capital E	101	e	Small e
6	ACK	Acknowledge	38	&	Ampersand	70	F	Capital F	102	f	Small f
7	BEL	Bell	39	'	Single quote	71	G	Capital G	103	g	Small g
8	BS	Backspace	40	(	Left parenthesis	72	H	Capital H	104	h	Small h
9	HT	Horizontal Tab	41	)	Right parenthesis	73	I	Capital I	105	i	Small i
10	LF	Line Feed	42	*	Asterisk	74	J	Capital J	106	j	Small j
11	VT	Vertical Tab	43	+	Plus	75	K	Capital K	107	k	Small k
12	FF	Form Feed	44	,	Comma	76	L	Capital L	108	l	Small l
13	CR	Carriage Return	45	-	Minus	77	M	Capital M	109	m	Small m
14	SO	Shift Out	46	.	Period	78	N	Capital N	110	n	Small n
15	SI	Shift In	47	/	Slash	79	O	Capital O	111	o	Small o
16	DLE	Data Link Escape	48	0	Zero	80	P	Capital P	112	p	Small p
17	DC1	Device Control 1	49	1	One	81	Q	Capital Q	113	q	Small q
18	DC2	Device Control 2	50	2	Two	82	R	Capital R	114	r	Small r
19	DC3	Device Control 3	51	3	Three	83	S	Capital S	115	s	Small s
20	DC4	Device Control 4	52	4	Four	84	T	Capital T	116	t	Small t
21	NAK	Negative Acknowledge	53	5	Five	85	U	Capital U	117	u	Small u
22	SYN	Synchronize	54	6	Six	86	V	Capital V	118	v	Small v
23	ETB	End of Transmission Block	55	7	Seven	87	W	Capital W	119	w	Small w
24	CAN	Cancel	56	8	Eight	88	X	Capital X	120	x	Small x
25	EM	End of Medium	57	9	Nine	89	Y	Capital Y	121	y	Small y
26	SUB	Substitute	58	:	Colon	90	Z	Capital Z	122	z	Small z
27	ESC	Escape	59	;	Semicolon	91	[	Left square bracket	123	{	Left curly bracket
28	FS	File Separator	60	<	Less than	92	\	Backslash	124	|	Vertical bar
29	GS	Group Separator	61	=	Equality sign	93	]	Right square bracket	125	}	Right curly bracket

Decimal	Symbol		Decimal	Symbol		Decimal	Symbol		Decimal	Symbol	
30	RS	Record Separator	62	>	Greater than	94	^	Caret / circumflex	126	~	Tilde
31	US	Unit Separator	63	?	Question mark	95	_	Underscore	127	DEL	Delete

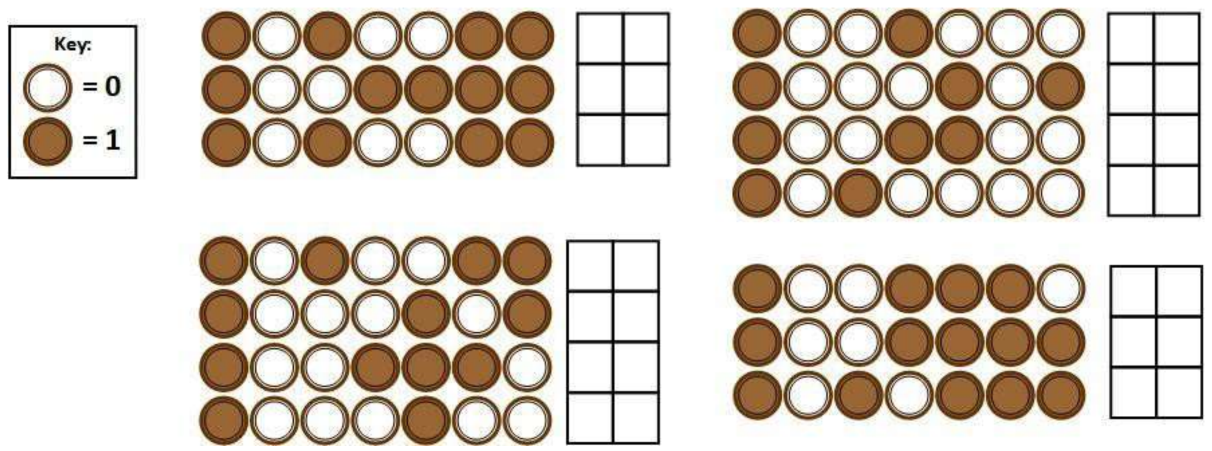
Activity 28: ASCII Decoder

Use the ASCII table to decode the message below. Write down the character below the corresponding decimal value.

84	104	105	115	32	105	115	32	97	32	115	101	99	114	101	116	32	109	101	115	115	97	103	101	33

Activity 29: Coconut puzzle

You are working for a GIS (Geographic Information System) company and while reviewing recent satellite images of an island, you noticed a pattern. A closer look shows coconuts arranged in a way that looks like a binary representation of the ASCII code. What was the message? [Hint: Convert binary to decimal]



Images copied from: (Teachers Pay Teachers, n.d.)

3.14.2 *Working with character values*

Programs using variables of data type character (char) will be written from Unit 4 onwards. In this section we will just explain how a character value can be read from the keyboard, and what the relationship between a character value and a String value is.

The char data type is used to store single characters, such as a class group 'C' or an answer – 'Y' or 'N'.

Declare a variable of datatype char.	`char cClassGroup;`	
Assign a character value (a literal value) to a character variable.	`cClassGroup = 'C';`	Note: Single quotes are used when a character value is assigned.

What is the relationship between a character and a String value?

String values are lists of characters. Each character in a string value is numbered by the programming language to indicate the character's position in the string. It can be illustrated by a diagram – the string value stored in the variable sName, containing the name "Katlego" is interpreted as follows:

sName

[0]	[1]	[2]	[3]	[4]	[5]	[6]
K	a	t	l	e	g	o

The character 'K' is stored in the first position of the string value sName – Java numbers the characters starting at 0 (zero). We say "the *index* of the first letter is 0".

Therefore – sName[0] has the value 'K'.

sName[1] has the value 'a'.

Read a character value from the keyboard.	The Scanner class does not have a method to read a character from the keyboard. To obtain a character from the keyboard, you need to read a String value, then use the charAt() method of the String class to obtain the first character (with index 0) from the string that was read.

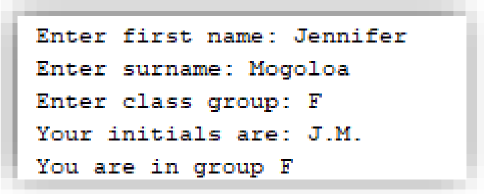
`System.out.print("Enter class group: ");`
`String sClassGroup = kb.next();`
`cClassGroup = sClassGroup.charAt(0);`
	Read a string from the keyboard, use the charAt(0) method to obtain the first character.

OR	

`System.out.print("Enter class group: ");`
`cClassGroup = kb.next().charAt(0);`
	Use the charAt() method on the string returned by the next() method.

Activity 30: Display initials (class DispInit)

Write a Java program that will read a student's first name, then the surname, as well as the student's class group. The program should then display the initials and the class group of the student.



```
Enter first name: Jennifer
Enter surname: Mogoloa
Enter class group: F
Your initials are: J.M.
You are in group F
```

3.15 Data compatibility

As discussed before, different data types use a different number of bytes to store values. This influences the range and the precision of the value that can be stored in a variable. As a result, a data value of one data type can not just be assigned to a variable of another data type in a program. For example, the following assignment statements are valid:

`short shortValue;`
`byte byteValue = 2;`
`shortValue = byteValue;`
	A short data type stores values in 2 bytes with a range of -32 768 to 32767.
	A byte data type stores values in 1 byte with a range of -128 to 127
	Therefore a short data type and a byte data type are *compatible*.

The following assignment statements are not valid, the data types are *not compatible*:

`int intValue;`
`double dblValue = 12.5;`
`intValue = dblValue;`
	An int data type stores values in 4 bytes with a range of -2 147 483 648 to 2 147 483 647.
	A double data type stores values in 8 bytes with a range from -1.7E308 to 1.7E308 with up to 16 significant digits.
	Such an assignment statement will not be allowed, and the Java compiler will display an error message:

```
Exception in thread "main" java.lang.
RuntimeException: Uncompilable source code - incompatible types: possible lossy conversion from double to int
```

It is clear that a data type that can contain fractional values and values much larger than an int, cannot be stored in an int datat type.

3.16 Data type conversion

Java does allow certain data types to be converted to others. The rules driving these conversions are based on the data types being ranked. Rank is based on the minimum and maximum sizes (the range) of values that can be stored in a data type. The ranks are as follows:

Data type	# bytes
double	8
float	4
long	8
int	4
char	2
short	2
byte	1

Highest rank
(bigger / wider data types)

Lowest rank
(smaller / narrower data types)

Some data conversions are allowed by Java, and happens automatically. Other conversions have to be done by the programmer by adding additional instructions (code). In the following paragraphs the different options in converting data types will be explained.

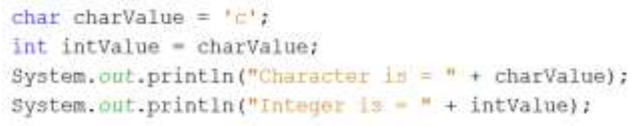
3.16.1 *Widening conversion*

Data type of a lower rank is automatically converted to a data type of a higher rank when a program is compiled. This is called widening conversion because lower ranked data types are smaller (narrower) and they can be converted to higher ranked types (bigger / wider).

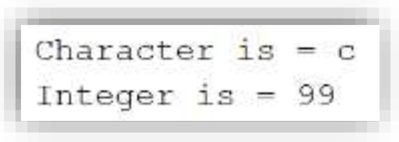
The following are examples of widening conversions:

```
double rValue;
int iValue = 36764324;
rValue = iValue;
```

```
long longValue;
short shortValue = -32700;
longValue = shortValue;
```



```
char charValue = 'c';
int intValue = charValue;
System.out.println("Character is = " + charValue);
System.out.println("Integer is = " + intValue);
```



```
Character is = c
Integer is = 99
```

The variable intValue will contain the ASCII value of the character 'c', that is 99.

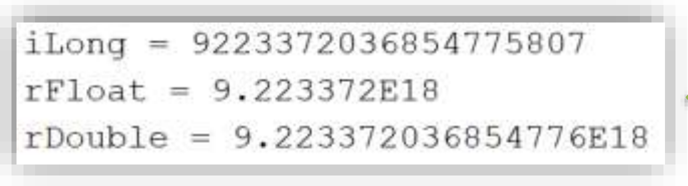
Widening conversion is also called: implicit type conversion / implicit casting / automatic conversion / promotion.

Some widening conversions are allowed by Java, but as a programmer you need to know that it can cause problems. One such example is when you cast a variable of data type long to a variable of data type float or double. According to the ranking of the data types it is allowed. A variable of type float or double can store values of a much wider range than a variable of data type long, the data types are compatible.

However, data type float can only store 7 significant digits, and double 16 significant digits. Look at the result of the code on the next page:

double	8 (64 bits)	Double-precision, floating-point numbers range from -1.7E308 to 1.7E308 with up to 16 significant digits.
float	4 (32 bits)	Single-precision, floating-point numbers from -3.4E38 to 3.4E38 with up to 7 significant digits.
long	8	Long integers from -9 223 372 036 854 775 808 to 9 223 372 036 854 775 807

```java
long iLong = 9223372036854775807L; //The biggest possible long value
float rFloat;
double rDouble;
rFloat = iLong;
rDouble = iLong;
System.out.println("iLong = " + iLong);
System.out.println("rFloat = " + rFloat);
System.out.println("rDouble = " + rDouble);
```



```
iLong = 9223372036854775807
rFloat = 9.223372E18
rDouble = 9.223372036854776E18
```

iLong has 19 significant digits.
rFloat has 7 significant digits
rDouble has 16 significant digits

In this example the value changed for example, from iLong = 9 223 372 036 854 775 807 to rFloat = 9 223 372 000 000 000 000 – quite a different value with loss of precision.

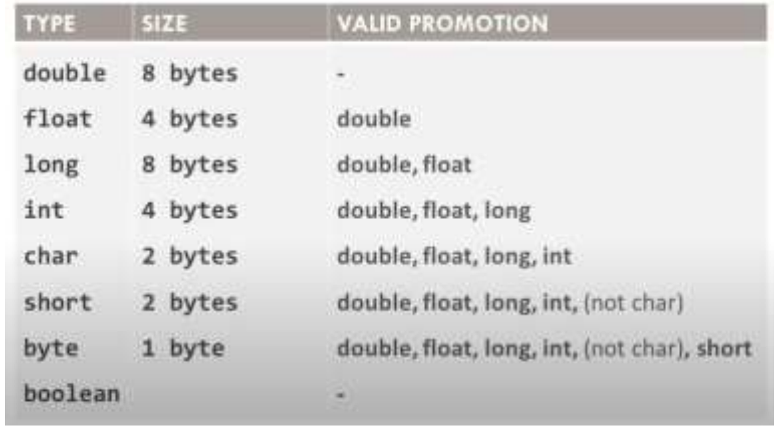
Therefore, as a programmer you need to know that even though Java permits some castings, you may not get the results required or expected since some of the digits will not be available any longer.

The problems relating to precision can be solved by rather using methods from the BigDecimal class to do calculations on larger numbers and numbers that need a large number of significant digits. However, using methods from the BigDecimal class does not form part of the scope of this module.

Another exception in widening conversion is converting data types short and int to char. Even though byte and short are lower ranks than char, a data type of short or byte can not automatically be converted to a char. The reason is because short and byte can contain negative values, and char values are represented by positive values.

The following diagram can be used to determine if widening conversion is allowed. A value with a data type in the left column, can be promoted to the various data types displayed in the last column.

Diagram obtained by making a screenshot from presentation: (Posch, 2013)

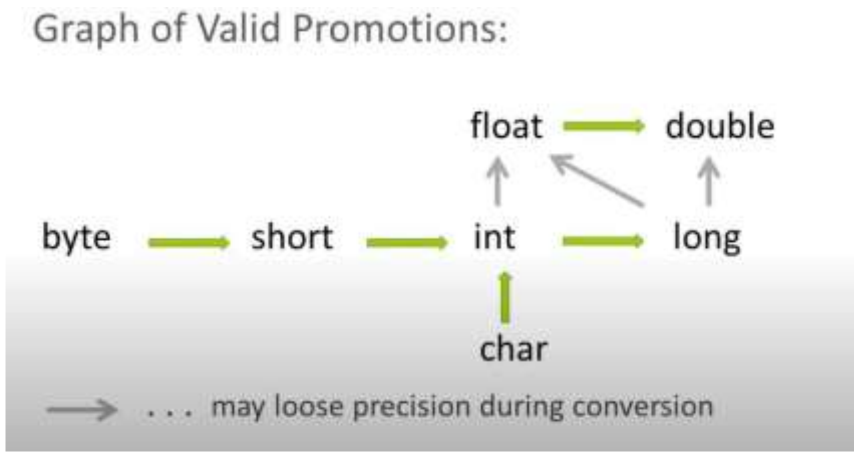


TYPE	SIZE	VALID PROMOTION
double	8 bytes	-
float	4 bytes	double
long	8 bytes	double, float
int	4 bytes	double, float, long
char	2 bytes	double, float, long, int
short	2 bytes	double, float, long, int, (not char)
byte	1 byte	double, float, long, int, (not char), short
boolean		-

In the following diagram, the green arrows (thicker) represent valid promotions, and the grey (thinner) arrows represent promotions that may go along with loss of precision. For example:

- int can be promoted to long.
- int can be promoted to float – but with loss of precision.
- char can be promoted to int, and to long.
- long cannot be promoted to int (no arrow connecting long to int)

Diagram obtained by making a screenshot from presentation: (Posch, 2013)


Graph of Valid Promotions:
. . . may loose precision during conversion

$$\begin{tikzcd}
& & \text{float} \arrow[r] & \text{double} \\
\text{byte} \arrow[r] & \text{short} \arrow[r] & \text{int} \arrow[u] \arrow[r] & \text{long} \arrow[u] \arrow[ul] \\
& & \text{char} \arrow[u] &
\end{tikzcd}$$

Activity 31: Widening conversion

Specify the output and/or data type and/or whether the casting is valid or not.

Operation	Casting valid? Y/N	Data type of output	Output displayed

`int x=7;`
`float y = x/2;`
`System.out.println(y);`			

`float x = 9.2/3;`
`System.out.println(x);`			

`byte x = 27;`
`x = x+1;`
`System.out.println(x);`			

`long x = 21L;`
`double y = x/3.0;`
`System.out.println(y);`			

`long x = 25L;`
`double y = x/5;`
`System.out.println(y);`			

`long x = 200L;`
`System.out.println(x/2.0);`			

`double x = 200;`
`int y = x/2;`
`System.out.println(y);`			

Operation	Casting valid? Y/N	Data type of output	Output displayed

`double x = 200.0;`
`int y = 2;`
`Long z = x/y;`
`System.out.println(z);`			

`long x = 10/5L;`
`byte y = x;`
`System.out.println(x);`			

`double x  =  10.0;`
`float y = x/2.0;`
`System.out.println(x);`			

3.16.2 *Narrowing conversion using a cast operator*

Java does not allow programming code where data from a more precise data type (higher rank) is assigned to a value of a less precise data type (lower rank). When a data value from a high rank is assigned to a data type of lower rank, some data may be lost, therefore Java will not allow it.

However, for some programs there may be valid reasons to want to 'lose' some data. Look at the following example:

```
display "How may litres of sanitizer is in a container? : "
enter rLitresContainer
display "How many litres in a small container?: "
enter rSmallContainer
iNumSmallBottles = rLitresContainer \ rSmallContainer
display "You can fill " + iNumSmallBottles + " small bottles."
```

We need to do integer division. The decimal part of the answer must be disregarded. For example, if a container has 12.5 litre sanitizer, and a small bottle can contain 150 ml (0.15 litre), we can fill

12.5\0.15 = 83 small bottles.

Remember : 12.5/0.15 = 83.3333 , but we want to 'loose' the decimal values.

In this case the programmer needs to use a casting operator to manually convert the data type of a higher rank (double), to a data type of a lower rank (int).

This is what the Java code will look like:

```
package sanitizer;
import java.util.Scanner;
public class Sanitizer {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        double rLitresContainer, rSmallContainer;
        int iNumSmallBottles;
        System.out.print("How may litres of sanitizer is in a container? : ");
        rLitresContainer = keyboard.nextDouble();
        System.out.print("How many litres in a small container?: ");
        rSmallContainer = keyboard.nextDouble();
        iNumSmallBottles = (int) (rLitresContainer / rSmallContainer);
        System.out.println("You can fill " + iNumSmallBottles + " small bottles.");
    }
}
```

A cast operator (int) was used.
A cast operator is a *unary operator* that appears as a data type name enclosed in a set of parentheses.
The (int) operator must be placed before the value being converted.
In this example, the operation rLitresContainer / rSmallContainer must be done first, and then the decimal part of the result must be removed, therefore the calculation itself must be placed in parentheses.

Narrowing conversion is also called: implicit type conversion / explicit casting / explicit type conversion.

The + operator is a binary operator, it requires two operands: iValue1 + iValue2.
Unary operators require only one operand. For example: iAnswer = (int) rAnswer;

Activity 32: Narrowing casting

Operation	Casting valid? Y/N	Data type of output	Output displayed

`double x = 11.9;`
`int y = (int)x;`
`System.out.println(y);`			

`double x  =  11.9;`
`float y = (int)x;`
`System.out.println(y);`			

`double x  =  11.0;`
`float y = (int)x/2;`
`System.out.println(y);`			

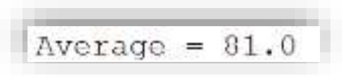
`int x = 11;`
`double y = (float)x/2;`
`System.out.println(y);`			

`int x = 10;`
`float y= (long)(x/5L);`
`System.out.println(y);`			

3.16.3 *Casting integer division to a floating-point answer*

Remember that Java always returns an integer answer when division is done with both integer operands. For example:

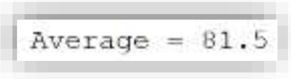
```
int iTest1 = 88, iTest2 = 75;
double rAvg = (iTest1 + iTest2)/2;
System.out.println("Average = " + rAvg);
```



```
Average = 81.0
```

In this case, (88 + 75) / 2 = 81.5, and you may want to round the value up, so you want to keep the decimal part. In this case you can use casting to ensure a floating-point answer. All the following code segments will provide correct results for this scenario:

```
double rAvg = (double) (iTest1 + iTest2) / 2;
double rAvg = ( (double) iTest1 + iTest2) / 2;
double rAvg = (iTest1 + (double) iTest2)/ 2;
```



```
Average = 81.5
```

If one of the operands in the calculation is a floating-point, all the other operands will be casted as a floating-point number. As the programmer, you need to explicitly cast one operand, Java will implicitly cast the other operands.

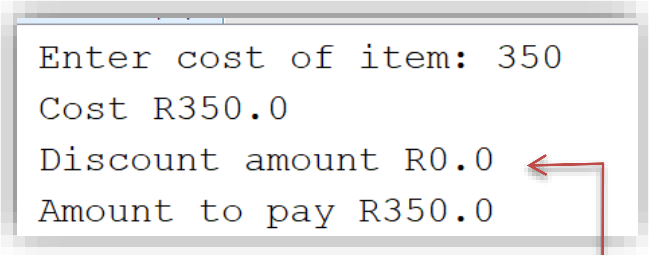
The following calcualtion is not casting, but it will also give the required result.

```
double rAvg = (iTest1 + iTest2) / 2.0;
```

Important

Look out for the following problem that may occur in programs if you forget about the way Java handles integer division. The following code is provided, with the resulting output:

```
import java.util.Scanner;
public class DiscountError {
    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        double rCost, rAmtToPay, rDiscAmt;
        final int DISC_PERC = 10;
        System.out.print("Enter cost of item: ");
        rCost = kb.nextDouble();
        rDiscAmt = DISC_PERC/100 * rCost;
        rAmtToPay = rCost - rDiscAmt;
        System.out.println("Cost R" + rCost );
        System.out.println("Discount amount R" + rDiscAmt );
        System.out.println("Amount to pay R" + rAmtToPay );
    }
}
```



```
Enter cost of item: 350
Cost R350.0
Discount amount R0.0
Amount to pay R350.0
```

The output you expected was that the discount amount should be R35!

The error occurred because of the way Java handles integer division. The constant value was declared as an integer.

```
final int DISC_PERC = 10;
```

Therefore, the calculation

```
rDiscAmt = DISC_PERC/100 * rCost;
```

is interpreted as: `10/100 * rCost.`

The value `10/100 = 0` since both operands are integer values.

Solutions:

Declare the constant value as a double.

```
final double DISC_PERC = 10.0;
```

or

```
rDiscAmt = DISC_PERC/100.0 * rCost;
```

or

```
rDiscAmt = (double) DISC_PERC/100 * rCost;
```

Activity 33: Number of 85 g packets (Refer to Unit 2 Activity 34) (class NumSmallPackets)

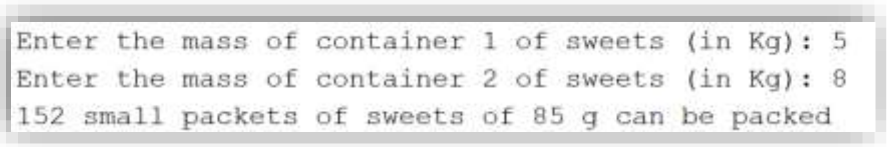
A school receives two containers of sweets, each containing a certain kilogram of sweets (read the mass of each from the keyboard).How many small packets of sweets of 85g each can be packed using the sweets from both containers?

Use constant values where possible. Note: A container can weigh, for example, 5.5 kg – a double value.

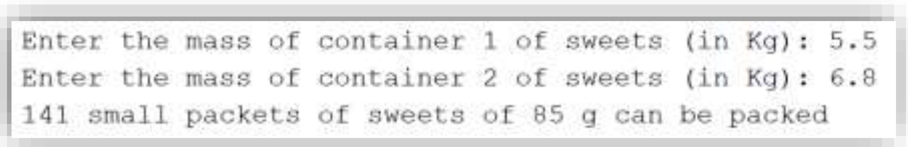
Write pseudo code to solve this problem. Complete the following table while you write the pseudo code:

Data type of variable	Variable name	What value will be stored in the variable	Is it an input/intermediate / output variable or a constant

Write a Java program to implement the pseudo code. Use the numbers in the example output as test data.



```
Enter the mass of container 1 of sweets (in Kg): 5
Enter the mass of container 2 of sweets (in Kg): 8
152 small packets of sweets of 85 g can be packed
```



```
Enter the mass of container 1 of sweets (in Kg): 5.5
Enter the mass of container 2 of sweets (in Kg): 6.8
141 small packets of sweets of 85 g can be packed
```

Activity 34: Knowledge checklist

Use the following checklist and ensure you can correctly define or explain the following concepts:

	✓ / ?
ASCII character	
Bits	
Bytes	
Casting	
Compatible	
Higher rank	
Loss of precision	
Lower rank	
Narrowing conversion *(explicit casting / explicit type conversion)*	
Precision	
Range	
Valid promotion	
Widening conversion (*implicit casting / implicit type conversion / automatic conversion / promotion)*	

Activity 35: Skills checklist

Use the checklist to ensure you are able to do the following:

	✓ / ?
Predict whether a casting operation is valid or not.	
Predict the resulting value of a variable when a casting operation was performed.	

Choose the appropriate type of casting (implicit or explicit) when writing Java code.	
Identify possible problems with implicit casting and suggest a solution to the problem.	
Write Java programs where widening and/or narrowing casting should be implemented to obtain the required results.	
Indicate when promotion may result in loss of precision.	

3.17 Mathematical calculations

3.17.1 *Methods from the Math class*

The Math class provides several standard mathematical methods. The class is located in the package `java.lang.Math`, but it is provided automatically and does not require an import statement. All the methods are static – therefore, you do not have to create an object based on the Math class, you use the *class name* to call a method. For example: Math.round().

Most of the methods in the Match class return values. In some programming languages methods that return values are called *functions*. Each of these methods needs certain arguments (data they receive that they use to calculate a return value). They return a value of a certain data type.

A summary of the methods you are most likely to use in writing programs for this module follows:

Rounding methods	Return data type	Description
Math.floor()	double	`int iPencilsEach = (int) Math.floor(iNumPencils \ iNumLeaners);` The argument is rounded down to its nearest integer, returned as a double value. In the example above, the (int) cast operator must be used to cast the double value that will be returned by the Math.floor method to an int value.
Math.ceil()	double	`int iNumTins = (int) Math.ceil(rLitrePaint/iLitreInTin);` The argument is rounded up to its nearest integer. In the example above, the (int) cast operator must be used to cast the double value that will be returned by the Math.floor method to an int value.

Math.round()	int if argument is float
	long if argument is double
		`int iValue1 = Math.round(3.6f); //returns the value 4`
		`long iValue2 = Math.round(12.5); //returns the value 13`

Exponent methods	**Return data type**	**Description**	

Math.sqrt()	double	`double rRoot = Math.sqrt(25); //returns the value 5.0`
		`double rExp = Math.pow(3, 4); //returns the value 81.0 =` 3^4
		Returns the square root of the argument.	

Math.pow()	double	`Double rExp = Math.pow(4.2, 2); //returns the value 17.64 =` $(4.2)^2$
		Returns the value of the 1st argument raised to the power of the 2nd argument.	

MIN Max and abs methods	**Return data type**	**Description**	

Math.min()	Can return an int, long, float or double, depending on the data type of the argument(s).	`Math.min(2, 3) returns 2`
		`Math.min(2, 3.5) returns 2.0`	

Math.max()		`Math.max(2, 3) returns 3`
		`Math.max(2.5, 3) returns 3.0`	

Math.abs()		`Math.abs(-2.5) returns 2.0`
		`Math.abs(-6) returns 6`
			The abs() method returns a non-negative value. It disregards the sign of a value.

Trigonometric methods	**Return data type**	**Description**	
Math.cos()	double	Returns the trigonometric cosine of an angle in radians. The function has one compulsory parameter – an angle (in radians).	
Math.sin()	double	Returns the trigonometric sine of an angle in radians. The function has one compulsory parameter – an angle (in radians).	
Math.tan()	double	Returns the trigonometric tangent of an angle in radians. The function has one compulsory parameter – an angle (in radians).	
The random() method	**Return data type**	**Description**	

Math.random()	double	Returns a pseudorandom number of data type double within a range that has the following limit: $0 \leq x < 1.0$
		`double x = Math.random(); // 0 ≤ x < 1.0`
		`double x = Math.random() * 25; //0 ≤ x < 25.0`
		`double x = 7 + Math.random() * 58; // 7 ≤ x < 65.0`	

The nextInt() method of the Random class can also be called to generate a random number.		
The nextInt() method	**Return data type**	**Description**

import java.util.Random;
Random randomizer = new Random();
//create an object based on the Random class
randomizer.NextInt()
	int	The nextInt() method from the Random class will return a pseudorandom integer value. The return value is an integer between 0 (inclusive) and the value supplied as an argument (exclusive).
		`int iRandValue = randomizer.NextInt(15);`
		0 ≤ iRandValue < 15

Math.PI is a predefined named constant value in the Math class. The constant has the value 3.14159265358979323846 which approximates the mathematical value ∏. It can be called in mathematical calculations, for example:

```
double rArea = Math.PI * Math.pow(rRadius,2);
```

3.17.2 *Rounding a value to a specific number of decimal places*

As you already noticed, the Math.round() method only rounds the number to the nearest integer. There are cases where you need to round your value to specific number of places, for example, a monetary value (currency) is normally rounded to 2 decimal places. Below is a technique you can use for such exceptional cases. Given the following value:

```
double x = 61.356345296325;
```

To round to 1 decimal place

You multiply the value you want to be rounded by 10 (take note of parentheses), then divide by 10.0 (outside of parentheses). Take note - if you leave out the ".0" your result will be an integer.

```
x = Math.round(x*10)/10.0;
System.out.println(x); //output will be 61.4
```

You will notice that the number of zeros used in the multiplication and division, determines the number of decimal places. So, to round to 3 decimal places you will multiply by 1000 and divide by 1000.0 and so on.

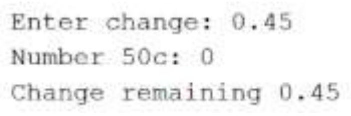
Round to 2 decimal places

```
x = Math.round(x*100)/100.0;
System.out.println(x); //output will be 61.36
```

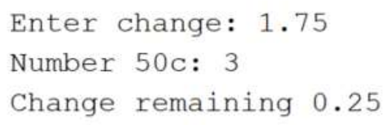
Activity 36: Doing mathematical calculations

a)	class RoomCost
You are a first-year student at University and are looking for a place to stay. Write a program that calculates the amount you must pay for your room, provided that the landlord charges R50.00 per m^2. Provide the answer displayed as currency (rounded to two decimals).	
b)	class ConvertPounds
Leo is a visitor from the UK and wants to know how much his money is worth in South Africa. Write a program that asks the amount Leo has in pounds and converts it to Rands. (1 pound = R22). Provide the answer displayed as currency (rounded to two decimals).	
c)	class HighestMark
Write a program that reads your two test marks and display your highest mark.	
d)	class AvgThree
Write a program that calculates an average of your 3 test marks, then rounds up your average to the nearest integer.	
e)	class FlashDrives
Your lecturer has 48 USB Flash Drives that needs to be distributed equally to all students in class, write a program that will read the number of students and calculate how many USB Flash Drives each student should get. (Hint: do not worry about the remaining ones).	
f)	class StudentNumber
Write a program that will generate a 9-digit student number starting with 22. The other 7 digits must be chosen randomly.	
g)	class CircleMeasurements
The formula to calculate the circumference of a circle is: $C=2\pi r$; and the formula to calculate the area is: $A=\pi r^2$. Write a program that will calculate the Area and Circumference of a circle after the user has entered the diameter.	
h)	class BoxesTiles
Company X is busy with renovations. They need to put in new floor tiles in one of their offices. They have tasked you to calculate how many boxes of tiles they should buy if 1 box of tiles can cover $2m^2$. (Remember you cannot buy 2.3 or 2.8 boxes of tiles.) Your program needs to read the length and the width of the office they want to tile.	

i)	class Change50cents
A shop always wants to give change to customers with as many 50c coins as possible. Write a program that will read the amount of change from the keyboard. The program should calculate and display the number of 50c coins that should be handed to the customer, as well as the remaining change.	



```
Enter change: 0.45
Number 50c: 0
Change remaining 0.45
```



```
Enter change: 1.75
Number 50c: 3
Change remaining 0.25
```

3.17.3 *Coding more advanced calculations*

Programmers are often asked to debug or make changes to an existing program. Therefore, they need to be able to interpret calculations written in programming code and translate it 'back' to mathematical formulas and vice versa.

In Computational Mathematics, you learned the order of operations (operator precedence):

- Parentheses ()
- Exponents ^
- Multiply Divide MOD (* / \ %) (from left to right)
- Add Subtract (+ -) (from left to right)

The same rules apply in Java (and all other programming languages and applications such as Excel), but since there are so many methods available in Java to do calculations, we must expand the rules for order of operations for Java.

Note: Exponents and square roots are done through method calls in Java. Therefore, if you have a calculation such as

$$\sqrt{4}^{3} = Math.pow(Math.sqrt(4), 3)$$

the calculation will be done from left to right. So the square root will be calculated first, then the exponent to the power 3.

Therefore, in the Java code, you will use *nested methods*, with theMath.sqrt() method as an argument for the Math.Pow() method.

- Parentheses and Method calls (Note: Exponents in Java is done through a call to the Math.pow() method.)
- Multiply Divide MOD (* / \ %) (from left to right)
- Add Subtract (+ -) (from left to right

The following are examples of mathematical equations and the equivalent pseudo code as well as Java assignment statements.

Mathematical equation	Pseudo code	Java assignment statement
A = (4 + 17) MOD 2-1	A = (4 + 7) MOD 2 -1	A = (4 + 7) % 2 - 1 ;
B = 4(3 + 4)	B = 4 * (3 + 4)	B = 4 * (3 + 4);
$C = \sqrt{25}$	C = 25 ^ (1/2) $C = \sqrt{25}$	C = Math.sqrt(25);
$D = \sqrt[3]{64}$	D = 64 ^ (1/3) $D = \sqrt[3]{64}$	D = Math.pow (64, (1/3.0));
E = K MOD 3 + A + B / C	E = K MOD 3 + A + B / C	E = K % 3 + A + B / C;
F = 3^2 + 4	F = (3 ^ 2) + 4	F = Math.pow (3,2) + 4;
G = 4 + 7 \ 2	G = 4 + 7 \ 2 $G = \lfloor 7/2 \rfloor$	F = 4 + Math.floor(7/2) ;
$H = X^2 + \frac{C}{A} + \frac{C}{2A}^2$	H = (X^2) + (C/A) + (C/(2*A)) ^ 2	H = Math.pow(X,2) + C/A + Math.pow(C/(2*A),2);
$I = A + \sqrt{AB} - 1 + 4\sqrt{B^3}$	$I = A + \sqrt{A * B} - 1 + 4 * \sqrt{B^3}$	I = A + Math.sqrt(A * B) – 1 + 4 * Math.sqrt(Math.pow(B,3));

Activity 37: Convert Math ⇆ Pseudo code ⇆ Java code

Complete the following table by writing a formula in the formats left out.

Mathematical equation	Pseudo code	Java assignment statement
$A = AB + \frac{B-A}{3B} + \frac{C^{(B+2)}}{A-2B}$	A =	A =
B =	B =	B = 175 % 54 % 5 / 2;
C =	C = 3 + 20 MOD 7 * (12 * 9/4)	C =
$D = 2 + 7^2(\sqrt[3]{69 - \frac{5^2}{5\ mod\ 10}})\backslash 2$	D =	D =
E =	E = 167 MOD 12 MOD 2 * 3 * 2 ^ 2 – (5/3)	E =
F =	F =	F = Math.round(Math.pow (3,5) / 4 % 8 /3) ;
G = 15 / 5 * 2 + 30 / 6 + 3	G =	G =
H =	$H = 2 * \sqrt{56}$ ^ 3 mod 17	H =
I =	I =	I = Math.sqrt(7* Math.pow (3,2) + 1) % 3;

3.18 Predicting output of a program

Part of removing all errors from a program is to work through the code and determine what the output of the current code will be. When you work through the logic and syntax of code, you can determine whether changes should be made to the code.

3.18.1 *Substituting values into assignment statements*

You need to be able to translate calculations from mathematics to pseudo code applying the rules of the order of operations and determine the answer of a calculation when specific values are substituted.

Example 1

Calculate the value of S where K = 50, A = 7, B = 6 and C = 3. Show all the calculations.

```
S = K mod 3 + A + B / C
```

Substitute values:

$$
\begin{aligned}
S &= \boxed{50 \bmod 3} + 7 + 6 / 3 \\
&= \underline{2} + 7 + \boxed{6 / 3} \\
&= \boxed{2 + 7} + \underline{2} \\
&= \boxed{\underline{9} + 2} \\
&= \boxed{\underline{11}}
\end{aligned}
$$

The $\boxed{\text{frame}}$ indicates which operation will be done next.

The $\underline{\text{underlined}}$ value is the result of the operation in the previous step.

Example 2

Calculate the value of O where X = 3, C = 4 and A = 2. Show all the calculations.

```
R = X^2 + (C/A) + (C/2*A) ^ 2
```

Substitute values:

$$
\begin{aligned}
R &= \boxed{3 \wedge 2} + (4 / 2) + (4 / 2 * 2) \wedge 2 \\
&= \underline{9} + (\boxed{4 / 2}) + (4 / 2 * 2) \wedge 2 \\
&= 9 + \underline{2} + (\boxed{4 / 2} * 2) \wedge 2 \\
&= 9 + 2 + (\boxed{\underline{2} * 2}) \wedge 2 \\
&= 9 + 2 + \boxed{\underline{4} \wedge 2} \\
&= \boxed{9 + 2} + \underline{16} \\
&= \boxed{\underline{11} + 16} \\
&= 27
\end{aligned}
$$

Example 3

Calculate the value of M where A = 4, B = 16 and C = 256. Show <u>all</u> the calculations.

$$
\begin{aligned}
M &= A + \sqrt{A * C} - 1 + 4 * \sqrt{B}\,\text{^}3 \\
&= 4 + \boxed{\sqrt{4 * 256}} - 1 + 4 * \sqrt{16\text{^}3} \\
&= 4 + \boxed{\sqrt{\underline{1024}}} - 1 + 4 * \sqrt{16}\,\text{^}3 \\
&= 4 + \underline{16} - 1 + 4 * \boxed{\sqrt{16}}\text{^}3 \\
&= 4 + 16 - 1 + 4 * \boxed{\underline{4}\text{^}3} \\
&= 4 + 16 - 1 + \boxed{4 * \underline{64}} \\
&= \boxed{4 + 16} - 1 + \underline{256} \\
&= \boxed{\underline{20} - 1} + 256 \\
&= \boxed{\underline{19} + 256} \\
&= \underline{48}
\end{aligned}
$$

Example 4

Calculate the value of D where W = 34, X = 4, Y = 9 and Z = 5. Show <u>all</u> the calculations.

$$
\begin{aligned}
K \quad &= Z + W \backslash (X + 2 * Y) - X \text{ ^ } 2 + Y \text{ mod } W \\
&= 5 + 34 \backslash (4 + \boxed{2 * 9}) - 4 \text{ ^ } 2 + 9 \text{ mod } 34 \\
&= 5 + 34 \backslash (\boxed{4 + \underline{18}}) - 4 \text{ ^ } 2 + 9 \text{ mod } 34 \\
&= 5 + 34 \backslash \underline{22} - \boxed{4 \text{ ^ } 2} + 9 \text{ mod } 34 \\
&= 5 + \boxed{34 \backslash 22} - \underline{16} + 9 \text{ mod } 34 \\
&= 5 + \underline{1} - 16 + \boxed{9 \text{ mod } 34} \\
&= \boxed{5 + 1} - 16 + \underline{9} \\
&= \boxed{\underline{6} - 16} + 9 \\
&= \boxed{\underline{-10} + 9} \\
&= \underline{-1}
\end{aligned}
$$

Activity 38: Pseudo code format and substitution

a) Calculate the value of g where m = 5, n = 7 and t = 2. Show <u>all</u> the calculations in the correct order. Underline all changes.

```
g = n * (m ^ t - 40)
```

b) Calculate the value of W where A = 5, B = 14, C = 23 and D = 2. Show <u>all</u> the calculations in the correct order. Underline all changes.

```
W = B – (B – A * D) ^ D + A mod C + C \ B
```

c) Calculate the value of k where a = 10, b = 8, c = 4 and d = 30. Show <u>all</u> the calculations in the correct order. Underline all changes.

```
k = a * (b – c) – a mod c ^ 2 * 2 + d \ c
```

d) Calculate the value of m where a = 5, b = 12, c = 6 and d = 3. Show <u>all</u> the calculations in the correct order. Underline all changes.

```
m = a – (b – c) ^ 2 – a mod c * 2 + d \ a
```

e) Rewrite the following equation in pseudo code format and then calculate the value of w if k = 5; y = 17; z = 10 and s = 24. Round the answer to 2 decimals.

Mathematical formula	**Pseudo code**	**Answer (result)**
$w = \frac{kz - y}{y - 6} + \frac{32 + s}{z + 4}$		
$w = (y - z)^{k-3}$		
$w = s - z - 3k^2$		
$w = \sqrt{k^4} + \frac{z}{z - k}$		
$w = k^2 + \frac{10}{\sqrt{z^2}} - s$		
$w = 4z^2 - sy + k$		

Mathematical formula	Pseudo code	Answer (result)
$w = \frac{ks}{z} - \frac{\sqrt{100}}{k}(s)$		
$w = \frac{s}{3} - kz + \frac{sz}{8}$		

Activity 39: Rewrite the following pseudo code statements as mathematical equations:

Pseudo code	Mathematical formula
a) answer = a + b ^ 2 - 5	
b) answer = k – (w + 7) / p + s	
c) answer = a * b – c + a + (c – k) / s ^ m	
d) answer $= 5 \bmod a * 3/\sqrt{c*b}$	
e) answer = k mod 3 + a + b / c	
f) result = p + r – s * g * 5 ^ a	
g) answer $= \sqrt{a \wedge (3+b) + c \wedge 2 \bmod d}$	
h) answer $= \sqrt[6]{a*b\wedge 2 - 2*c\backslash d}$	
i) answer = a \ b mod 5 * d ^ 2	
j) $answer = (2 * a/c) * ((b3/d) / 5) * e$	

3.18.2 *Creating a trace table*

A trace table is a very useful tool to use to predict output of a code segment. It helps you to keep track of the output of eachstatement, thereby testing programs or algorithms for logical errors. It simulates a step by step execution of statements.

Example 1

Consider the following Java code:

```
     int a,b,c,d,e;
1    a = 20;
2    b = 5;
3    c = a * b;
4    d = a + b;
5    e = a - b;

6    System.out.print(c); // prints then put the cursor on the same line
7    System.out.println(d); // prints then put the cursor on a new line
8    System.out.println(e); // prints then put the cursor on a new line
```

A trace table contains several columns.

↓	↓	↓ **Each variable used in the code gets a column.** ↓					↓	*These comments have been added to help you, it is not part of an official trace table.*
Step	**Type of statement**	**a**	**b**	**c**	**d**	**e**	**Display/ Output**	**Comments / Statement**
1	assign	20						a = 20, The value 20 is assigned to a
2	assign		5					b = 5, the value 5 is assigned to b
3	calculation			20 * 5 = 100				c = a * b, the results of the product (a * b) is assigned to c
4	calculation				20 + 5 = 25			d = a + b, the results of the sum (a + b) is assigned to d
5	calculation					20 – 5 = 20		e = a – b, the results of the sum (a + b) is assigned to d
6	output						100	The value of c (100) will be displayed.
7	output						100**25**	The value of d (25) will be displayed on the same line, because the 1st output statement is *print* and **NOT** *println().*

8	output						20	The 2nd output statement was *println()*, so 3rd output will be on a new line.

```
10025
20
```

← Exact output

The value of each variable can be seen in the column belonging to the variable.

Example 2.

Replace the last 3 statements on the above code with the following:

```
6      System.out.println(c); // prints then put the cursor on the new line
7      System.out.print(d); // prints then put the cursor on the same line
8      System.out.println(e); // prints then put the cursor on the same line
```

Step	**Type of statement**	**a**	**b**	**c**	**d**	**e**	**Display**	**Comments / Statement**
				100	25	20		
6	output						100	*println()* used.
7	output						25	The value of d (25) is displayed on the next line.
8	output						25**20**	The last output will be displayed on the same line, because the 2nd output statement is *print()* and **NOT** *println()*.

```
100
2520
```

← Exact output

Example 3

Consider the following pseudo code. Create a trace table to determine the final output of the algorithm. Use a value of 5 as input for variable iNumItems.

```
1.   final int iOneItem = 10
     int iNumItems
2.   display "How many items do you want to buy?"
3.   enter iNumItems
4.   double rCost = iOneItem * iNumItems
5.   rcost = rcost * 0.9     //give 10% discount
6.   display "You bought " + iNumItems + " at R" + iOneItem + " each."
7.   display "After discount you need to pay R" + rCost     //on a new line
```

Note: You can create a trace table for Java code or for pseudo code.

tep	Type of statement	iOneItem	iNumItems	rCost	Display	Comments / Statement
1	assign	10				
2	display				How many items do you want to buy?	
3	enter		5			In the problem statement it was specified that a value of 5 should be used as test value.
4	calculation			10*5=**50**		
5	calculation			50*0.9=**45**		The value of rCost was 50. Now it is re-calculated, so the previous value will be over-written.
6	display				You bought 5 items at R10 each	
7	display				After discount you need to pay R45	

Note that the last value in the column is the correct value of rCost. The value of 50 will not be available any longer, it has been replaced by 45.

```
How many items do you want to buy?
You bought 5 items at R10 each
After discount you need to pay R45
```

← Exact output

Activity 40: 0Predict the output using trace tables

Create trace tables to predict the exact output of each algorithm.

a)
```
ShowOutputA
     A = 5
     B = 7
     C = A * 2 – 1B = B + 2
     display "The value of B = " + B
     display "The value of C = " + C  //on a new line
 end
```

b)
```
ShowOutputB
        a = 7
        b = a ^ (a - 5)c = b + a * 3
        d = a \ b + (b – a) mod 3
        a = c – b * d mod 14
        display a + d
        display (a + d)     //on a new line
 end
```

c)
```
ShowOutputC
        display "Show results:  "
        a = 5
        b = 2
        c = a + b * 2
        b = a * 3 + b ^ 2
        display b                // on a new line
        b = c mod 3
        a = a \ 3 + 2
        display "a is " + a      // on same line as previous output
        display "b = " + b       // on a new line
 end
```

d) ShowOutputD
```
        w = 12
        y = 17
        x = w + y mod 5
        z = y – w / 3 + x
        display "x = " + x + "        z = " + z
 end
```

e) ShowOutputE
```
   X = 15
   Y = 23
   Z = 2
   Y = X * Z \ 7
   W = X mod 8
   display "The value of Y is " + Y
   display "The value of W is " + W  //on a new line
end
```

Activity 41: Skills checklist

Use the checklist to ensure you can do the following:

	✓ / ?
Create solutions where methods from the Math class should be called as part of the solution.	
Create random number calling the nextInt() method of the Random class.	
Round a number to a specific number of decimal places.	
Convert mathematical equations into Java code.	

Substitute values into an expression supplied in pseudo code and apply order of precedence principles to determine the value of the expression.	
Create a trace table to determine the output of an algorithm.	

3.19 Unary operators

Previously we worked with binary operators such as +; - ; *; / and MOD. A binary operator works with two operands, for example:

int iTotal =	iCost	+	10;
	operand	operator	operand

Java has several operators that work with *one operand* and are therefore called *unary* operators.

3.19.1 *The unary minus operator – (negation)*

The negative sign applies the minus operator to an operand; therefore, it changes the sign. For example:

```
int iResult = -5; //The value of iResult is -5
int iResult = -(10 – 20); //The value of iResult is 10
```

The unary minus operator is called the negation sign.

Negation is the act of setting a value to its negative equivalent (Mueller, n.d.).

3.19.2 *The unary plus operator*

When you have a variable with data type char, the unary plus operator can be used to emphasize that the data type of the variable will be converted from char to int.

This is *called unary numeric promotion*. This is an example of the code that you can write:

```
char c = 'c';
int i = +c;  //The variable i will have the value 99
```

Please note: The unary plus operator is not really necessary in this case. The following code will have the same result:

```
char c = 'c';
int i = c;//The variable i will have the value 99
```

However, some programmers prefer to make the promotion more ‘visible’ in their code.

3.19.3 *The increment operator ++*

Typically, the ++ operator is used to increase the operand by 1. The increment operator can be placed before a variable name (prefix), or after a variable name (postfix).

	Use of increment operator	Equivalent code without increment operator
Unary post increment Assign then increment	**Post increment example with one variable**	

	`int iNum = 10;`
	`iNum ++; //iNum = 11`
	Provides same result as
		`int iNum = 10;`
		`iNum = iNum + 1; //iNum = 11`

	Post increment example with two variables	

	`int iNum = 10;`
	`int iResult = iNum++;`
	`//iResult = 10; iNum = 11`
	Provides same result as
		`int iNum = 10;`
		`iResult  = iNum; //1. Assign iResult = 10`
		`iNum = iNum + 1; //2. Increment iNum = 11`

Assignment has precedence over the postfix increment operator.		

	Use of increment operator	Equivalent code without increment operator
Unary pre increment Increment then assign	**Pre increment example with one variable**	

	`int iNum = 10;`
	`++iNum; //iNum = 11`
	`//It is the same as iNum ++;`
	Provides same result as
		`int iNum = 10;`
		`iNum = iNum + 1; //iNum = 11`

	Pre increment example with two variables	

	`int iNum = 10;`
	`int iResult = ++iNum;`
	`//iNum = 11; iResult = 11`
	Provides same result as
		`int iNum = 10;`
		`iNum = iNum + 1; //1. Increment iNum = 11`
		`iResult  = iNum; //2. Assign iResult = 11`

The prefix increment operator has precedence over assignment.		

3.19.4 *The decrement operator --*

Typically, the -- operator is used to decrease the operand by 1. (Subtract one from the value of a variable.) The decrement operator can be placed before a variable name (prefix), or after a variable name (postfix).

	Use of decrement operator		Equivalent code without decrement operator
Unary post decrement Assign then decrement	**Post decrement example with one variable**		

	`int iNum = 10;`
	`iNum --; //iNum = 9`
		Provides same result as	`int iNum = 10;`
			`iNum = iNum - 1; //iNum = 9`

	Post decrement example with two variables		

	`int iNum = 10;`
	`int iResult = iNum--;`
	`//iResult = 10; iNum = 9`
		Provides same result as	`int iNum = 10;`
			`iResult  = iNum; //1. Assign iResult = 10`
			`iNum = iNum - 1; //2. Decrement iNum = 9`

Assignment has precedence over the postfix decrement operator.			
	Use of decrement operator		**Equivalent code without decrement operator**
Unary pre decrement Decrement then assign	**Pre decrement example with one variable**		

	`int iNum = 10;`
	`--iNum; //iNum = 9`
	`//It is the same as iNum --;`
		Provides same result as	`int iNum = 10;`
			`iNum = iNum - 1; //iNum = 9`

	Pre decrement example with two variables		

	`int iNum = 10;`
	`int iResult = -- iNum;`
	`//iResult = 9; iNum = 9`
		Provides same result as	`int iNum = 10;`
			`iNum = iNum - 1; //1. Decrement iNum = 9`
			`iResult  = iNum; //2. Assign iResult = 9`

The prefix decrement operator has precedence over assignment.			

Activity 42: Unary operators

What is the value of each variable after the calculation is executed? Assume that the initial value of each variable in each statement is the integer value 10.

Statements	y	x
y = y * ++x;		
y =y * x++;		
y = y / ++x;		
y = y / x++;		
y = -(y + x--);		
y = -(y + --x);		
y = -(y + x++);		
y = -(-y + (x+1)*++x);		
y = -(y % x) * (--x / y)		
y = -(++y % x) * (x-- / y)		
y = -(++y % x) * (++x / y)		

3.20 Compound (augmented) assignment operators

It is often necessary in coding to change the value of a variable, for example:

```
iNum = iNum + 1;
rTotal = rTotal – rDiscount;
rPrice = rPrice * 1.05;
```

When the same variable is involved on both sides of the assignment operator, Java provides five *compound assignment operators* that enables a programmer to write shorter code.

Compound operator	'Ordinary' code	Code using compound assignment operator
+=	`iNum = iNum + 1;`	`iNum += 1;`
-=	`rTotal = rTotal – rDiscount;`	`rTotal -= rDiscount;`
*=	`rPrice = rPrice * 1.05;`	`rPrice *= 1.05;`
/=	`rFinal = rFinal / 5;`	`rFinal /= 5;`
%=	`iSecLeft = iSecLeft % 60;`	`iSecLeft %= 60;`

The value on the right-hand side of the compound operator can be:

- a literal (for example, 1 or 5) or
- a variable (for example rDiscount).

Activity 43: Compound (augmented) operators

Change ordinary code to code using a compound assignment operator and vice versa.

	Ordinary Code	Code using Compound operator
a)	`iSum = iSum + iX++;`	
b)		`iProduct *=--iX;`
c)	`rPrice = rPrice – rPrice * 0.05;`	
d)		`iNum %= iNum / 5;`
e)	Write 4 different Java statements that increase the value of the variable x by 1.	

f)	Write 4 different Java statements that decrease the value of the variable x by 1.
g)	Use only one statement to assign the product of the current value of integer variables of a and b to x, and increment the value of a.

Activity 44: Knowledge checklist

Use the following checklist and ensure you can correctly define or explain the following concepts:

	✓ / ?
Decrement	
Equivalent	
Increment	
Negation	
Post-	
Pre-	
Precedence	
Unary operator	

Activity 45: Skills checklist

Use the checklist to ensure you can do the following:

	✓ / ?
Use unary sign and unary numeric promotion in code	
Predict output of statement using the prefix increment and postfix increment operators	
Predict output of statement using the prefix decrement and postfix decrement operators	
Change ordinary code to code using compound(augmented) operators	

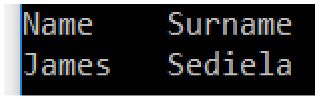
3.21 Formatting output

3.21.1 *Using escape sequences*

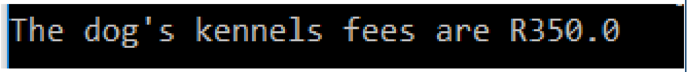
All our programs so far displayed output on the console in plain text, only using spaces to make text more readable. A number of special characters can be added to an output line (System.out.println() or System.out.print()) to add formatting to the output. These special characters are called *escape sequences*. An escape sequence always starts with a backslash (\) to indicate that a 'command' will follow, and then another character to indicate what should be done. The following table contains examples of escape sequences and the resulting output.

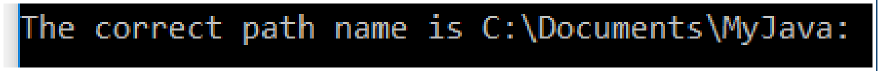
Sequence	'Command' character	Result of escape sequence
`\n`	New line	The cursor moves to a new line.
`\t`	Tab	A tab space is entered.
`\r`	Return	Cursor moves to the beginning of the line.
`\"`	Double quote	A double quote is printed in the output
`\'`	Single quote	A single quote is printed in the output
`\\`	Backslash	A backslash is printed in the output

Example code	Output

`String sName = "James";`
`String sSurname = "Sediela";`
`System.out.println("Name\tSurname");`
`System.out.println(sName + "\t" + sSurname);`
 	
Do not leave any spaces between the words that will act as headings.
	
	`Name    Surname`
	`James   Sediela`

`System.out.println("Name\tSurname\n" + sName + "\t" + sSurname);`	Same output, but only one println() statement was used.
`System.out.println("Push the \"escape\" character");`	

`double rAmount = 350;`
`System.out.println("The dog\'s kennels fees are R" + rAmount);`

`The dog's kennels fees are R350.0`	

`System.out.println("The correct path name is C:\\Documents\\MyJava:");`

`The correct path name is C:\Documents\MyJava:`	

Activity 46: Predict output when escape sequences are used

Predict the output of the following statements:

Statements	Output

`int iValue = 22, iCount = 10; String sName = "Sofia";`
`System.out.println ("Hi" + "\n"+ sName);`
`System.out.println ("Calculate:  answer \t\"times\" \t count is:");`
`System.out.println ("\t"+iCount*iValue);`	

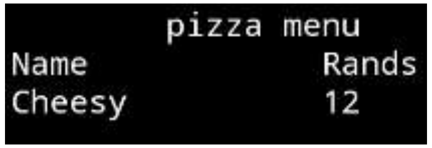
`System.out.println("Sonia said  \"Hello! \" to me.");`	

Activity 47: Write code to format output using escape sequences

Write Java programs for each of the following tasks:

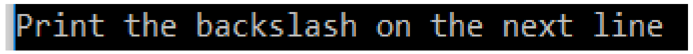
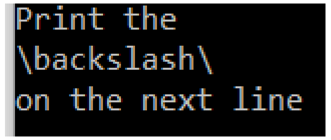
a) Complete the statements so that your output looks like the following screenshot using a single println() to display the output.

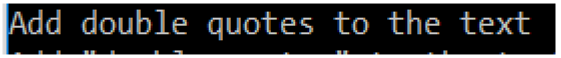
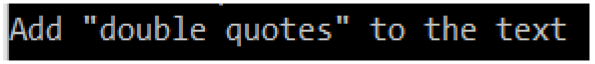
Output screenshot	Statements

 `        pizza menu`
`Name             Rands`
`Cheesy           12`
	`int iPrice=12;`
	
	…………………………………………….

b) Use a single println() with escape sequences to change the output statement to produce the formatted output.

Original code and output	Formatted output

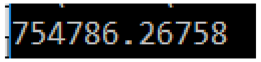
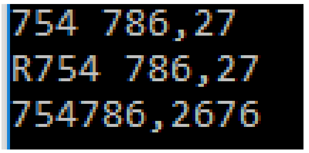
`System.out.println("Print the backslash on the next line");`
 `Print the backslash on the next line`
	 `Print the`
	`\backslash\`
	`on the next line`

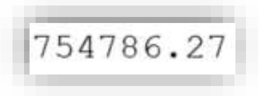
`System.out.println("Add double quotes to the text");`  `Add double quotes to the text`	 `Add "double quotes" to the text`

3.21.2 *Using the DecimalFormat class*

Until now we have displayed floating-point numbers as they are – displaying all decimal values, even when we displayed monetary values. However, this does not always look professional, so all programming languages have techniques to display output in a specificformat.

There are various ways to format floating-point values. Weare going to use an object (instance) of the DecimalFormatclass to do the formatting.

Example of unformatted output	Examples of formatted output
 `754786.26758`	 `754 786,27` `R754 786,27` `754786,2676`

To display a floating-point number in a specific format, you need to do the following:	
1. Import the DecimalFormat class from the java.text package.	`import java.text.DecimalFormat;`
2. Call the constructor method called DecimalFormat to create an object based on the DecimalFormat class. Pass an argument (a string) that contains a *formatting pattern* to the object.	`DecimalFormat formatter = new DecimalFormat ("0.00");`
3. Call the format method from the object to display a floating-point value applying the formatting as specified in the formatting pattern.	`double rNumber = 754786.26758;` `System.out.println(formatter.format(rNumber));`
This will be the formatted output.	 `754786.27`

In the formatting pattern, each character corresponds to a position in a number. Let us look at the pattern "0.00".

0	**.**	**00**
Refers to one digit before the decimal point	↑ the position of the decimal point	refers to the 2 digits after the decimal point.
754786	.	27
There are many digits in front of the decimal point. So, the single 0 did not have any influence in the formatting pattern.		Because there were only 2 zeroes in the formatting pattern, only two digits are displayed (the value is rounded to two decimals).

Several characters can be used I the formatting pattern.

Character	**Description/action**
#	A digit should be displayed in this position if it is present. If there is no digit present, nothing should be displayed.
,	Represents the decimal separator (in South Africa it is a space).
.	Represents the digital point (in South Africa a comma is used).
0	A 0 (zero) should be displayed in this position if there is no digit present in this position.
%	A % sign can be added as the last character in the formatting pattern. This causes the number to be multiplied by 100, and the % sign is added at eh end of the displayed value.

Here are some examples of unformatted values, the formatting patterns that were applied, and resulting formatted output.

Value	Formatting pattern	Result	Comments
754786.26758	"0.00"	754786.27	The value is rounded to 2 decimals.
0.26758	"000.00"	000.27	The value is rounded to 2 decimals. The 3 zeros before the decimal point indicates that 3 digits must be displayed, and since there are no non-zero digits in this number, three zeros will be displayed.
0.26758	"##.###"	0.268	The 2 hashes before the decimal point indicates that 2 values should be displayed if there are two. But there is only one value (the zero), so only one digit is displayed before the decimal point.
23754786.4	"###,###.##"	23, 754, 786.4	
723754786.4	###,###.00	723 754 786,40	
0.6	"0.00%"	60,00%	
4563.765	"R#,###.00"	R4, 563.77	
834431620	"'0'000000000"	0834431620	This can be used to format cell phone numbers.

Use the following code to experiment with different values and formatting patterns.

```
import java.text.DecimalFormat;
public class DecimalFormatExamples { //start class
     public static void main(String[] args) {
          DecimalFormat formatter = new DecimalFormat();
          double rValue = 754786.26758;
          String sPattern = "0.00";
          formatter.applyPattern(sPattern);
          System.out.println(rValue + "\t\"" + sPattern + "\"\t" + formatter.format(rValue));
     }
} //end class
```

You can call the constructor method without an argument. Then you can send the formatting pattern to the object using the applyPattern() method.

Note

The DecimalFormat class' methods just make the value display in a certain way. The actual value of the variable is still the same. For example, look at the following values:

The value of variable rValue	Formatting pattern	Value displayed
754786.26758	"0.00"	754786,27

If another calculation is done, for example:

```
double rFinal = rValue * 100000; //multiply by 10⁵
System.out.println(rFinal);
```

rValue	754786.26758
rFinal	75478626758

The value 75478626758 will be displayed. All the decimal values are still stored in the variable in the memory of the computer. The value displayed on the screen is just formatted to look different, it does not actually change.

Activity 48: Predict the format of a number when DecimalFormat class is used

Complete the following table.

How will the values be displayed after they have been formatted with the following formatting patterns?

Values	Formatting patterns	Formatted values
123456.789	"R###,###.##"	
0.045	"0.00%"	
4561237.3	"###,###.00"	
4561237.3	"###,###.##"	
5486523	"0120000000"	

Activity 49: Format output using methods from the DecimalFormat class

What formatting pattern should be used to change (modify) the unformatted output to the formatted one.

Formatting pattern	Unformatted output	Formatted output
	45.0768	R45.08
	0.007892	00.8%
	4.05	04.50
	4.05	04.05
	974586.2	974,586.20

Activity 50: Knowledge checklist

Use the following checklist and ensure you can correctly define or explain the following concepts:

	✓ / ?
Formatter	
DecimalFormat	
Escape sequences	
Formatting pattern	
# character	
Instance of a DecimalFormat class	

Activity 51: Skills checklist

Use the checklist to ensure you can do the following:

	✓ / ?
Use escape sequences in Java programs to format output	
Predict output when escape sequences are used	
Predict the format of a number when DecimalFormat class methods and formatting patterns are used	
Format output using DecimalFormat class methods	

3.22 Finding errors in a Java program

Often, after writing an essay, you were advised to read through it once or twice again before submitting it for evaluation. And often, your final version was always slightly different than the original and better crafted with fewer mistakes. This is because you were ableto identify and correct some spelling mistakes, tense sequencing errors, sometimes going as far as rewriting entire paragraphs or change the sequence of your sentences to better convey your ideas. Your final mark was greatly influenced by the number of mistakes that you could not correct before submitting as well as how logical your reasoning was.

Similar principles to find errors apply to programming. However, when writing a program, it is not about marks, your program must bewithout errors, otherwise it may not compile, or it will fail to provide correct output.

When errors occur, programmers say:

- An error was thrown,
- An exception was thrown,
- The program bombed out or
- The program crashed.

There are different types of errors that can occur in a program, for example:

- The programmer disregards the syntax rules of the programming language, for example, when a word is misspelled or notcapitalized, brackets are not in pairs, or a semi-colon is omitted at the end of a statement – this is a syntax error.
- An illegal operation is performed during run-time, such as dividing a number by zero –this is a run-time error.
- A program does not display correct result, even though the input values are correct – this is a logical error.

No matter how smart or how careful you are, errors are your constant companion. With practice, you will get slightly better at not making errors, and much, much better at finding and correcting them, which is the mostimportant aspect.

Programming errors are generally known as *bugs* and the process to remove bugs from program is called *debugging.*

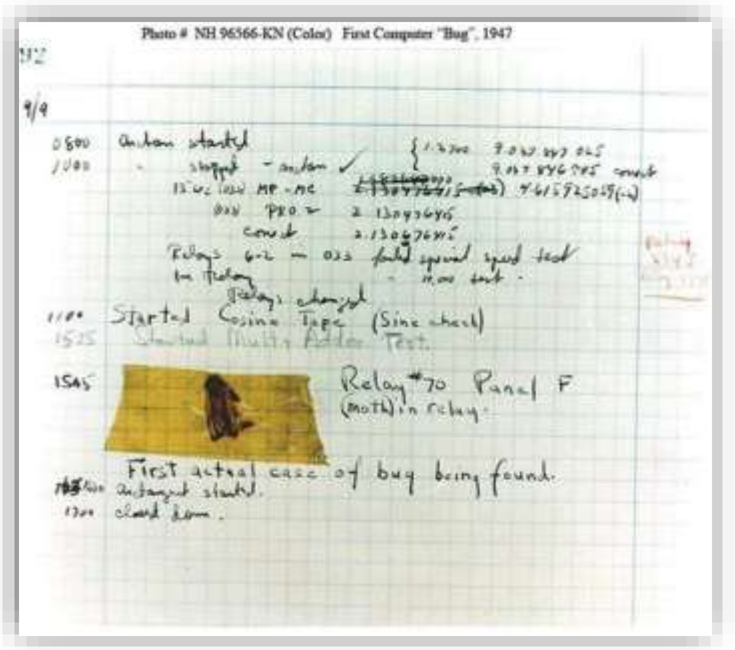


Photo # NH 96566-KN (Color) First Computer "Bug", 1947
Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.

The term "bug" to describe defects has been a part of engineering jargon since the 1870s, long before the invention of the electronic computer and computer software. It was used in hardware engineering to describe mechanical malfunctions. (Magoun & Israel, 2013). Thomas Edison wrote the following wordsin a letter to an associate in 1878:

"It has been just so in all of my inventions. The first step is an intuition, and comeswith a burst, then difficulties arise—this thing gives out and [it is] then that "Bugs"—as such little faults and difficulties are called—show themselves and months of intense watching, study and labour are requisite before commercial success or failure is certainly reached." (Today in Science History, n.d.)

The terms *bug* and *debugging* as currently used in computer programming was most probably made popular by Admiral Grace Hopper. In 1947, while she was working on a Mark II computer at Harvard University, her associates discovered an insect (a moth) stuck in a relay, thereby impeding operation. In the logbook she recorded 'Firs tactual case of bug found'. The moth was taped in the logbook.(Computer History Museum, 2020)

Let us now look in more detail at the three types of errors

3.22.1 *Syntax or compile-time errors*

Errors that are detected by the compiler are called *syntax errors* or *compile-time errors*. Syntax errors result from errors in codeconstruction. These are some common syntax errors (Murach, 2017, p. 183):

- Misspelling keywords.
- Forgetting to declare a data type for a variable.
- Forgetting an opening or closing parenthesis, bracket, brace, or comment character.
- Forgetting to code a semicolon at the end of the statement.
- Forgetting an opening or closing quotation mark.
- Misspelling or incorrectly capitalizing an identifier.

- Using a reserved word as an identifier.
- The class name is not the same as the name of the file the class is stored in.

These errors are similar to your general spelling mistakes on your essay paper. Since each programming language has its own syntax and set of rules, syntax errors are common to both beginners and advanced programmers writing code in an unfamiliar language. Syntax errors are easily detected and thrown by the compilers and will prevent your program from running. In addition, the compiler will pinpoint where they are and what caused them.

3.22.2 *Run-time or semantic errors*

Run-time or semantic errors cause a program to terminate abnormally ('bomb out'). When the programming environment detects an operation that is impossible to carry out while a program is running, a run-time error will occur. Such errors are also called run-time exceptions and programmers say, 'the program threw an exception'. The exception stops the normal execution of the program.

These are common causes of run-time errors:

- Performing an illegal mathematical operation such as dividing a value by zero or calculating the square root of a negative number.
- The user of the program enters an illegal value, for example, the program code indicated that an integer value should be read from the keyboard, but the user enters a floating-point number.
- The program needs to read a value from a file, but the file does not exist.

Run-time errors can be prevented by the programmer to a large extent. Later in this module you will learn how to write code to let a program deal with errors, instead of bombing out. This is called developing robust programs.

3.22.3 *Logical errors*

Logical errors occur when a program does not perform the way it was intended to. This happens for many different reasons but in most cases, it is because the programmer did not respect the specifications of the problem when coding.

Logical errors are the hardest of all error types to detect. They do not cause the program to crash or simply not work at all, the program will run perfectly but will “misbehave” in some way, returning wrong output of some kind. Logical errors are usually very simple mistakes involving missing or wrong “computer logic.”

These are examples of logical errors:

- A logical mathematical mistake in a formula, for example, calculating the average of three values as follows:
  ```
  double rAvg = iMark 1 + iMark2 + iMark 3 / 3.0;
  ```
- A local variable field is declared but not initialized.

Logical errors “make sense” as to the computer language itself (they are valid program statements), but they simply do not fit into the program correctly thus fail to achieve the desired goal.

Summary of Java errors

- Syntax errors are like “spelling and grammar” problems. They often stem from typing errors where parentheses or singlecharacters are entered incorrectly. With syntax errors your program will not compile at all.
- Run-time or semantic errors have to do with meaning/context. It is like using a wrong word or in the wrong place in a human language sentence. A programming language example would be trying to assign a name to a variable of integer data type. Yourprogram will compile, but when the user enters a string of characters instead of a number, the program crashes.
- Logic errors have to do with program flow. If you use the wrong arithmetic operator or operations in the wrong order, it will resultin a logical error.

Activity 52: Knowledge checklist

Use the checklist and ensure you can correctly define or explain the following concepts:

	✓ / ?
Syntax error	
Run-time error	
Logical error	
Debugging	
Compile-time error	
Semantic error	

Activity 53: Skills checklist

Use the checklist to ensure you can do the following:

	✓ / ?
Interpret compiler error messages and fix syntax errors.	
Identify and fix run-time errors.	
Identify and fix logical errors.	
Provide examples of syntax errors.	
Provide examples of run-time errors.	
Provide examples of logical errors.	

3.23 Sources consulted and/or referenced

Agarwal, H. (n.d.). *Geeksforgeek*. Retrieved April 17, 2020, from Geeksforgeeks: A Computer Science Portal For Geeks: https://www.geeksforgeeks.org

ASCII-Table. (2020). *RapidTables*. Retrieved from RapidTables: https://www.rapidtables.com/code/text/ascii-table.html

Baeldung. (2019, August 2). *Lossy Conversion in Java*. Retrieved March 07, 2020, from Baeldung: https://www.baeldung.com/java-lossy-conversion

Columbia University. (n.d.). *Signifivant figures*. Retrieved March 15, 2020, from Columbia University: http://ccnmtl.columbia.edu/projects/mmt/frontiers/web/chapter_5/6665.html

Computer History Museum. (2020). *What happened on September 9th*. Retrieved June 06, 2020, from Computer History Museum: https://www.computerhistory.org/tdih/September/9/

Department: Government communication and Information System REPUBLIC OF SOUTH AFRICA. (2013, March). *Guidelines: Editorial Style Guide.* Retrieved May11, 2020, from Government communications: https://www.gcis.gov.za/sites/default/files/editorial_styleguide_2011.pdf

edpresso. (2020). *How to use the Math.Random() Method in Java?* Retrieved April 21, 2020, from Educative: Interactive Course for Software Developers: https://www.educative.io/edpresso/how-to-use-the-mathrandom-method-in-java

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.

GCSE. (2020). *Introducing binary*. Retrieved from BBC: https://www.bbc.co.uk/bitesize/guides/zwsbwmn

Ishida, R. (2015). *Character encodings for beginners*. Retrieved from W3C: https://www.w3.org/International/questions/qa-what-is-encoding

JavaTpoint. (2020). *Java.Util.Random.NextInt() Method - Tutorialspoint*. Retrieved April 20, 2020, from JavaTpoint: https://www.tutorialspoint.com/java/util/random_nextint_inc_exc.htm

Johari, A. (2019, November 26). *What are the different Applications of Java?* Retrieved from edureka!: https://www.edureka.co/blog/applications-of-java/Liang, D. Y. (2013). *Introduction to Java Programming: Comprehensive Version* (9th ed.). Harlow, England: Pearson Education Limited.

Magoun, A. B., & Israel, P. (2013, August 01). *Did You Know? Edison Coined the Term "Bug"*. Retrieved May 19, 2020, from IEEE Spectrum: https://spectrum.ieee.org/the-institute/ieee-history/did-you-know-edison-coined-the-term-bug

Mueller, J. P. (n.d.). *Java: Negation, Bitwise Not, and Boolean Not*. Retrieved from dummies.com: https://www.dummies.com/programming/java/java-negation-bitwise-not-and-boolean-not/

Murach, J. (2017). *Murach's Java programming* (5th ed.). United States of America: Mike Murach & Associates, Inc.

National Lottery. (2020, April 06). *Results:Lotto*. Retrieved from National Lottery: https://www.nationallottery.co.za/results.lotto

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). *Oracle Java Documentation.* Retrieved May 23, 2020, from Primitive Data Types: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Oracle. (2019). *Primitive Data Types*. Retrieved April 30, 2020, from The Java Tutorials: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Pandya, S. (2019). *Re:Difference between TRNG or PRNG?* Retrieved from ReearchGate: https://www.researchgate.net/post/Difference_between_TRNG_or_PRNG#:~:text=The%20difference%20between%20true%20random,(completely%20computer%2D generated).

Posch, M. (2013, May 17). *Java basics - Promotion and Casting.* Retrieved June 19, 2020, from YouTube: https://www.youtube.com/watch?v=FNPEKwQUJwUReges, S., & Stepp, M. (2011). *Building Java Programs: A back to basics approach* (2nd ed.). Boston: Addison-Wesley.

SC: Advancing the Chemical Sciences. (n.d.). *gridlocks*. Retrieved from RSC: Advancing the Chemical Sciences: http://www.rsc.org/learn-chemistry/resources/gridlocks/downloads/SignificantFigures.pdf

Siegfried, R. M. (2017). *CSC 170 - Introduction to Computers and Their Applications.* Retrieved from Adelphi University: https://home.adelphi.edu/~siegfried/cs170/170l1.pdf

Spacey, J. (2016, October 20). *Pseudorandom vs Random*. Retrieved from Simplicable: https://simplicable.com/new/pseudorandom-vs-random

Sportsclub. (2020, Macrh 02). *Stats: Soweto derby in numbers*. Retrieved March 10, 2020, from Sportsclub: Sportsclub: https://www.sportsclub.co.za/stats-soweto-derby-in-numbers-2/

Teachers Pay Teachers. (n.d.). *Lost on an Island Binary Challenge Worksheet*. Retrieved June 03, 2020, from Teachers Pay Teachers: https://www.teacherspayteachers.com/Product/Lost-on-an-Island-Binary-Challenge-Worksheet-222634

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

Today in Science History. (n.d.). *Science quotes by Thomas Edison*. Retrieved June 06, 2020, from Today in Science History: https://todayinsci.com/E/Edison_Thomas/EdisonThomas-Quotations.htm

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

w2schools.com. (2020). *Java data Types*. Retrieved April 30, 2020, from w3schools.com: https://www.w3schools.com/java/java_data_types.asp

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.

Whyman, A. (n.d.). *The World's First Computer Bug*. Retrieved June 09, 2020, from Global App Testing: https://www.globalapptesting.com/blog/the-worlds-first-computer-bug-global-app-testing

3.24 Recommended additional learning resources

These URLs links to various training courses.

What is Java? A Beginner's Guide to Java and its Evolution | Edureka

https://www.w3schools.com/java/java_packages.asp

https://www.w3schools.com/java/default.asp

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

http://www.cs.umd.edu/~clin/MoreJava/Intro/prim-char.html

https://www.coursera.org/learn/java-programming-design-principles?specialization=java-programming

https://www.coursera.org/learn/java-programming

https://www.youtube.com/results?search_query=margret+posch+java