Tshwane University of Technology

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Principles of Programming A (Extended)

PPAF05D and TROF05D

Unit 4

Selection control structures in Java

DEVELOPED: 2019-2021

You OWN your learning – therefore you need to reflect on your learning process. At the end of each lesson, you need to have a discussion with yourself and/or fellow students and/or lecturer to reflect on the lesson of the day. See the paragraph called ‘Reflection’ at the end of the chapter for examples of questions you can include in your discussions.

Unit 4: Selection control structures in Java

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

UNIT 4: Decision making and repetition in programming **Outcome:** The student must be able to develop programming solutions involving decisions and repetition
Assessment Criteria
• An algorithm to solve a basic mathematical problem requiring decision making structures can be designed, presented in pseudo code, and implemented in a programming language providing correct output. • Logical operators such as ≤; ≥; <; >; ≠; AND; OR; NOT can be used to create simple and compound relational and logical expressions for decision making structures. • Output for algorithms using decision and repetition structures can be predicted by means of a trace table.

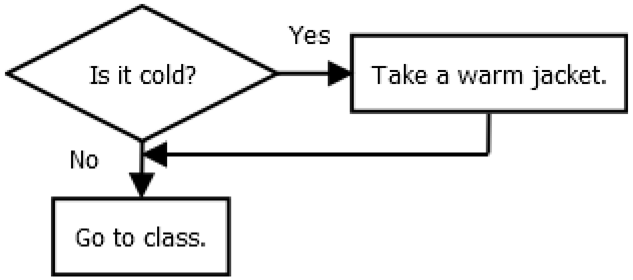
The knowledge and skills as described in the following Units and Outcomes are relevant and have to be applied in this Unit:

Unit 1: Problem solving; **Outcome:** The student must be able to apply logic and knowledge to solve basic problems.

Unit 2: Basic programming principles; **Outcome:** The student must be able to apply algorithmic problem solving to basic mathematical and programming problems.

4.1 Making decisions in real life

Decision making is part of our every-day life. For example, to decide what you will wear to class, you ask yourself a question, and based on the answer, you make a decision. You ask yourself: "Is it cold?". If the answer is yes, you will take a warm jacket with you, then go to class. To help to visualize this decision-making process, we can use a flowchart.



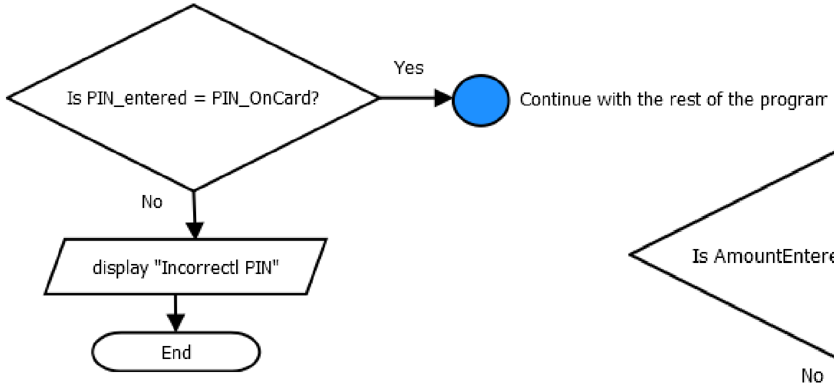
- Sequential
- Conditional

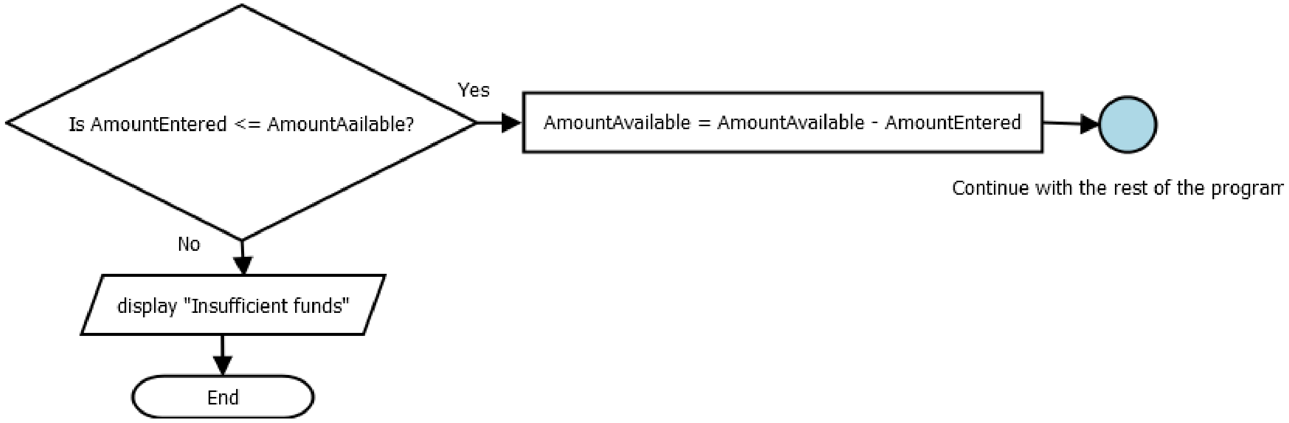
- ATM
- PIN

Similarly, the program controlling an ATM has to make decisions by 'asking' questions, for example:

- Is the PIN entered the same as the PIN on the card?
- Is the amount entered smaller than or equal to the amount in the account?

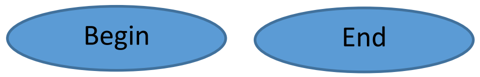
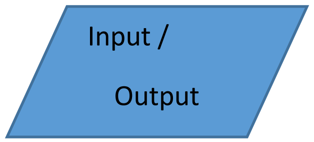
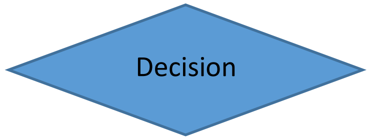
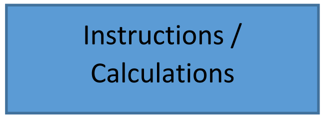
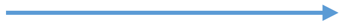
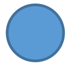
This can also be demonstrated using a flowchart.


Is PIN_entered = PIN_OnCard?
Yes
Continue with the rest of the program
No
display "Incorrectl PIN"
End


Is AmountEntered <= AmountAailable?
Yes
AmountAvailable = AmountAvailable - AmountEntered
Continue with the rest of the program
No
display "Insufficient funds"
End

4.2 Creating flowcharts to graphically represent decision-making

To draw a flowchart there are some basic symbols that needs to be used.

 Indicate the start and end of a program.	 Indicate when the content of a variable should be read from the keyboard, or what should be displayed on the console panel.	 Contains the question when a decision needs to be made (the condition).
	 Lines to connect input, output, conditions, and instructions. This indicates flow of the program.	 Connector to link different branches of a program.

Activity 1: Create a flowchart for a real-life scenario

a) To enter a school concert, students must collect a card that they will present at the door before they enter. Girls are receiving pink cards and boys are receiving green cards. Draw a flow chart that will test what colour of card a student will receive to enter the concert.

b) At the end of a workshop on healthy diets, all participants should be measured and weighed, and calculate their BMI. Before exiting the venue, the students who did calculate their BMI receive a sample of a new protein enriched porridge, then exit the venue. Those who did not calculate their BMI, must go back to ask the assistant nutritionist what their BMI is, then they can collect the sample and leave. Draw a flow chart that will test who goes back to the nutritionist assistant before they collect the sample and leave.

c) To make a good cup of instant coffee, there are a number of steps you have to follow: For example, fill the kettle with water, switch the kettle on, get a mug, add a teaspoon of instant coffee to the mug, if you want some sugar add some sugar to the mug, check if the water has boiled. Pour the water into the mug only if the water has boiled, then stir until granules dissipate and finally decide if you want milk, add some milk. Draw a flow chart that will show the steps of making a cup of instant coffee.

d) There are steps that are important when crossing the road.

- You should first observe the traffic,
- Then, only cross the road when there is no car approaching.
- Using the above steps, draw a flowchart that will test when to cross the road.

4.3 Making decisions in programming

You learned to write programs that primarily did mathematical calculations. In the programs you wrote, all statements were executed, from top to bottom. We say the statements were executed *sequentially*. To write programs that can make decisions, we need a programming structure where we can give a program an instruction to select which statements to execute – therefore to make a decision. This programming structure is called the *if* statement. It is one of the *selection control* structures in Java. It controls the flow of the program by selecting (deciding) which statements to execute.

Here are some examples of if statements in Java code:

Example 1

```
System.out.print("What is the final mark? ");
int iPerc = keyboard.nextInt();
// Processing
if (iPerc >= 75)
    System.out.println("Well done you have a distinction");
System.out.print("Best wishes with your studies!");
```

(iPerc >= 75) is called a condition.

If the condition is true, the output will be displayed.

Example 2

```
double rAmount, rDiscount, rFinal;
System.out.print("What amount did the customer buy for? ");
rAmount = keyboard.nextDouble();
// Processing
if (rAmount > 500)
    rDiscount = rAmount * 0.1; //Give 10% discount
```

(rAmount > 500) is the condition.

If the condition is true, the discount will be 10% of the amount, otherwise the discount will be 0.

```
         else
             rDiscount = 0;
         rFinal = rAmount -  rDiscount;
         System.out.println("Original amount: R" + rAmount);
         System.out.println("Discount amount: R" + rDiscount);
         System.out.println("Final amount to pay: R" + rFinal);
```

When you want to make a decision, the most important aspect is what question to ask. For example, if you want to decide whether to take a warm jacket to class, you must ask: “Is it cold?”.

When you want to write a program to make a decision, you have to formulate the question in a certain way. You cannot just ask Java “Is it cold?”. Programming languages can only work with variables and values, according to strict syntax rules. So, you have to ‘translate’ the question “Is it cold?” to create a condition (relational expression) that involves comparing two values.

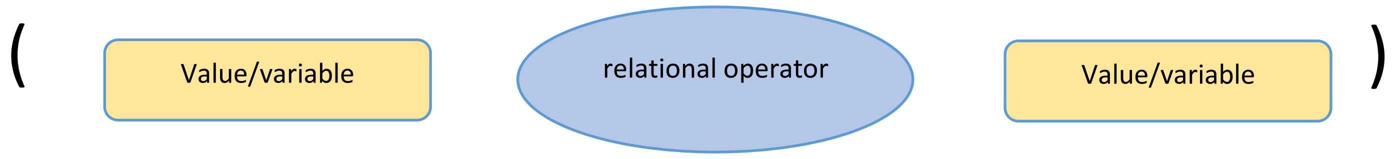
4.4 Creating a relational expression (condition)

We can, for example, agree that it is cold when the temperature is less than 20°C. Then we can create a relational expression where we compare the temperature to 20, and use it in a question (when two values are compared, it is called a relational expression):

The question in English	The question using values	The relational expression
Is it cold?	Is the temperature lower than 20°?	rTemp < 20

A relational expression always consists of two values that are compared to one another. In this relational expression, the content of the variable rTemp is compared to the value 20 using the *less than* relational operator (<). (Some sources also refer to a relational operator as a comparison operator.)

Format of a relational expression



(rTemp < 20)

(iAge >=80)

(50 < rFinalmark)

(iNumItems == 100)

(iFirst > iSecond)

Examples of relational expression – note that:

- both values may be variables.
- the variable may be the second value, and the literal the first.

There are a number of relational operators you can use, similar to those you use in mathematical subjects.

Math symbol	**Java relational operator**	**Name**	**Example** (the value of rTemp is 20)	**Result**
>	>	Greater than	rTemp > 20	false
<	<	Less than	rTemp < 20	false
≥	>=	Greater than or equal to	rTemp >= 20	true
≤	<=	Less than or equal to	rTemp <= 20	true
=	==	Equal to	rTemp == 20	true
≠	!=	Not equal to	rTemp != 20	false

You can compare characters since it is the same as comparing their Unicode values. For example, 'a' is larger than 'A' because the Unicode (ASCII) value for 'a' is 97, and for 'A' it is 65 . (Lang, 2013). Therefore, the following relational expressions are valid:

```
if (cGroup == 'A')
    System.out.print("Mr Kgoete is the teacher for Life Science group A.");

if (cClassGroup > 'M')
    System.out.print("There are no class groups greater than M.");
```

Take note of the single quotes when comparing a character value.

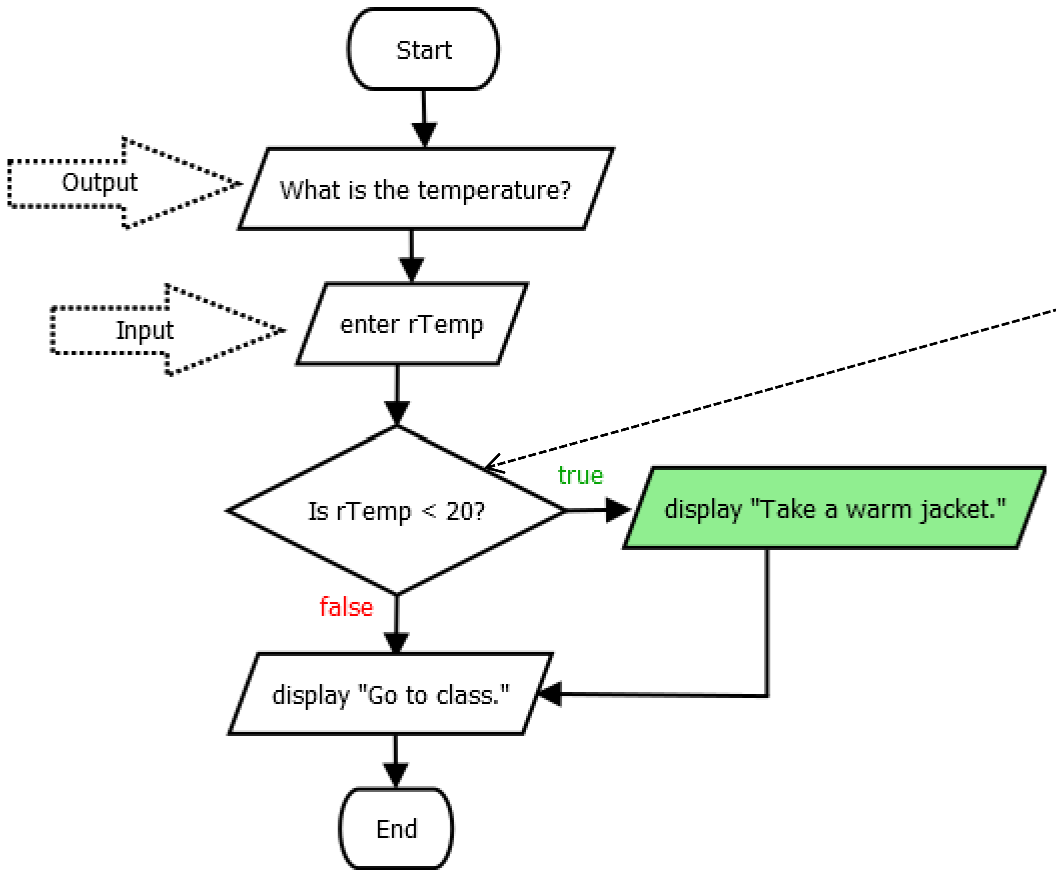
Activity 2: Create relational expressions and predict results

Determine the result (true / false) of the following conditions. The first row contains an example.

Condition (question) in English	Value of variable	Relational expression	Result
Example: *Is the person 21 years or older?*	*iAge = 18*	*(iAge >= 21)*	*False*
Is your percentage equal to 5 more than 50?	iPercent=45		
Is the building 6 meters high or less?	iHeight= 5		
Is the number of eggs not the same as a dozen?	iNumEggs=12		
Do you have at least 80 pens?	iNumPen=90		
Is the maximum load 50 kg?	iWeight=60		
Is the number less than 1.85?	rNumber=1.79		
Is your lucky number not more than 'Z'?	iLuckyNum=97		

4.5 Comparing algorithms and Java programs containing selection control structures

Let us now create a complete flowchart for the example on determining whether to take a jacket to class:



The value of the variable rTemp will be compared to 20 using the relational operator < (less than). The question is therefore:

Is the value of rTemp less than 20?

The answer can only be 'Yes' (true) or 'No' (false).

If the result of the relational expression is true (answer to the question is 'Yes'), then the program flows to where the 'true' arrow is pointing.

If the result of the relational expression is false (answer to the question is 'No'), then the program flows to where the 'false' arrow is pointing.

When we write programs that includes decisions, it is sometimes easier to use a flowchart that provides a more graphic view of the flow of the program. A flowchart is also a type of algorithm – therefore, you can write a Java program directly after having created a flowchart – you do not have to write pseudo code first.

We do show the pseudo code and Java code based on the flow chart here, so that you can compare them and see the similarities and the differences.

- Algorithm
- Flowchart
- Similar
- Different

Pseudo code

```
ChooseClothes // the class name
begin //start of the class
     main method
     begin //main method
          //declare variables
          double rTemp

          //input
          display "What is the temperature? "
          enter rTemp
          //decision
          if (rTemp < 20) then //test the condition
               begin
                    display "Take a warm jacket."
               end
          display "Go to campus."
     end //main method
end //end of the class
```

Java code

```
package ChooseClothes;
import java.util.Scanner;
public class ChooseClothes
{ //start class
   public static void main(String[] args)
   { //start main method
      //declarations
      Scanner keyboard = new Scanner(System.in);
      double rTemp;
      //input
      System.out.print("What is the temperature? ");
      rTemp = keyboard.nextDouble();
      //decision
      if (rTemp < 20) //test the condition
      { //begin the if-block - statements to execute if condition is true
         System.out.println("Take a warm jacket.");
      } //end the if-block
      System.out.println("Go to campus.");
   } //end main method
} //end class
```

4.6 The if statement

The if statement is used to determine whether one or more statements should be executed before continuing with the rest of the program. In general it can be described as follows:

```
if (condition)
{ //beginning of if-block
     execute these statements if the condition is true
     …………
} // end if- block
```
continue with the program

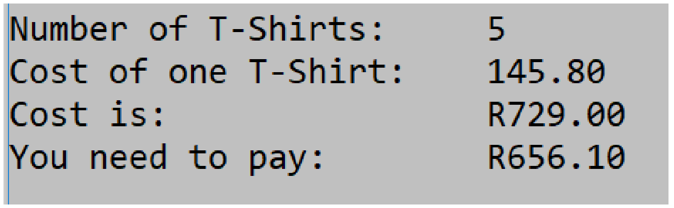
- *If* is a Java key word.
- The condition must be surrounded by parentheses.
- The braces indicate the beginning and the end of the block of statements that must be executed.
- A block of statements may consist of one statement only.

These are examples of Java programs that use an if-statement to determine whether some statements should be executed.

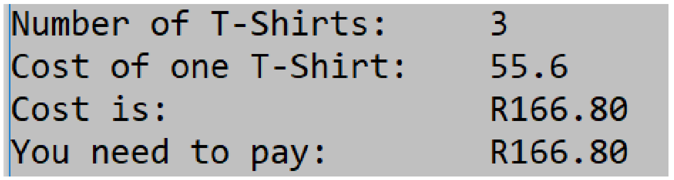
Example 1 (T-shirt discount)

This program reads the number of T-shirts a customer wants, and the cost of one T-shirt. It calculates the amount to pay. If the amount due is more than R200, a discount of 10% is provided.

```
package TshirtDiscount;
import java.util.Scanner;
public class TshirtDiscount
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iNumShirts;
        double rPrice1Shirt, rAmtDue;
        System.out.print("Number of T-Shirts: ");
```


```
Number of T-Shirts:      5
Cost of one T-Shirt:     145.80
Cost is:                 R729.00
You need to pay:         R656.10
```

```
        iNumShirts = keyboard.nextInt();
        System.out.print("Cost of one T-Shirt: ");
        rPrice1Shirt = keyboard.nextDouble();
        rAmtDue = iNumShirts * rPrice1Shirt;
        System.out.println("Cost is: R" + rAmtDue);
        // Processing
        if (rAmtDue > 200)
        {
            rAmtDue = rAmtDue * 0.9; //Give 10% discount - keep 90% of the price
        }
        System.out.println("You need to pay: R" + rAmtDue);
    } //end main method
} //end class
```



```
Number of T-Shirts:     3
Cost of one T-Shirt:    55.6
Cost is:                R166.80
You need to pay:        R166.80
```

Activity 3: Format output of T-shirt program

(a) Write the name of each variable next to the corresponding value in each screenshot (referring to example 1 above).

(b) Draw a flowchart for this program (of example 1).

(c) Create the TshirtDiscount class (code provided in example 1 of topic 4.6.). Add code to format the output as displayed in the screen shots. See Unit 3 Topic 3.21 for examples of escape sequences and methods of the DecimalFormat class that you can use.

Example 2 Courier company (Adapted from (Wassermann, et al., 2011))

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

- The first part of the cost (basic cost) is calculated as R5.50 per kg + transport cost per kilometre.
    - The transport cost is based on the mode of transport.
        - per road: 80^c per km.
        - per train: 50^c per km
        - by air: R1.50 per km
- If the customer wants to insure the parcel, the cost is increased by 11%.

- VAT of 15% must be added to the cost (after insurance has been added).

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

          final int VAT = 15, INSURE_PERC = 11;
          final double PERKG = 5.5, ROAD = 0.8, TRAIN = 0.5, AIR = 1.5;
          double rMass, rKM, rTotal, rTransp = 0;
          char cTransp, cInsure;

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

          //set the transport cost based on the mode of travel
          if (cTransp == 'R')
               rTransp = ROAD;
          if (cTransp == 'T')
               rTransp = TRAIN;
          if (cTransp == 'A')
               rTransp = AIR;
```

```
        //calculate the cost based on the mass and the distance
        rTotal = rMass * PERKG + rTransp * rKM;

        //add insurance cost if chosen
        if (cInsure == 'Y')
            rTotal = rTotal + INSURE_PERC/100.0 * rTotal;

        //add VAT to the total
        rTotal = rTotal + VAT/100.0 * rTotal;

        //output
        System.out.println("You need to pay: R" + rTotal);
    } //end main method
} //end class
```

Remember to divide by 100.0 (a double data type), not 100 (an integer data type).

```
11/100 = 0
11/100.0 = 0.11
```

Activity 4: Questions regarding the Courier company class

Answer the following questions regarding the Courier company example:

a) What values are input values? Provide the names of the variables as well as a description of what will be stored in each variable.

Name of input variable	Description	Data type

b) If the customer chooses to send the parcel by train, what will the cost per km be?

c) What is the name of the constant where the cost of transport per km is stored when a customer chooses to send the parcel by air?

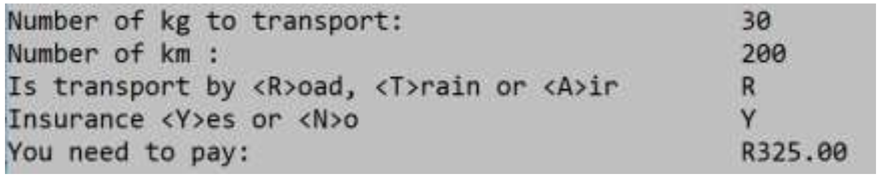
d) Use the values in the following table and calculate what the cost should be using a pocket calculator.

Mass	**Kilometres**	**Transport mode**	**Insurance**	**Total cost**
55.6kg	245	Train	Yes	
10kg	55.8	Road	No	
500.5kg	1050	Air	Yes	

e) Write the program as provided in the example. Use the values provided in the previous question to test your output.

f) What will happen if the customer enters the value y instead of Y when asked whether insurance is required?

g) Add code to format the output as in the screenshot alongside.



```
Number of kg to transport:                    30
Number of km :                                200
Is transport by <R>oad, <T>rain or <A>ir      R
Insurance <Y>es or <N>o                       Y
You need to pay:                              R325.00
```

h) Change this declaration statement

```
    double rMass, rKM, rTotal, rTransp = 0;
```

to the following;

```
    double rMass, rKM, rTotal, rTransp;
```

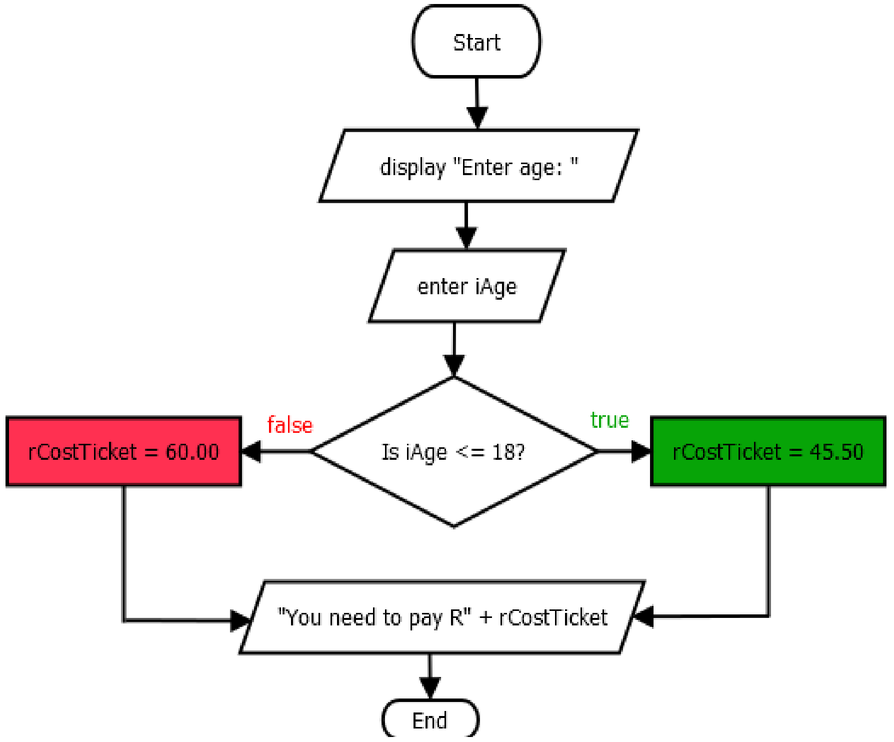
Why will the program not compile after you removed = 0?

4.7 The if .. else statement

When using the if statement, a program can execute a number of statements when a relational expression evaluates to true. The if control statement can be extended to let a program execute a different set of instructions when the relational expression evaluates to false. This control structure is the if .. else statement. In general it can be described as follows:

```
if (condition)
{ //beginning of if-block
    execute these statements if the condition is true
    …………
} // end if- block
else
{ //beginning of else-block
    execute these statements if the condition is false
    …………
} //end else-block
continue with the program
```

The flowchart alongside illustrates a scenario where the price of a concert ticket depends on the age of a person. If the person is 18 years or younger, the ticket price is R45.50, otherwise the price is R60.00.



Pseudo code	Java code
	`package TicketPrice;`
	`import java.util.Scanner;`
`TicketPrice // the class name`	`public class TicketPrice`
`begin //start of the class`	`{ //start class`
`    main method`	`    public static void main(String[] args)`
`    begin //main method`	`    { //start main method`
`        //declare variables`	`        Scanner keyboard = new Scanner(System.in);`
`        double rcostTicket`	`        double rcostTicket;`
`        int iAge`	`        int iAge;`
`        //input`	`        //input`
`        display "Enter age: "`	`        System.out.print("Enter age: ");`
`        enter iAge`	`        iAge = keyboard.nextInt();`
`        //decision and output`	`        //decision and output`
`        if (iAge <= 18) then`	`        if (iAge <= 18)`
`            begin`	`        { //begin if-block`
`                rcostTicket = 45.50`	`            rcostTicket = 45.50;`
`            end`	`        } //end if-block`
`        else`	`        else`
`            begin`	`        { //begin else-block`
`                rcostTicket = 60`	`            rcostTicket = 60;`
`            end`	`        } /end else-block`
`        display "You need to pay R" + rCostTicket`	`        System.out.println("You need to pay R" + rCostTicket);`
`    end //main method`	`    } //end main method`
`end //end of the class`	`} //end class`

In this example, the relational expression *iAge <= 18* is used with an if..else statement in Java.

If the answer to the question is ‘Yes’, we say the result of the relational expression is *true*, and then the statement(s) in the *if block* will be executed.

If the result of the relational expression is *false* (answer to the question is ‘No’), the statement(s) in the *else block* will be executed.

4.8 Notes on the if ... else statement

- When you consult various sources, you will see that programmers format the if ... else statement in different ways (notice where the braces { } are placed).

```
if (rTemp < 20) { //beginning of if-block
     System.out.println("Wear a long-sleeve shirt.");
}    // end of if-block
else { //beginning of else block
     System.out.println("Wear a T-shirt.");
} //end of else-block
```

- A logical expression should always be placed in parenthesis, for example (rTemp < 20).
- Depending on the scenario and what you want to display, you may want to process more than one statement in a block, for example:

Two statements in the if block.

```
rPocketMoney = 500;
if (rFinalMark >= 75)
{
      System.out.println("Well done – distinction – you get R100 bonus! ");
      rPocketMoney += 100; //Remember this is the same as rPocketMoney = rPocketMoney + 100;
}
else
{
      System.out.println("Sorry – no bonus for you :-( ");
{
```

- In some programs you may not need an else block. Look at this example's flow chart and Java code:

  The cost of an item is R500. The user needs to indicate whether a person is a student, or a working adult. If the person is a student, 10% of the cost will be given as discount. If it is a working adult, the discount will stay 0, and the cost of the item will stay 500. In this program the discount and the cost are initialized when the variables are declared.

  - When a block contains only one statement, you do not need to use braces { } to indicate the start and end of the block. For example, the following pseudo code and Java code segment are correct.

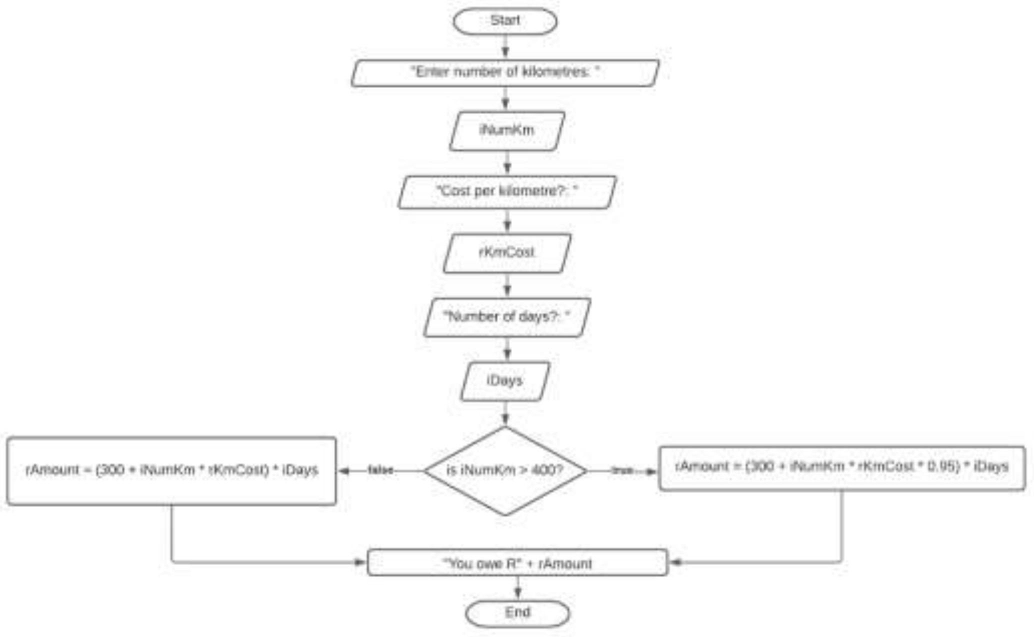
  - Segment

```
if (rTemp < 20)
     display "Wear a long-sleeve shirt."
else
     display "Wear a T-shirt."
```

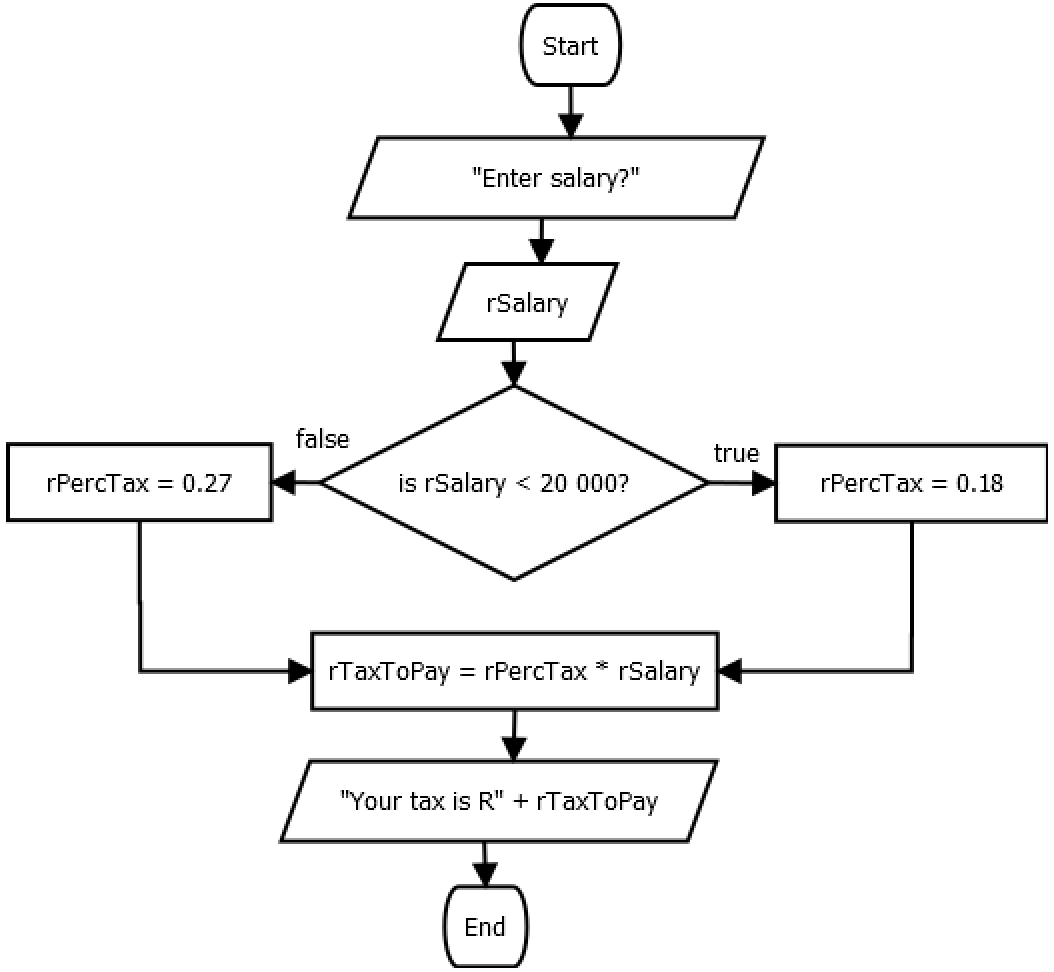
```
if (rTemp < 20)
     System.out.println("Wear a long-sleeve shirt.");
else
     System.out.println("Wear a T-shirt.");
```

Activity 5:   Create Java programs from the provided flowcharts

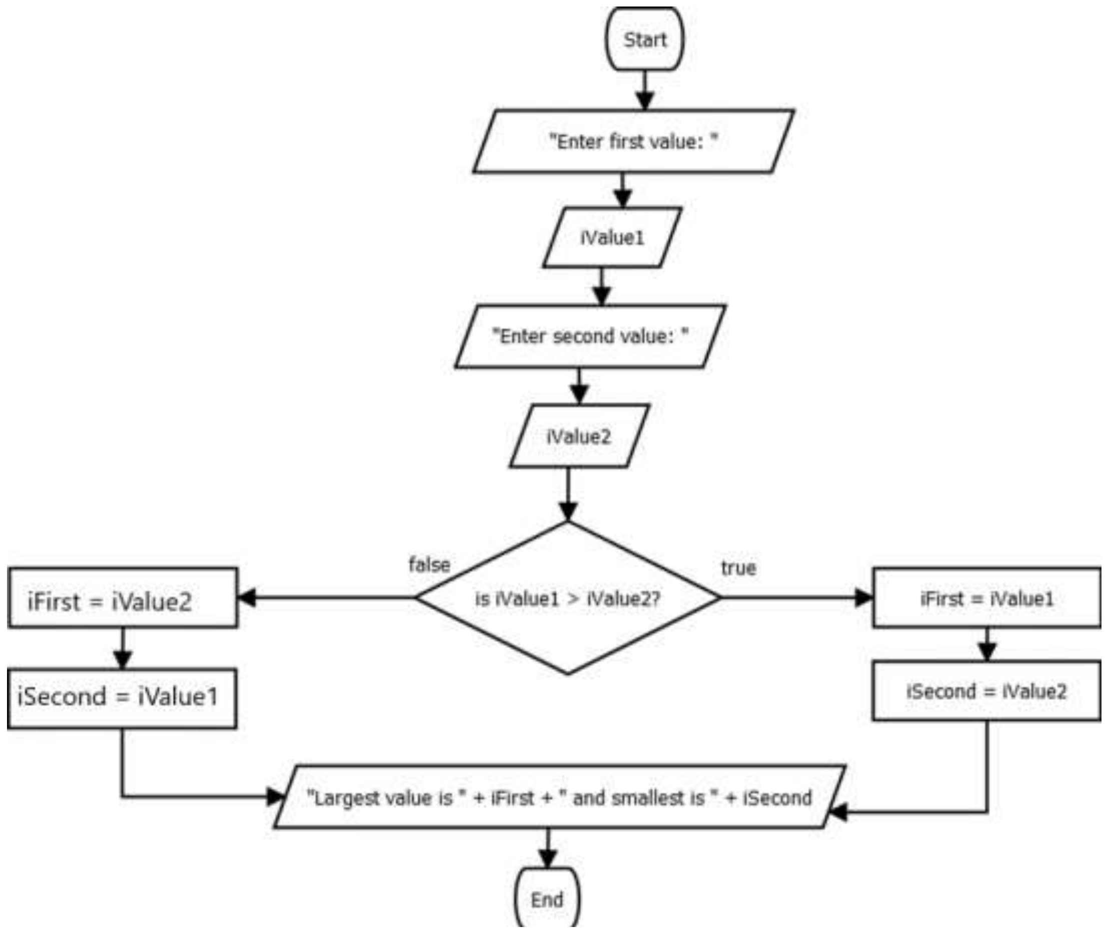
a) Henry rents a bakkie to move stock to a store. The basic cost is R300 per day. He also pays a specific amount per kilometre travelled. If the number of kilometres exceeds 400, he gets a discount of 5% on the amount for the kilometres travelled. Enter the number of kilometres travelled, the amount he should pay per kilometre, as well as the number of days that he rented the trailer, then calculate and display the amount due by him.


Start
"Enter number of kilometres: "
iNumKm
"Cost per kilometre?: "
rKmCost
"Number of days?: "
iDays
is iNumKm > 400?
false
rAmount = (300 + iNumKm * rKmCost) * iDays
true
rAmount = (300 + iNumKm * rKmCost * 0.95) * iDays
"You owe R" + rAmount
End

b) Enter an employee’s salary and determine and display the tax payable by the employee. If the employee earns less than R20 000 per month, the percentage tax is 18%, and for all other salaries the percentage tax is 27%.


Start
"Enter salary?"
rSalary
is rSalary < 20 000?
false
true
rPercTax = 0.27
rPercTax = 0.18
rTaxToPay = rPercTax * rSalary
"Your tax is R" + rTaxToPay
End

c) Enter two integers. Determine, and display the largest and the smallest values. The largest value should be stored in the variable *iFirst* and the smallest value should be stored in the variable *iSecond*. Assume the two numbers will never be equal.



Activity 6: Predict the output of code segments

1a)

Let rNumerator = 5; and rDenominator = 20.

1b)

Let rNumerator = 3; and rDenominator = 0.

```java
package PreventDivideZero;
import java.util.Scanner;
public class PreventDivideZero
{ //start class
     public static void main(String[] args)
     { //start main method
          Scanner keyboard = new Scanner(System.in);
          double rNumerator, rDenominator, rQuotient;

          System.out.print("Enter the numerator: ");
          rNumerator = keyboard.nextDouble();
          System.out.print("Enter the denominator: ");
          rDenominator = keyboard.nextDouble();

          if (rDenominator == 0)
          {
               System.out.println("You cannot divide by zero!");
          }
          else
          {
               rQuotient = rNumerator / rDenominator;
               System.out.println("Quotient is: " + rQuotient);
          }
     } //end main method
} //end class
```

2 a)

Let iYearOfBirth = 1998; and iCurrentYear = 2021.

2 b)

Let iYearOfBirth = 2021; and iCurrentYear = 2000.

```java
package CheckBirthYear;
import java.util.Scanner;
public class CheckBirthYear
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iYearOfBirth, iCurrentYear, iAge;

        System.out.print("In which year were you born? ");
        iYearOfBirth = keyboard.nextInt();

        System.out.print("What year is it currently? ");
        iCurrentYear = keyboard.nextInt();

        if (iYearOfBirth > iCurrentYear)
        {
            System.out.println("You cannot be born in the future!");
        }
        else
        {
            iAge = iCurrentYear - iYearOfBirth;
            System.out.println("You turn " + iAge + " in " + iCurrentYear);
        }
    } //end main method
} //end class
```

3 a)
 	
Let iNumber = 20; and iFactor = 3.
 	
3 b)
 	
Let iNumber = 20; and iFactor = 5
	```
	package FactorOf;
	import java.util.Scanner;
	public class FactorOf
	{ //start class
	     public static void main(String[] args)
	     { //start main method
	          Scanner keyboard = new Scanner(System.in);
	          int iNumber, iFactor;

	          System.out.print("What is the big number?: ");
	          iNumber = keyboard.nextInt();

	          System.out.print("Possible factor of " + iNumber + "?: ");
	          iFactor = keyboard.nextInt();

	          if (iNumber % iFactor == 0)
	          {
	               System.out.println(iFactor + " is a factor of " + iNumber);
	          }
	          else
	          {
	               System.out.println(iFactor + " is NOT a factor of " + iNumber);
	          }
	     } //end main method
	} //end class
	```

Activity 7: Create a flowchart for if-statement scenarios, then write a Java program

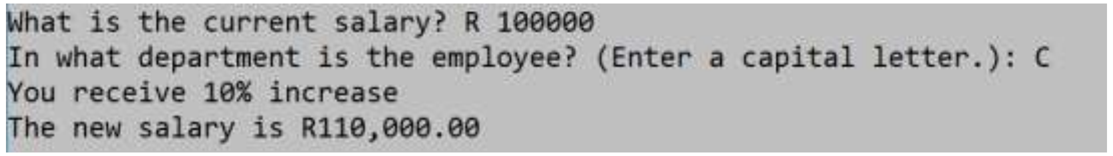
a) A grocery store wants to increase all items in the store by 5%, except those in the perishable department, department 3, that must be increased by 7%. Enter the price of an item and the department number it falls in. Calculate the new price of the item and display it on the screen.

b) A number of people go to a Parlotones Concert. The price of a ticket is R550, but if a group of more than 35 wants to buy tickets, the price is only R500 per ticket. Enter the number of people in the group. Calculate and display the total amount for all the tickets.

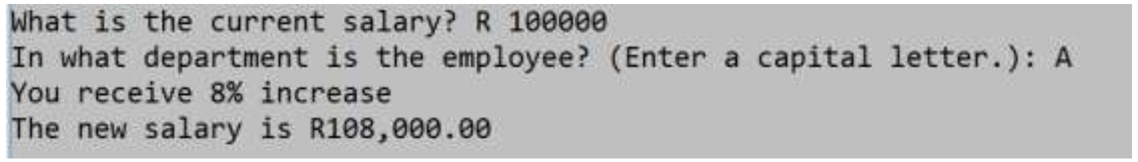
Activity 8: Create a flowchart for if..else statement scenarios, then write a Java program

a) Suppose all the employees in a company receives an 8% increase but the employees from Department C receive an increase of 10%, as they performed better than the others. Read the current salary and department name (a capital letter), then calculate and display the increased salary.

These are examples of input and output.



```
What is the current salary? R 100000
In what department is the employee? (Enter a capital letter.): C
You receive 10% increase
The new salary is R110,000.00
```

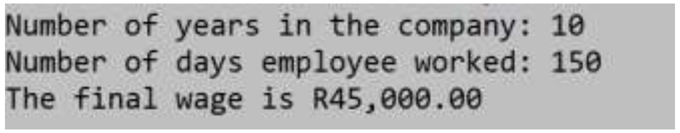


```
What is the current salary? R 100000
In what department is the employee? (Enter a capital letter.): A
You receive 8% increase
The new salary is R108,000.00
```

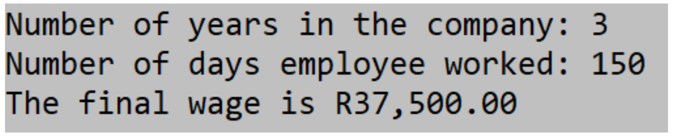
b) The ABC Cell Phone Company charges customers a basic rate of R60 per month to send text messages. For the first 100 messages per month, an additional 5c per message is charged. For all messages more than 100, an additional 10c per message is charged. Enter the number of messages texted by a customer in a month and calculate and display the bill amount for the customer. (Tip: draw a diagram to illustrate the price structure.)

c) An employee who has worked for at least 5 years earns R300 per day, while an employee who has been in service for less years only earns R250 per day. Read the number of years an employee worked in a company, as well as the number of days the employee worked. Then calculate and display the final wage of the employee.

These are examples of input and output.



```
Number of years in the company: 10
Number of days employee worked: 150
The final wage is R45,000.00
```



```
Number of years in the company: 3
Number of days employee worked: 150
The final wage is R37,500.00
```

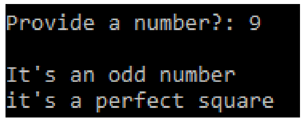
Activity 9: Create pseudo code for if..else-statement scenarios, then write a Java program

a) Write a program that will read a number. The program should then report:

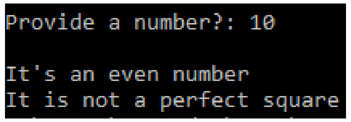
- whether the number is even or odd
- whether the number is a perfect square or not

Note:

- An even number is a number that is divisible by 2 without a remainder.
- A number is a perfect square if the square root of the number has no decimal part (therefore the square root is the same as the rounded square root).

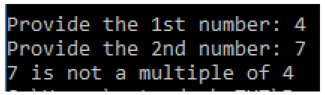


```
Provide a number?: 9

It's an odd number
it's a perfect square
```

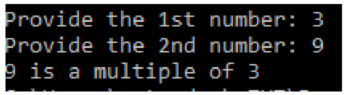


```
Provide a number?: 10

It's an even number
It is not a perfect square
```

b) Create a program which will ask the user to enter two values. The program must determine whether the second value is a multiple of the first value or not. For example, if the values 3 and 12 are entered, the program should report: '12 is a multiple of 3'. If the values 3 and 13 are entered, the program should report: '13 is not a multiple of 3'.

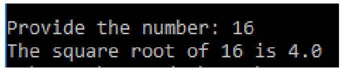


```
Provide the 1st number: 4
Provide the 2nd number: 7
7 is not a multiple of 4
```

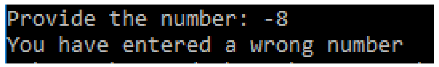


```
Provide the 1st number: 3
Provide the 2nd number: 9
9 is a multiple of 3
```

c) Write a program where the user has to enter a number, and the program should calculate and display the square root of the number if the number entered is positive. If the number entered is negative, an error message should be displayed.

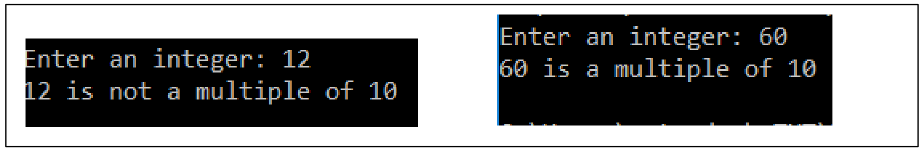


```
Provide the number: 16
The square root of 16 is 4.0
```



```
Provide the number: -8
You have entered a wrong number
```

d) Write a program that will prompt the user to enter an integer. The program should determine
if the integer is a multiple of 10 or not.

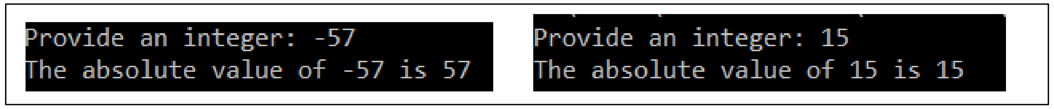


```
Enter an integer: 12
12 is not a multiple of 10
```

```
Enter an integer: 60
60 is a multiple of 10
```

e) Write a program that will print the absolute value of an integer entered. For example

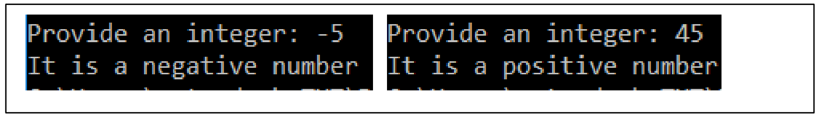
input: -57	output: 57
input: 15	output: 15



```
Provide an integer: -57
The absolute value of -57 is 57
```

```
Provide an integer: 15
The absolute value of 15 is 15
```

f) Write a program that will check whether the number entered by the user is negative or positive.



```
Provide an integer: -5
It is a negative number
```

```
Provide an integer: 45
It is a positive number
```

4.9 Boolean expressions (Compound conditions)

In some programming scenarios, it may be necessary to combine a number of conditions. This also happens in real-life scenarios. For example: If your homework is done, and your room is clean, you may go out with friends.

You can write this in a more condensed way:

```
if (your homework is done) AND (your room is clean)
    display "You can go out with friends"
else
    display "You must stay home"
```

Using the word 'AND' is part of the way we speak. We also say: "You can go out with friends if you wash the car OR if you mow the lawn.". We also say: "If your room is dirty you are NOT going out tonight".

In programming, AND, OR and NOT are called *logical operators / Boolean operators*, and they are used to connect two or more relational expressions into one. Such a combination of relational expressions through logical operators are called *Boolean expressions / compound conditions*.

This is a summary of the logical operators available in Java, and their effects (Gaddis, 2010).

Logical operator	**Meaning**	**Effect**	
&&	AND	Binary operators. Connects two relational expressions into one.	All expressions must be true for the overall expression to be true.
||	OR		It is only necessary for one expression to be true, for the overall expression to be true.
!	NOT	Unary operator. Works with one expression.	Reverses the Boolean expression.

Here are some examples of Boolean expressions and their results:

Boolean expression	**Result**	**Boolean expression**	**Result**
true && true	true	true || true	true
true && false	false	true || false	true
false && true	false	false || true	true
false && false	false	false || false	false
! true	false		
! false	true		

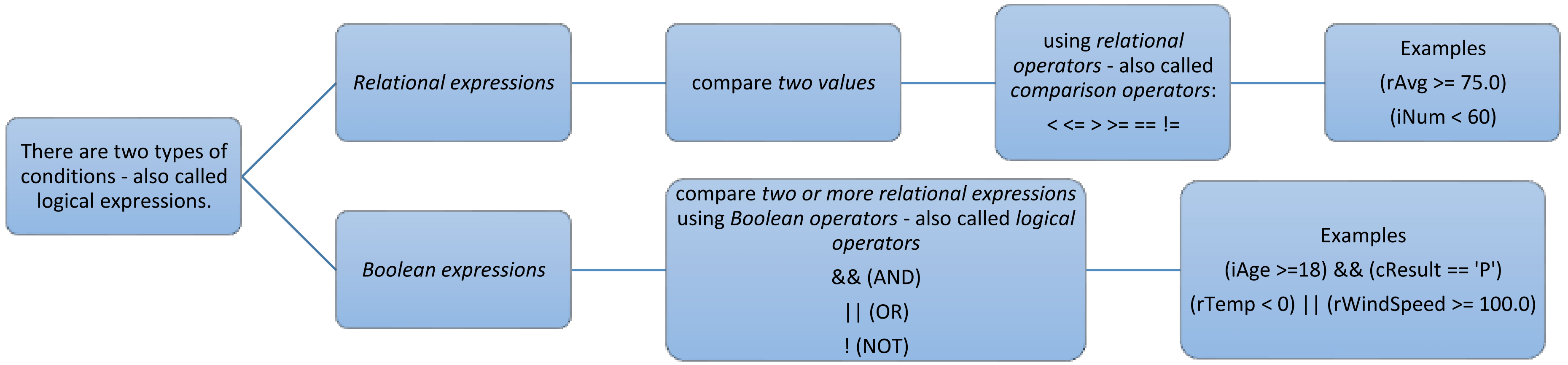
Activity 10: Evaluate Boolean expressions

Evaluate the following Boolean expressions.

Code segments with Boolean expressions			Result (true / false)
a)	double rSalary = 4000; int iNumberYears = 3;	(rSalary >= 3000 && iNumberYears >=2)	
b)	double rSalary = 1500; int iNumberYears = 3;	(rSalary >= 3000 && iNumberYears >=2)	
c)	double rSalary = 2000; int iNumberYears = 2;	(rSalary >= 3000 && iNumberYears >=2)	
d)	Int iResult=65; int iSubPassed=5	(iResult>49 || iSubPassed>3)	
e)	Int iResult=25; int iSubPassed=0	(iResult>49 || iSubPassed>3)	
f)	Int iResult=49; int iSubPassed=4	(iResult>49 || iSubPassed>3)	
g)	Double rHeight=1.72	! (rHeight>2)	
h)	Int iNumber1=45; Int iNumber2=12;	(iNumber1>iNumber2 &&iNumber2>25)	
i)	int iSides=3;	! (iSides>=3)	

4.10 Terminologies

Various sources refer to concepts by various names. The following diagram illustrates the relationship between a number of terminologies referring to logical expressions.


There are two types of conditions - also called logical expressions.
Relational expressions
compare *two values*
using *relational operators* - also called *comparison operators*:
< <= > >= == !=
Examples
(rAvg >= 75.0)
(iNum < 60)
Boolean expressions
compare *two or more relational expressions* using *Boolean operators* - also called *logical operators*
&& (AND)
|| (OR)
! (NOT)
Examples
(iAge >=18) && (cResult == 'P')
(rTemp < 0) || (rWindSpeed >= 100.0)

Activity 11: Create pseudo code for scenarios with Boolean expressions, then write a Java program

a) Students are selling tickets for entrance to a festival. They receive 10% of their ticket sales as payment. The price of student tickets is R50.00 per ticket and lecturers' tickets are R60.00 per ticket. If a student sells more than 10 lecturer tickets and more than 20 student tickets, he/she receives 12% of his/her ticket sales as payment. Write a program to enter the number of student tickets and the number of lecturer tickets sold. The program must calculate and display the total ticket sales and the payment the student will receive.

b) The speed limit on a road is 80km per hour. When the speed limit has been violated up to 120km per hour, a fine of R250.00 is imposed for every 10km per hour beyond the speed limit. For every 10 km per hour the speed limit on this road is exceeded beyond 120 km per hour, the fine is R500 per 10km per hour. Enter the speed the vehicle travelled and display the amount that needs to be paid if the speed limit has been violated. Display a suitable message if the speed limit has not been violated.

Example: For a speed of 100km per hour the fine will be R500.00

For a speed of 130km per hour the fine will be R 1000.00 for the speed up to 120km per hour + R500.00 = R1500.00

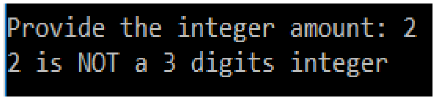
c) Write a program that will generate two random numbers. The program should prompt the user to enter the sum of the two numbers. The program will then determine if the user's answer is correct and display a suitable message when the user is wrong or correct.

d) Write a program that will ask the user to enter two integers that represent XY coordinates of a given point. X coordinate is stored in iOne and Y is stored in iTwo. The program should determine if the point is in the second quadrant or not. Display a suitable message.

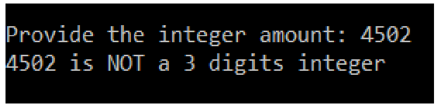
   Note: A point belongs to the second quadrant when X is negative, and Y is positive

e) Ask the user to enter an integer amount they would like to deposit into their account. If the amount is a 3 digits number, display a message: the number is a 3 digits integer otherwise display a message that the number is not a 3 digits integer.

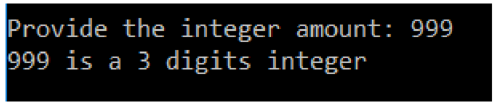
Example of output



```
Provide the integer amount: 2
2 is NOT a 3 digits integer
```



```
Provide the integer amount: 4502
4502 is NOT a 3 digits integer
```



```
Provide the integer amount: 999
999 is a 3 digits integer
```

4.11 Precedence and associativity of relational and Boolean operators

Boolean operators also have orders of precedence.

!	NOT
&&	AND
||	OR

The operand of the ! operator usually needs to be placed in parentheses. For example, look at the following expressions (Gaddis, 2010, p. 165):

`int x = 4;`	
`if !(x > 2)`	`if !x > 2`
This expression will evaluate to false. (x > 2) is true. Therefore !(x>2) is !true = false So, the question was actually: "is x not greater than 2?"	This expression attempts to apply the ! to x only. However, the ! operator can only be applied to a Boolean value, and x is an int value. Therefore, this expression will cause a compiler error.

The following table contains a summary of all the operators we have discussed so far (Gaddis, 2010, p. 166):

Order of Precedence	Operators (processed from left to right)	Description
1.	`-  !`	Unary negation, logical NOT
2.	`*  /  %`	Multiplication, division, modulus
3.	`+  -`	Addition, subtraction
4.	`<  >  <=  >=`	Less than, greater than, less than or equal to, greater than or equal to
5.	`==  !=`	Equal to, not equal to
6.	`&&`	Logical AND
7.	`||`	Logical OR
8.	`=  +=  -=  *=  /=  %=`	Assignment and combined assignment

Look at the following expressions (Gaddis, 2010, p. 166):

(a > b) && (x < y)	is the same as	a > b && x < y
(x == y) || (b < a)	is the same as	x == y || b < a
a < b || y == z && m > j	is the same as	(a < b) || ((y == z) && (m > j)) == has the highest precedence, therefore y ==z will be evaluated first. && has a higher precedence than ||, therefore (m>j) will be evaluated second.

Activity 12: Boolean expressions taking precedence into account

a) Students at a technical college can be promoted to the next year of their studies without having to write the examination papers if:

- they passed their semester test <u>and</u> project and
- obtained at least 80% for either semester test <u>or</u> for the practical project.

Display a message that indicates whether the student will be promoted without having to write the examination paper or not.

b) When you are given three sticks, you may or may not be able to arrange them in a triangle. To make a triangle each stick should be less than the sum of the other two sticks. Write a program that will ask the user to enter three integers and outputs whether you can or cannot make a triangle with the three lengths.

c) Enter three integers and store them in iValue1, iValue2 and iValue3 .If iValue1 is greater than iValue2 but less than iValue3, or iValue2 is equal to iValue3 increase the quotient of iValue1 and iValue2 by 5%, else decrease the product of iValue1 and iValue3 by 12.5%

d) During the happy hour, Axen.inc employees are playing a game and are divided in two teams. Team A has employees who have 5 to 10 years' experience or those who work at least 6 hours a day. The rest of the employees belong to team B. The user must provide the number of years of experience and the number of hours they work per day. The program will display in which team the employee belongs.

e) Enter four values. If the first value is not greater than the second value or the 2nd value is not less than the 4th value and equal to the 3rd value display" correct values" otherwise display" wrong values".

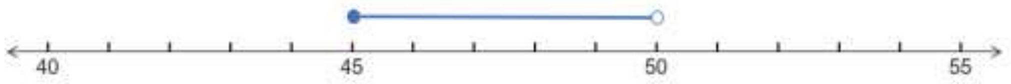
4.12 Checking numeric or character ranges with Boolean operators

- Number line representation
- Interval notation
- Set builder notation
- Include
- Exclude
- Between

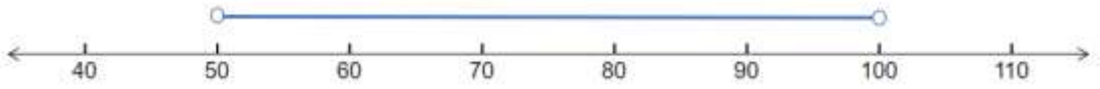
When we need to test if a value falls within a specific range, we will use descriptions such as those listed in the following table.

Description	Number line representation Interval notation Set builder notation	Java if statement with Boolean expression

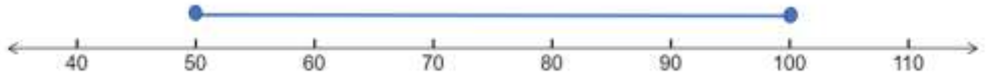
When the percentage (a decimal value) is 45 or higher and below 50, the student should write a supplementary exam.	[45,50)
	
	{x| 45 ≤ x < 50, x∈R}
		`if (rPerc >= 45 && rPerc < 50)`
		`   System.out.println("Write supplementary");`

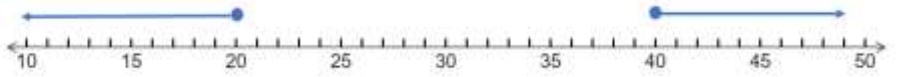
An integer value between 50 and 100.
 	
(If it is not mentioned specifically that both numbers are included, then 'between' means both numbers are excluded.)
	(50, 100)
	
	{x| 50 < x < 100, x∈Z}
		`if (iValue > 50 && iValue < 100)`
		`   System.out.println("Between 50 and 100);`

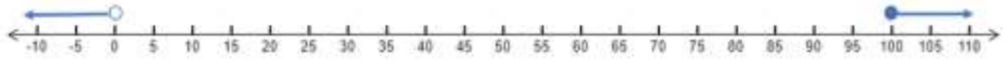
A decimal value between 50 and 100 (both included).	[50, 100]
	
	{x| 50 ≤ x ≤ 100, x∈Z}
		`if (iValue >= 50 && iValue <= 100)`
		`   System.out.println("Between 50 and 100);`

The decimal value should not be between 20 and 40. You can also say the value is 20 or less, *or* it is 40 or more.	{x | x ≤ 20 or x ≥ 40, x∈R}	`if (rValue <=20 || rValue >= 40)`
		`   System.out.println("Valid value");`

The integer value should be less than zero, or greater or equal to 100.	{x | x < 0 or x ≥ 100, x∈R}	`if (rValue < 0 || rValue >= 100)`
		`   System.out.println("Valid value");`

Did you notice?

- When did you use the Boolean operator AND, and when the OR? When you had to determine whether a value is in a range or not?
- When do you use the operator >, and when ≥?

Activity 13: Checking numeric ranges with Boolean operators

a) Write java statements for the following set builders' notations or intervals (use rValue to store the value of x) and predict the results.

Interval notation / Set builder notation	Java if statement with Boolean expression	Value of rValue	Result

{x| 1 < x <15, x∈N}	**if (rValue > 1 && rValue <15)
	System.out.println("Valid value");**
		rValue= 1	**False**

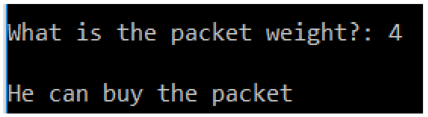
{x| x < 10 or x >15, x∈N}		rValue=45	
[57,78) (x ∈ R)		rValue=78	
{x| -25 < x < 27, x∈Z}		rValue=0	
(−∞, 2] U (7, +∞) (x ∈ R)		rValue=3	

b) The tickets for the school concert are being sold. Children less than 5 years old are not allowed. The normal ticket price is R50. Senior citizens (more than 60 years old) and children from 5 to 12 years old receive 20 % discount.

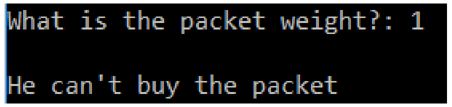
- Write Java if statement that will test who qualifies for the discount. Display a message that indicates whether the person receives a discount or not.
- Assuming all the ages are integers, provide the Number line representation, set builder notation and the interval notation that represent different age groups that qualify for the 20% discount.

c) Stephan owns a restaurant; he needs to buy some sugar for his restaurant. He can only buy a packet that can weigh at least 1.5 kg with a maximum of 5kg. Write Java if statement that will tell the user if Stephan is taking a specific packet of sugar or not. The user will enter the weight of a packet in kilogram and the program will display if Stephan can buy it or not.

Output examples



```
What is the packet weight?: 4

He can buy the packet
```



```
What is the packet weight?: 1

He can't buy the packet
```

4.13 Conditional (ternary) operator (enrichment – not for examination)

- Ternary

Java has an operator that can be used to write an if .. else statement in a shorter way. The operator is called the *conditional operator*. It works on three operands – therefore it is also called the ternary operator. The operator uses two symbols - ? and :

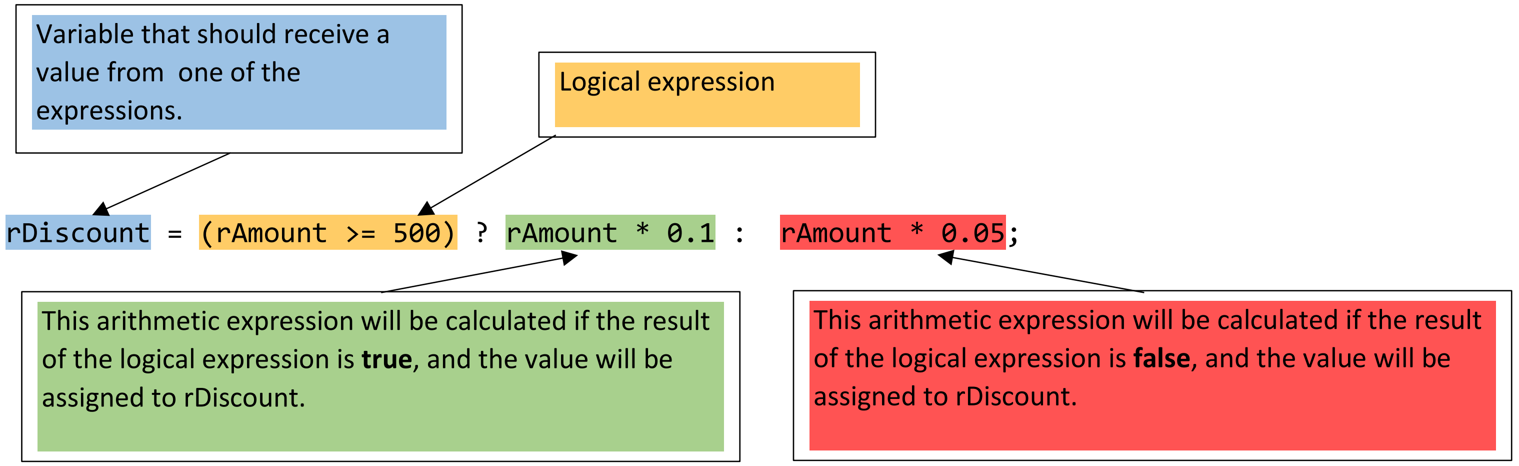
It has the following syntax:

```
variable = (Logical expression) ? expression1 : expression2;
```

For example, let us see how the if .. else statement can be translated/converted to a statement using the conditional operator.

if .. else statement	**Statement using conditional operator**

`if (rAmount >= 500)`
`{`
`     rDiscount = rAmount * 0.1;`
`}`
`else`
`{`
`     rDiscount = rAmount * 0.05;`
`}`
`System.out.println("Discount = R" + rDiscount);`
	`rDiscount = (rAmount >= 500) ? rAmount * 0.1 :  rAmount * 0.05;`
	
	OR
	
	`rDiscount = (rAmount >= 500)`
	`     ? rAmount * 0.1 //the if block`
	`     :  rAmount * 0.05; //the else block`
	`System.out.println("Discount = R" + rDiscount);`



Variable that should receive a value from one of the expressions.

Logical expression

```
rDiscount = (rAmount >= 500) ? rAmount * 0.1 :  rAmount * 0.05;
```

This arithmetic expression will be calculated if the result of the logical expression is **true**, and the value will be assigned to rDiscount.

This arithmetic expression will be calculated if the result of the logical expression is **false**, and the value will be assigned to rDiscount.

Activity 14: Rewrite if .. else statements using the ternary operator

If…else statement	**Statement using conditional operator**

`if ( iAge >=65)`
`{`
`      sCategory= "Pensioner"`
`}`
`else`
`{`
`        sCategory= "Normal"`
`}`
`System.out.println(sCategory)`	

`if !(iNum%2==0)`
`{`
`    iSum=iNum+1;`
`}`
`else`
`{`
`    iSum=iNum+2;`
`}`
`System.out.println("The sum is  " + iSum);`	

`if (100>iGameTrial)`
`{`
`    iGameTrial=iGameTrial-1`
`}`
`else`
`{`
`    iGameTrial=0`
`}`
`System.out.println("Remaining trials for the game = " + iGameTrial);`	

4.14 Using 'flags'

A variable that holds a Boolean value, is called a *Boolean* variable. Such a variable can only hold one of two values – true, or false. Boolean values are literals, the same as a value that can for example be 10. A Boolean value can be declared and initialised as follows:

```
boolean  bPassedSubject = true;
```

Note: the words true and false are keywords in Java. You cannot use them as variable names. You should not write bPassedSubject = "true"; No inverted commas are needed.

One of the ways Boolean variables can be used, is to create a 'flag' – a Boolean value containing the result of a Boolean expression. These flags can then be combined in another expression to make an if .. else statement easier to read. The following is an example of such an application: (Example obtained from (Wassermann, et al., 2014, p. 86).)

A student qualifies for a bursary if the following three criteria are met:

- An average grade of 65% or higher for all four marks.
- None of the grades may be less than 50% (you can also say all grades must be 50 or higher).
- At least one of the grades must be 70% or higher.

The following code can be used to solve this problem:

```
DecideBursary
begin //start of the class
    main method
    begin //main method

        boolean bFlag1, bFlag2, bFlag3
        int iMark1, iMark2, iMark3, iMark4
        double rAvg
        //Input - read all four test marks
        // display .. enter not shown here
        //Calculate the average mark
        rAvg  = (iMark1 + iMark2 + iMark3 + iMark4) / 4

        // Set the first flag: Average must be 65 or higher
        if rAvg >= 65
             bFlag1 = true
        else
             bFlag1 = false

        //Set the second flag: All grades must be 50 or higher
        if (iMark1 >= 50) AND (iMark2 >= 50) AND (iMark3 >= 50) AND (iMark4 >= 50)
             bFlag2 = true
        else
             bFlag2 = false
        //Set the third flag: At least one grade must be 70 or higher
        if (iMark1 >= 70) OR (iMark2 >= 70) OR (iMark3 >= 70) OR (iMark4 >= 70)
             bFlag3 = true
        else
             bFlag3 = false
        //Test if all three flags have been 'set'
        if (bFlag1 = true AND bFlag2 = true AND bFlag3 = true)
             display "Bursary awarded"
        else
             display "NO Bursary"
    end //main method
end //end of the class
```

The three if statements produce quite a number of lines of code. However, there are shorter ways to code this in Java. Two variations of this program will be provided.

Java code for flag example: Version 1

```java
import java.util.Scanner;
public class DecideBursaryV1
{ //start class
     public static void main(String[] args)
     { //start main method

          int iMark1 = 50;
          int iMark2 = 70;
          int iMark3 = 80;
          int iMark4 = 90;
          boolean bFlag1 = false;
          boolean bFlag2 = false;
          boolean bFlag3 = false;

          //Calculate the average mark
          double rAvg  = (iMark1 + iMark2 + iMark3 + iMark4) / 4.0;
          // Set the first flag: Average must be 65 or higher
          if (rAvg >= 65) bFlag1 = true;
          //Set the second flag: All grades must be 50 or higher
          if (iMark1 >= 50 && iMark2 >= 50 && iMark3 >= 50 && iMark4 >= 50) bFlag2 = true;
          //Set the third flag: At least one grade must be 70 or higher
          if (iMark1 >= 70 || iMark2 >= 70 || iMark3 >= 70 || iMark4 >= 70) bFlag3 = true;
          //Test if all three flags have been 'set'
          if (bFlag1 && bFlag2 && bFlag3)
               System.out.println( "Bursary awarded");
          else
               System.out.println( "NO Bursary");
     } //end main method
} //end class
```

You can adapt the program to read four values form the keyboard.

Set all the flags to false.

You can also program the last if statement like this:

```java
if (bFlag1 == true && bFlag2 == true && bFlag3 == true)
```

but clearly it is a lot of additional, unnecessary code.

Java code for flag example: Version 2

You can also write this program using the conditional operator.

```
import java.util.Scanner;
public class DecideBursaryV2
{ //start class
     public static void main(String[] args)
     { //start main method

          int iMark1 = 50;
          int iMark2 = 70;
          int iMark3 = 80;
          int iMark4 = 90;
          //Calculate the average mark
          double rAvg  = (iMark1 + iMark2 + iMark3 + iMark4) / 4.0;
          // Set the first flag: Average must be 65 or higher
          boolean bFlag1 = (rAvg >= 65) ? true : false ;
          //Set the second flag: All grades must be 50 or higher
          boolean bFlag2 = (iMark1 >= 50 && iMark2 >= 50 && iMark3 >= 50 && iMark4 >= 50) ? true : false;
          //Set the third flag: At least one grade must be 70 or higher
          boolean bFlag3 = (iMark1 >= 70 || iMark2 >= 70 || iMark3 >= 70 || iMark4 >= 70) ? true : false;
          //Test if all three flags have been 'set'
          if (bFlag1 && bFlag2 && bFlag3)
               System.out.println( "Bursary awarded");
          else
               System.out.println( "No Bursary");
     } //end main method
} //end class
```

Activity 15: Using flags with Boolean operators

a) A university has the following requirement for engineering students:

- The applicant must have had Mathematics and Physical Science in grade 12 and each of these subjects should be above 70%.
- At least four subjects must each be above 80%.
- No subject may be lower than 60%.
- The average of the learner's marks should be at least 65%.

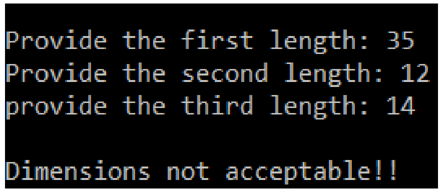
The input of the program is the learner's six subjects (excluding LO) and the percentage obtained for each subject. Write a program using flags, to determine if a learner qualifies for engineering studies. Display all subjects and percentages obtained. Use the following data to test your program:

Learner 1	Maths 80%	Phys Sc 75%	Afr 65%	Eng 60%	Geo 78%	IT 80%	Result=
Learner 2	Afr 83%	Eng 84%	Maths 88%	Phys Sc 82%	IT 72%	Music 61%	Result=

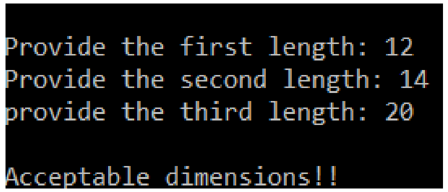
b) Students need to design a triangular photo frame for the art project. The three lengths of the frame (integers in centimetres) need to be entered. The design should respect the following conditions:

- A side should not be more than the sum of the other two sides.
- The perimeter of the triangle should not be above 100 centimetres.

Example of outputs



```
Provide the first length: 35
Provide the second length: 12
provide the third length: 14

Dimensions not acceptable!!
```



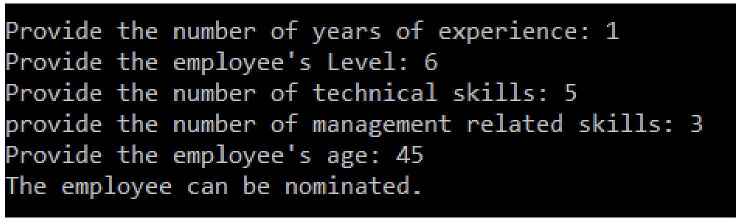
```
Provide the first length: 12
Provide the second length: 14
provide the third length: 20

Acceptable dimensions!!
```

c) The employees of Alx&Co are being nominated for the best employee of the year award. To qualify for the nomination:

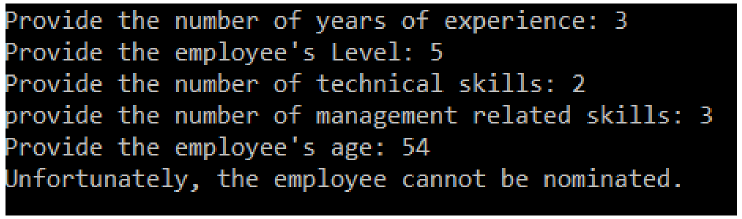
- The employee should have at least 2 years of experiences or should be at level 5 or 6.
- The employee should have at least 3 technical skills or at least 5 project management related skills.
- The employee should not be more than 60.

The number of years of experience and the employee's level should be entered. The user must provide the number of technical skills as well as the number of management related skills. The age of the employee must also be entered.

Example of output



```
Provide the number of years of experience: 1
Provide the employee's Level: 6
Provide the number of technical skills: 5
provide the number of management related skills: 3
Provide the employee's age: 45
The employee can be nominated.
```



```
Provide the number of years of experience: 3
Provide the employee's Level: 5
Provide the number of technical skills: 2
provide the number of management related skills: 3
Provide the employee's age: 54
Unfortunately, the employee cannot be nominated.
```

4.15 Comparing String objects

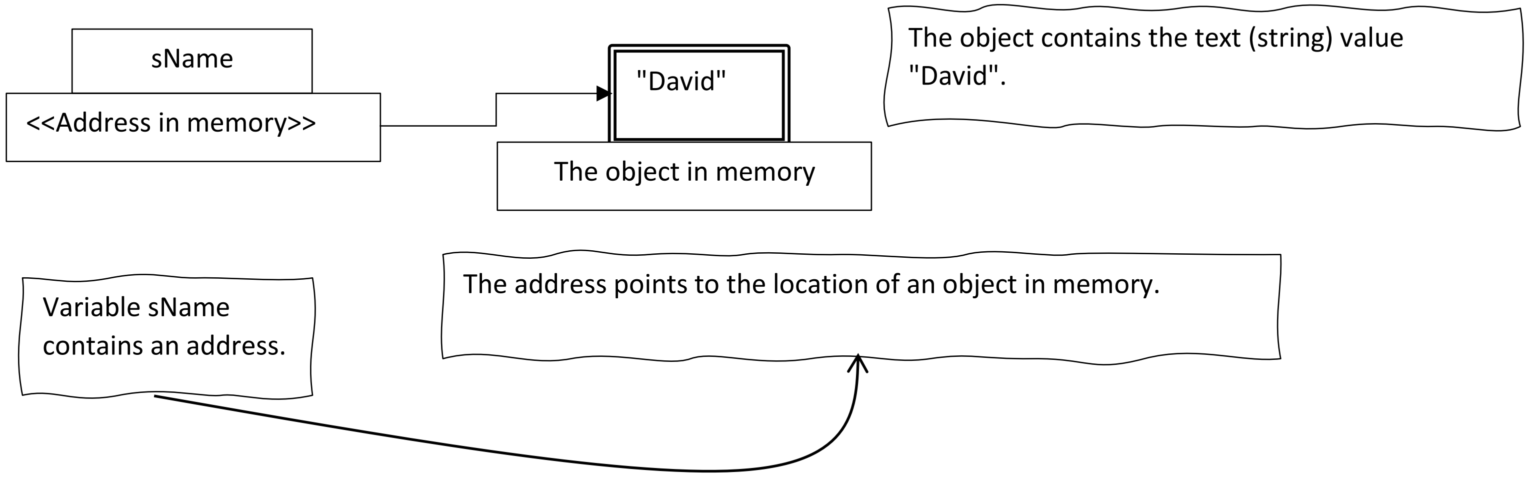
4.15.1 String equality

Remember that a String is not a primitive data type, it is a class. Therefore, when we do the following declaration:

```
String sName = "David";
```

we actually created a String object, and the variable sName contains the address of the object in the computer's memory. In programming terminology, we say the variable sName references the object, therefore sName is a *reference variable*.

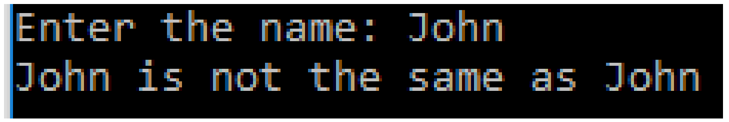
The relationship between the reference variable sName, and the String object containing the string "David" can be illustrated as follows:


sName
<<Address in memory>>
"David"
The object in memory
The object contains the text (string) value "David".
Variable sName contains an address.
The address points to the location of an object in memory.

This can create problems when you want to compare strings. Look at the following program:

```
import java.util.Scanner;
public class CompareStrings
{ //start class
     public static void main(String[] args)
     { //start main method
          Scanner keyboard = new Scanner(System.in);
          String sName1 = "John";
          String sName2;
          System.out.print("Enter the name: ");
          sName2 = keyboard.nextLine();


          if (sName1 == sName2)
          {
               System.out.println("Same name");
          }
          else
          {
               System.out.println(sName1 + " is not the same as " +
sName2);
          }
     } //end main method
} //end class
```

This will be the input and output of the program:



```
Enter the name: John
John is not the same as John
```

The reference variable sName1 contains the address of a String object that contains a string value "John".

The reference variable sName2 contains a different address that refers to another object that also contains the string value "John".

Therefore, sName1 can never be the same as sName2. The == operator compares the addresses (references) of the variables.

The best way to compare the string values of different String objects is to use methods of the class String. The previous program should therefore look as follows:

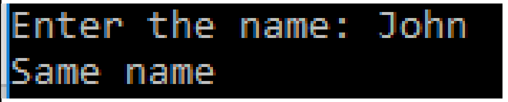
```
import java.util.Scanner;
public class CompareStringsBetter
{ //start class
     public static void main(String[] args)
     { //start main method
          Scanner keyboard = new Scanner(System.in);
          String sName1 = "John";
          String sName2;
          System.out.print("Enter the name: ");
          sName2 = keyboard.nextLine();
          if (sName1.equals(sName2))
               System.out.println("Same name");
          else
               System.out.println(sName1 + " is not the same as " + sName2);
     } //end main method
} //end class
```

The method equals() should be used to determine if the string values in the two String objects are the same.

Remember – you can also code the logical expressions as follows:

```
if (sName1.equals(sName2) == true)
```

The output will now be correct.



```
Enter the name: John
Same name
```

In summary

The **==** operator checks if both variables point to the same address (memory location) of a String object.

The method *equals()* determines whether the string values in the objects the variables are pointing to are the same.

Activity 16: Determine whether Strings have the same value

a) To register for an extended course, students need to choose between January and July as the starting month. Write a program that prompts the user to enter the month they wish to start the extended course. If they choose January, the program should display: "You chose round 1", if they choose July the program should display "You chose round 2". If they choose any other month, the program should display: "Error, you have chosen the wrong month".

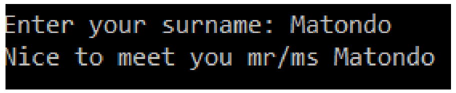
b) 4 students from different families are visiting the department. Write a program that will ask for the student 's surname and determine if it is the director's, the secretary's, or the judge's son. You can assume that all the students have different surnames.

- The director's surname is **Jones**
- The secretary's surname is **Ali**
- The judge's surname is **Dlamini**

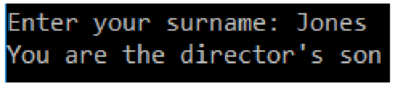
If they are not the judge's or the secretary or the director's child, the program should display a

a message with the student 's surname. E.g. if the student's surname is **Kambale**, the program should display: Nice to meet you **Kambale**

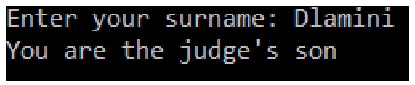
Example of output



```
Enter your surname: Matondo
Nice to meet you mr/ms Matondo
```



```
Enter your surname: Jones
You are the director's son
```



```
Enter your surname: Dlamini
You are the judge's son
```

4.15.2 *Equal, bigger or smaller*

Programs can be written to sort a list of surnames in alphabetical order – think of a class list. To do that, programming code must be able to determine that the string "Mabala" is smaller than "Mabidi", and therefore Mabala should be displayed before Mabidi. (You will learn how to write such programs in module B when you learn how to use arrays.)

Programming languages compare strings character by character to determine which one is the 'smallest' or 'biggest'. For example:

	M	a	b	a	l	a
Unicode value of the character. →	77	97	98	97 105		
	M	a	b	i	d	i

The first three letters are the same, therefore they are 'equal'. For the fourth letters in the string, the letter 'a' has a numeric value of 97 that is smaller than the value for 'i' that is 105. The comparison will stop, and the string "Mabala" is considered to be smaller than "Mabidi".

These comparisons do not depend on the number of characters in a string, for example:

"Joanne" is smaller than "John", even though "John" has less characters than "Joanne".

J	o	a
74	111	97 104
J	o	h

The comparison is done from the left until the first time a letter is different. The letter 'a' is smaller than 'h', therefore "Joanne" is smaller than "John".

Remember that lower case characters have larger Unicode values than uppercase letters. Therefore:

"HALLO" < "hallo"

Note: The 'size'of a string is not the same as the length of a string. "Mabala" and "Mabidi" both have 6 characters – so the length of each string is 6, but "Mabala" < "Mabidi".

Activity 17: Which strings are equal, smaller or bigger?

Fill in the correct relation operator for each of the following relational expressions. A few examples are supplied.

Example	*"John" > "Joan"*
Example	*"Joan" < "John"*
Example	*"Light" < "Line"*
a)	"copy"_____ "cope"
b)	"Past "_____"pass"
c)	"Linear"_____"Liar"
d)	"Order"_____"Ordinary"
e)	"**C**ommon"____"**c**ommunity"
f)	"**c**ells"___"**C**ellular"

Note: The comparison is read from left to right. "John" is bigger than "Joan". You can also say "Joan" is smaller than "John".

We saw earlier how you can use the method *equals()* of the String class to determine whether two strings are equal (exactly the same). For example:

```
if (sName1.equals(sName2))
     System.out.println("Same name");
else
     System.out.println(sName1 + " is not the same as " + sName2);
```

To determine if one string is equal, smaller or bigger than another string, we can use the method *compareTo()* from the String class. The following programming code demonstrates the way the method compareTo() works.

```
public class CompareStrings
{ //start class
    public static void main(String[] args)
    { //start main method
        String sName1 = "John";
        String sName2 = "Joanne";

        if (sName1.compareTo(sName2) == 0)
            System.out.println(sName1 + " is the same as " + sName2);

        if (sName1.compareTo(sName2) < 0)
            System.out.println(sName1 + " is smaller than " + sName2);

        if (sName1.compareTo(sName2) > 0)
            System.out.println(sName1 + " is bigger than " + sName2);

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

The method *compareTo()*has one argument – the variable name of a String object.

The method *compareTo()*returns a numeric value.

You can therefore also write a statement such as:

```
int iCode =  sName1.compareTo(sName2);
```

If the return value = 0, the two strings are the same.

If the return value < 0, the first string is smaller than the second string.

If the return value > 0, the first string is bigger than the second string.

Note

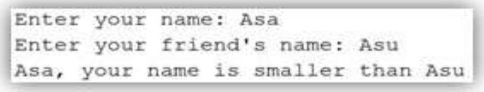
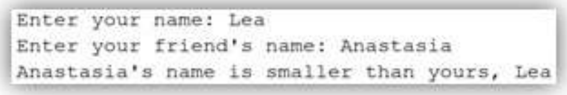
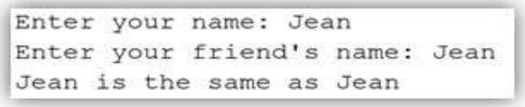
- Efficient programming

Using three separate if-statements is not the most efficient way to write the program determining the relationship between two strings. In one of the sections in this Unit we will teach you how to write it in a more efficient way.

Activity 18: Use the method compareTo() to determine the relationship between strings

a) Write a Java program that will ask the user to enter his name and his friend's name. The program should determine the relationship between the names (smaller, bigger or equal).

Examples of output		

 `Enter your name: Asa`
`Enter your friend's name: Asu`
`Asa, your name is smaller than Asu`
	 `Enter your name: Lea`
	`Enter your friend's name: Anastasia`
	`Anastasia's name is smaller than yours, Lea`
		 `Enter your name: Jean`
		`Enter your friend's name: Jean`
		`Jean is the same as Jean`

b) Write a program that prompts the user to enter 3 names. If the first name is the same as the third name but less than the second name display the following message: "Coincidence" otherwise display : "Not lucky":

4.16 Ignoring case in String comparisons

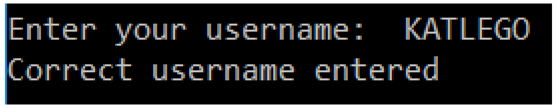
- Case-sensitive
- Uppercase
- Lowercase

Look at the following scenario: We want to write a program where a user should enter their username to log onto a system, and then we want to compare the username to the one they registered with. The username is not case-sensitive. However, we know that users may make mistakes, and they may enter the name with some letters as uppercase and others not. For example, the username is stored as Katlego, but the user enters katlego, or KATLEGO. It is not possible to write if .. else statements to make provision for all the different ways a user may enter the username, therefore we want to ignore the case of the letters entered by the user. There is more than one way to achieve this.

4.16.1 Using comparison methods from the String class

The following code demonstrates how to achieve this:

```
import java.util.Scanner;
public class CompareUsername {
    public static void main(String[] args){
        Scanner keyboard = new Scanner(System.in);
        String sUsernameStored = "katlego";
        System.out.print("Enter your username:  ");
        String sUsernameEntered = keyboard.nextLine();

        if (sUsernameEntered.equalsIgnoreCase(sUsernameStored))
            System.out.println("Correct username entered");
        else
            System.out.println("WRONG username entered (" + sUsernameEntered + ")");

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



```
Enter your username:  KATLEGO
Correct username entered
```

The String class also has a method **compareToIgnoreCase()** that can be used to compare strings without taking the case of the letters into consideration. Look at the following program segment:

```
String str1 = "HELLO";
String str2 = "hello";

System.out.println(str1.compareToIgnoreCase(str2));
```

The value 0 will be printed.

If the two strings are not the same (with case disregarded), a value other than a zero will be returned by the compareToIgnoreCase() method.

4.16.2 *Converting all letters to upper or lower case*

The String class has methods you can use to convert all the letters in a string value to be either all uppercase, or all lowercase. Look at the following alternative code for the username program:

```
import java.util.Scanner;
public class CompareUsernameConvert {
     public static void main(String[] args){
          Scanner keyboard = new Scanner(System.in);
          String sUsernameStored = "katlego";
          System.out.print("Enter your username:  ");
          String sUsernameEntered = keyboard.nextLine();

          sUsernameEntered = sUsernameEntered.toLowerCase();

          if (sUsernameEntered.equals(sUsernameStored))
               System.out.println("Correct username entered");
          else
               System.out.println("WRONG username entered (" + sUsernameEntered + ")");
     } //end main method
} //end class
```

The correct username is stored in lowercase letters.

The input from the user is converted to contain lowercase letters only.

The method *equals()* can be used, instead of the method *equalsIgnoreCase()*.

The String class also has a method called toUpperCase() that will convert all characters in a string value to be uppercase letters.

Activity 19: Ignore case when reading a string from the keyboard

a) Considering the following code, what will the statements in the table return (zero or a non-zero value)?

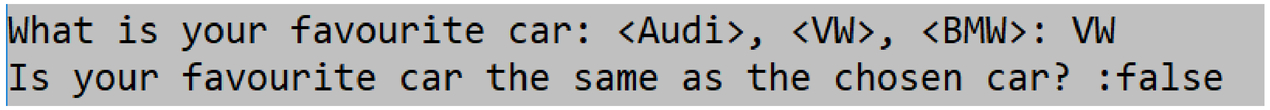
```
String sWord1 = "fastANDfurious";
String sWord2 = "fastandfurious";
```

Code	Zero or non-zero value
`sWord1.compareTo(sWord2)`	
`sWord1.compareToIgnoreCase(sWord2);`	

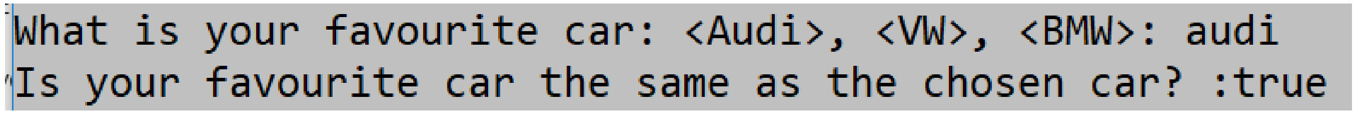
b) Write two different Java programs that will ask the user to pick their favourite European city between Paris and Berlin, and then display the choice of the user. If the user enters a different city, the program should display an error message.
   - In the first program use equalsIgnoreCase method
   - In the second program use compareToIgnoreCase method

c) The answer to a math problem is 2xy. Write a Java program that will ask the user to provide the answer of the problem. The program should display: "Correct answer" when the user enters 2xy regardless of the letters' cases. If another answer is entered display: "Wrong answer".

d) Write a Java program that will ask the user to enter the name of their favourite car and compare it with a car name that is stored in the variable *sAnswer1*. Store the result of the equality comparison in a Boolean variable bFlag1 and display the value of the Boolean. Use the equalsIgnoreCase method so the user is able to enter the car name in any format. You can decide which brand is your favourite, and store that value in the variable *sAnswer1.* For the example output, the value "Audi" was stored in sAnswer1.



```
What is your favourite car: <Audi>, <VW>, <BMW>: VW
Is your favourite car the same as the chosen car? :false
```



```
What is your favourite car: <Audi>, <VW>, <BMW>: audi
Is your favourite car the same as the chosen car? :true
```

e) What will be the output of following code? first predict the output, then write the Java program and compare your answer with the Java output.

Java code	Output

`public class Activity19 {`
`    public static void main (String args[])`
`    {`
`        String sPhrase1 = ("The year is Almost Over");`
`        String sPhrase2 = sPhrase1.toUpperCase();`
`        System.out.println(sPhrase2);`
`    }`
`  }`	

f) The variable sText contains the word “Atlantic”. Write Java code that will convert all the characters is sText to lower case.

4.17 Ignoring case in character comparisons

The following program will ignore the case of letters when characters are compared.

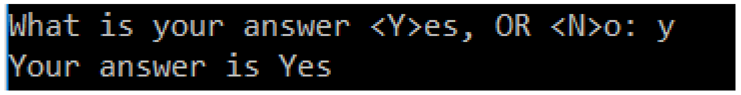
Read a character and use the method toUpperCase() from the Character class.

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

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

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

Read one character from the keyboard.

Use the method *toUppercase()* from the class Character to convert the letter entered by the user to an uppercase letter when the comparison is done.

Even when the user enters a lowercase letter y, it will be accepted as if the user entered Y.



```
What is your answer <Y>es, OR <N>o: y
Your answer is Yes
```

You can also use the character.toUpperCase() method when the value is read from the keyboard. The content of the variable will then be converted to a value that contains only uppercase letters.

```
          char cAnswer = Character.toUpperCase(keyboard.next().charAt(0));
          if (cAnswer == 'Y')
               System.out.println("Your answer is Yes");
          else
               System.out.println("Your answer is No");
```

Note

This code is not the best way to determine whether a user entered a 'y' or an 'n'. If the user accidentally enters the letter 't' (just next to the 'y' on the keyboard), this program will accept that the user entered an 'n'. Later in this module you will learn to write code that will repeatedly ask the user to enter an option until either a 'y' or an 'n' is entered.

Important information

- The Character class can be found in the java.lang package. You do not need to import the java.lang package to use any of the classes in it.
- The Character class a method toUpperCase() that will convert the character it receives as argument to consist of only upper-case letters.
- The Character class also has a method toLowerCase() that will convert the character it receives as argument to consist of only lower-case letters.

Activity 20: Ignore case when reading characters from the keyboard

a) Consider the following lines of a Java program (assignment statements):

```
String sMytext = "jOsEpH";
char cFirstChar = sMytext.charAt(0);
char cSecChar = sMytext.charAt(1);
```

Predict what the output for each of the following Java statements will be. Then write a Java program containing these assignment and output statements and determine whether your prediction of the output was correct.

1) What will the output be?

Java code	Your prediction	Java output
`System.out.println(Character.toUpperCase(cFirstChar));`		
`System.out.println(Character.toLowerCase(cSecChar));`		
`System.out.println(sMytext);`		

2) What will the following lines return?

Java code	True or false (Your prediction)	Java output
`System.out.println(Character.toUpperCase(cFirstChar)==74);`		
`System.out.println(Character.toLowerCase(cSecChar)>=111);`		
`System.out.println(Character.toLowerCase(cFirstChar)>106);`		

b) The user should enter the status of their test result: pass or fail. Write a program that will check if the first letter or the status entered is P or F. The program should display "You did well" if the status is P and "Work harder" if the status is F. if the user enters lower case letters your program should be able to convert them to upper case letters and accept p and f as P and F.

Note

In all the examples and activities where a decision had to be made, we had two possibilities, and therefore one logical expression. For example:

- The username is correct, or it is not - (two possibilities) – one logical expression `if (sName1.equals(sName2) == true)`
- The average is 75 or higher, or it is lower than 75 - (two possibilities) – one logical expression – `if (rAvg >= 75)`

In the next section of this unit, we are going to see how to use if .. else statements when decisions have to be made, and there are three possibilities. For example:

1. A user has to enter 'Y' or 'N' to choose an option, but the user can also enter an illegal character. So, there are three possibilities: 'Y', 'N' or illegal character.
2. A customer will receive 15% discount if more than R500 was spent, 10% discount if the amount was more than R300, otherwise 5% discount will be provided.

For these scenarios (three possibilities) you need to have two logical expressions, and you will use more than one if..else statement.

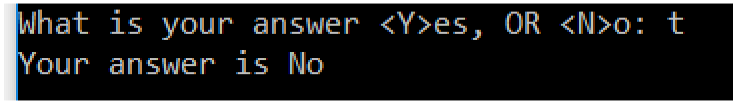
4.18 Nested if..else statements

In a previous example, we looked at code where a user had to indicate a choice by entering a character 'Y or 'N'. The code looked like this:

```
System.out.print("What is your answer <Y>es, OR <N>o: ");
char cAnswer = keyboard.next().charAt(0);
```

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

But look at the following output:



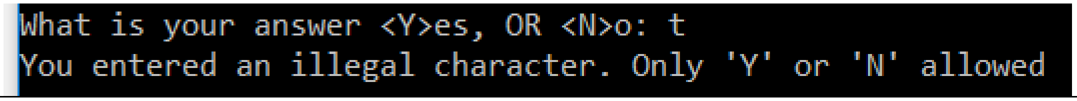
```
What is your answer <Y>es, OR <N>o: t
Your answer is No
```

The user wanted to enter a 'y', but typed the letter 't' instead, and now the program assumes an answer of No. This is not effective – a programmer should anticipate that a user may make mistakes when entering data and write code that will warn a user if something illegal has been entered. This is an example where there are actually three possibilities - 'Y', 'N' or illegal character. The following code provides for such a scenario:

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

          if (Character.toUpperCase(cAnswer) == 'Y')
               System.out.println("Your answer is Yes");
          else
               if (Character.toUpperCase(cAnswer) == 'N')
                    System.out.println("Your answer is No");
               else
                    System.out.println("You entered an illegal character. Only 'Y' or 'N' allowed");

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

Nested if .. else statement.

Three options – two logical expressions.

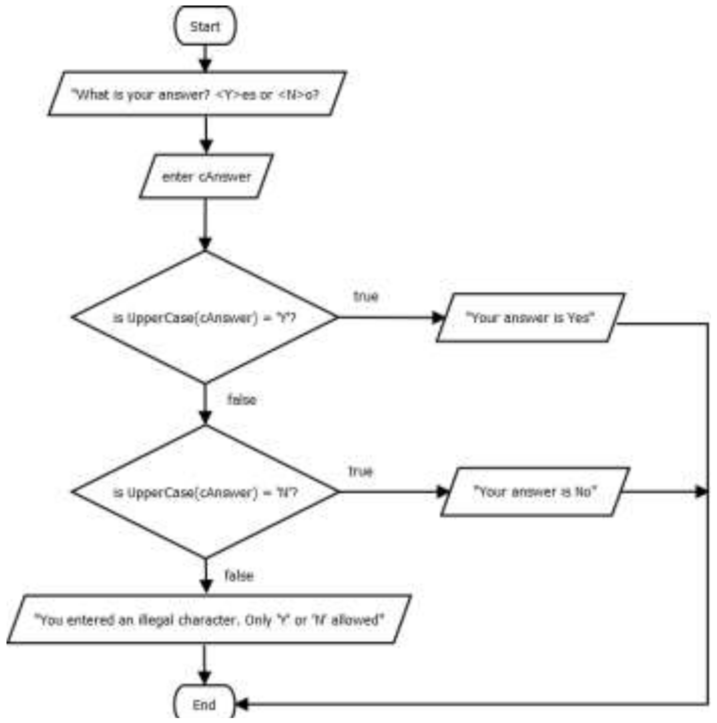


```
What is your answer <Y>es, OR <N>o: t
You entered an illegal character. Only 'Y' or 'N' allowed
```

The else-block contains a complete if..else statement. When the user entered the letter 't', the relational expression returns a value *false.* Therefore, the else-block will be executed. In the else-block, the nested if-statement tests whether the character was an 'N'. This logical expression will also return a *false* value. The else-block will be executed, and the message "You entered an illegal character. Only 'Y' or 'N' allowed" will be displayed.

This technique is called a *nested if .. else* statement. A complete if .. else statement is used inside either the if-block or the else-block.

The solution for this scenario can also be presented in a flow chart.


Start
"What is your answer? <Y>es or <N>o?
enter cAnswer
is UpperCase(cAnswer) = 'Y'?
true
"Your answer is Yes"
false
is UpperCase(cAnswer) = 'N'?
true
"Your answer is No"
false
"You entered an illegal character. Only 'Y' or 'N' allowed"
End

Note

In the previous scenario, there were three possibilities, therefore we needed two logical expressions:

```
if (Character.toUpperCase(cAnswer) == 'Y')
     System.out.println("Your answer is Yes");
else
     if (Character.toUpperCase(cAnswer) == 'N')
          System.out.println("Your answer is No");
     else
          System.out.println("You entered an illegal character. Only 'Y' or 'N' allowed");
```

First logical expression. If this outcome is false, there are two possibilities left: Either the user entered 'N', or an illegal character was entered.

Second logical expression. There are only two possibilities.

Using a trace table to find errors in if .. else statements

As you saw before, you can use a trace table to 'step through' each line of code that will be executed in a program. It is useful to create a trace table when you have to work through code where if..else statements are used. When if..else statements are used in a program, you need one more column in the trace table – to document the results of any of the logical expressions. Look at the following pseudo code and trace tables.

Write an algorithm to enter a value (rNumber) that is greater or equal to 1 and then perform the following actions depending on the value of the variable called number. Display the new value of number. You have to check for values that are less than 1 which will then result in displaying a suitable message to indicate that an incorrect value was entered.

Value of number	Action to take
Less than 1	Display an error message
1 – 20	Increase the value of rNumber by 5%
21 and above	Decrease the value of rNumber by 8%

```
IncOrDec
begin //start of the class
    main method
    begin // main method
        // declare variables
        double rNumber
        // Enter the value of the variable number greater or equal to 1
        display "Enter a value that is greater than 0 "
        enter rNumber
        // Test the value of number and perform action
        if rNumber < 1
            display "Incorrect value was entered."
        else  // the value of number is >=1
            if rNumber <= 20
                rNumber = rNumber + rNumber * 0.05 // increase by 5%
            else
                rNumber = rNumber – rNumber * 0.08 // decrease by 8%
            endif
        endif
        // Show the result on the screen
        display "The new value is " + rNumber
    end //main method
end //class
```

We will now test the algorithm using a trace table for the 3 possible input values

Value of rNumber = -5

Additional column

Instruction	rNumber	Outcome of if	Output
display			Enter a value that is greater than 0
enter	-5		
if		true	
display			Incorrect value was entered
display			The new value is -5

Value of rNumber = 15

Instruction	rNumber	Outcome of if	Output
display			Enter a value that is greater than 0
enter	15		
if		false	
if		true	
calculate	15.75		
display			The new value is 15.75

Value of rNumber = 30

Instruction	rNumber	Outcome of if	Output
display			Enter a value that is greater than 0
enter	30		
if		false	
if		false	
calculate	27.6		
display			The new value is 27.6

Activity 21: Predict the output of nested if..else statements

1. To get an award during the school award ceremony, students must get 75 percent or more and pass at least 4 subjects. The following pseudo code tests if a student qualifies for an award. The user will provide their percentage and the number of subjects passed.

```
Award // The class name
begin //start of the class
     main method
     begin //main method
          //declare variables
          int iPercent
          int iPassSubject
          //obtain inputs
          display "Provide the number of subjects passed”
          enter iPassSubject
          display "Provide the percentage"
```

```
            enter iPercent
            // decision
            if (iPercent>=75) then
            begin
                 if (iPassSubject>=4) then
                    display "You will get an award"
                 else
                          display "Sorry, you do not qualify"
            end
            else
                 display "Sorry, you do not qualify"
       end // end main method
  end // end of the class
```

What will be the output for the following cases?

Percentage	**Number of subjects passed**	**output**
75	3	
85	5	
64	5	

2. What will be displayed when the following code is executed for the respective values of A?:

```
a.     A = 24
b.     A = 0
c.     A = 64

if A <= 15 then
      begin
            display "**"
      end
```

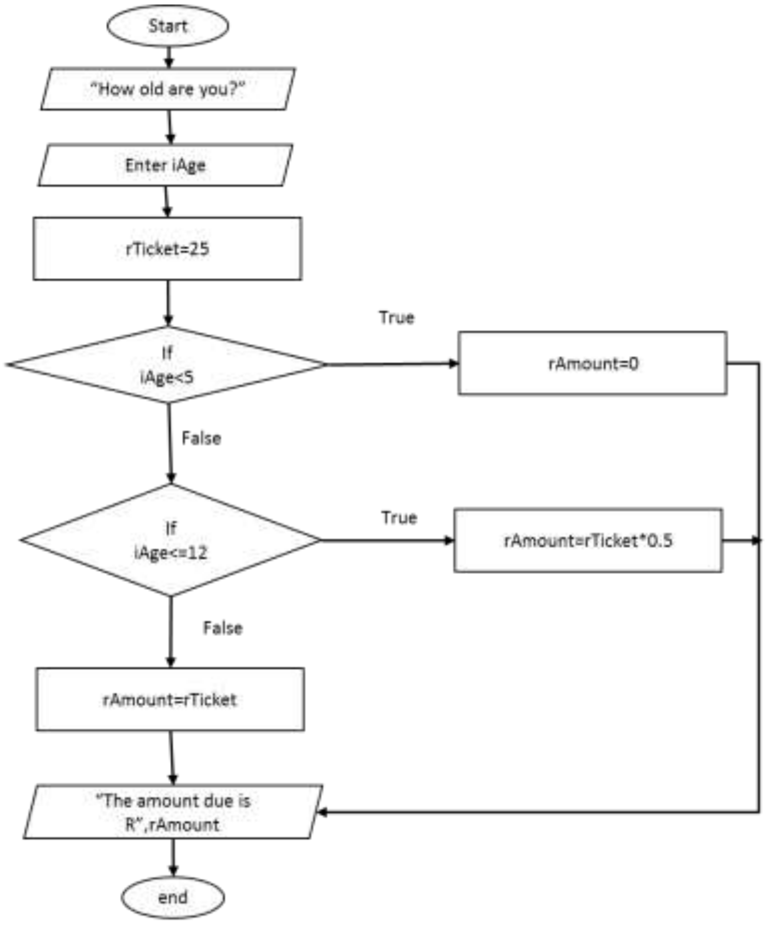
```
else
      begin
            if A < 50 then
                  begin
                        display "$$"
                  end
            else
                  begin
                        display "##"
                  end
      end

      display "The End"
```

3. What will be displayed on the screen after the following steps have been processed?

```
PredictOut // The class name
begin //start of the class
      main method
      begin //main method
            //declare variables
            double K,X
            int G
            K = 4
            G = 3
            X = K + G mod 2 * 15 – G ^ 3 + (K * G)
            // decision
            if (X != 0) then

                  begin
                        if (X < 0) then
                              begin
                              K = K \ 3
                              end
                        else
                              begin
                              K = K * 0.75
                              end
                  end
            else
               begin
                  K = K * 2
               end
            display "The value of X is ", X            ~     display on a new line
            display "The value of K is ", K            ~     display on a new line

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

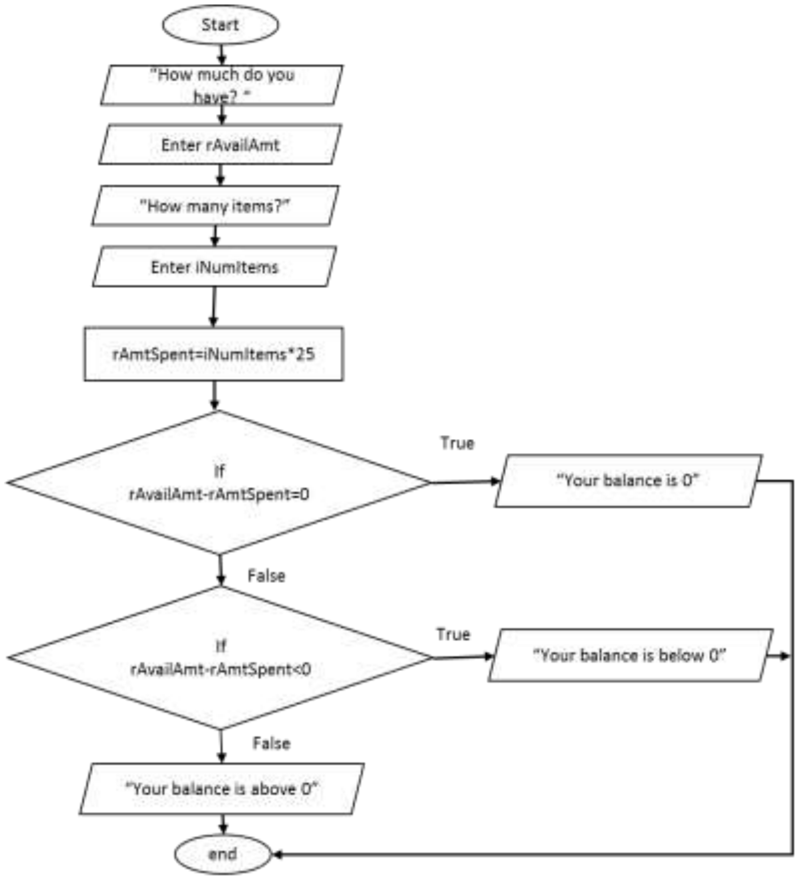
4. Considering the following flowchart, what will the output be in the following cases:

Age	Output
45	
4	
12	

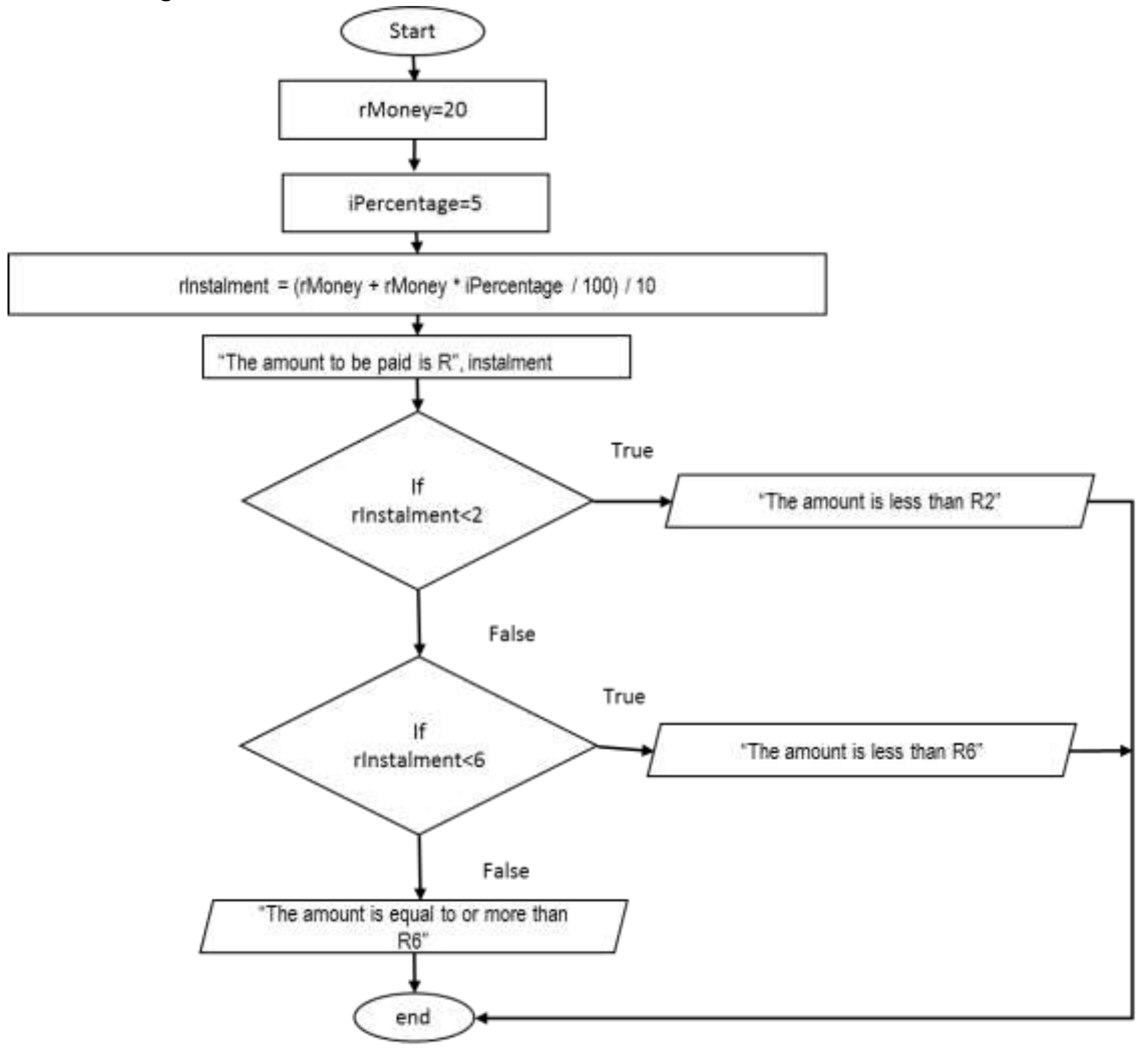


5. The following flowchart tests if the user’s bank account balance is below zero, zero or above zero after buying some items. The user enters how much they have in their account and how many items they are buying. Each item costs R25. We may assume the user can spend more than the available amount they have in their account. Complete the following by predicting the output based on the given data (available amount and number of items bought)

Available amount	Number of items	Output
R45	3	
R85	4	
R450	7	
R300	15	



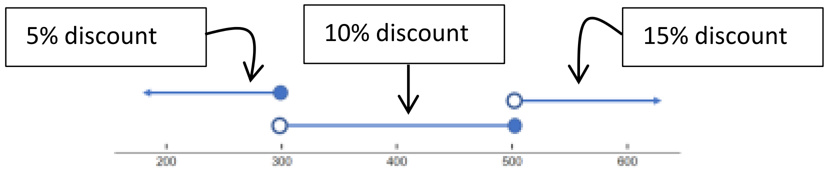
6. What will be the output of the following flowchart?


Start
rMoney=20
iPercentage=5
rInstalment = (rMoney + rMoney * iPercentage / 100) / 10
"The amount to be paid is R", instalment
If rInstalment<2
True
"The amount is less than R2"
False
If rInstalment<6
True
"The amount is less than R6"
False
"The amount is equal to or more than R6"
end

Look at the following scenario as a second example of the nested if..else statement:

A customer will receive discount for purchases based on the amount of money that was spent. To describe the rules to determine the discount, we can use different techniques:

1. Describe in words: A client will receive 15% discount if more than R500 was spent, 10% discount if the amount was more than R300, otherwise 5% discount will be provided.
2. Draw a line graph



3. Summarise the intervals in a table:

Amount spent	Percentage discount
R300 or less	5%
Between R300 and R500	10%
More than R500	15%

OR

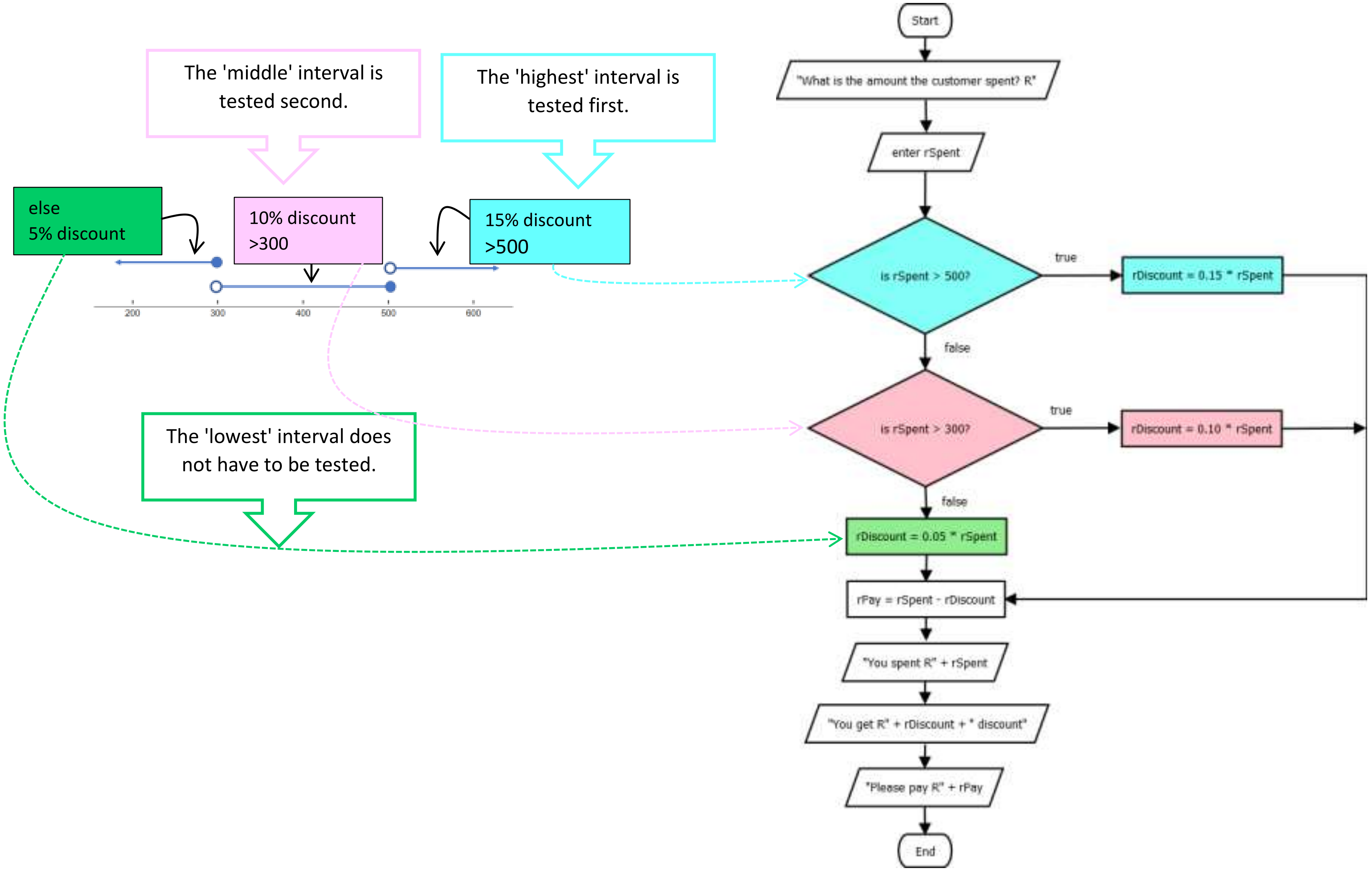
Amount spent	Percentage discount
More than R500	15%
Between R300 and R500	10%
R300 or less	5%

The following pseudo code is a solution where a nested if..else statement is implemented to correctly calculate the discount:

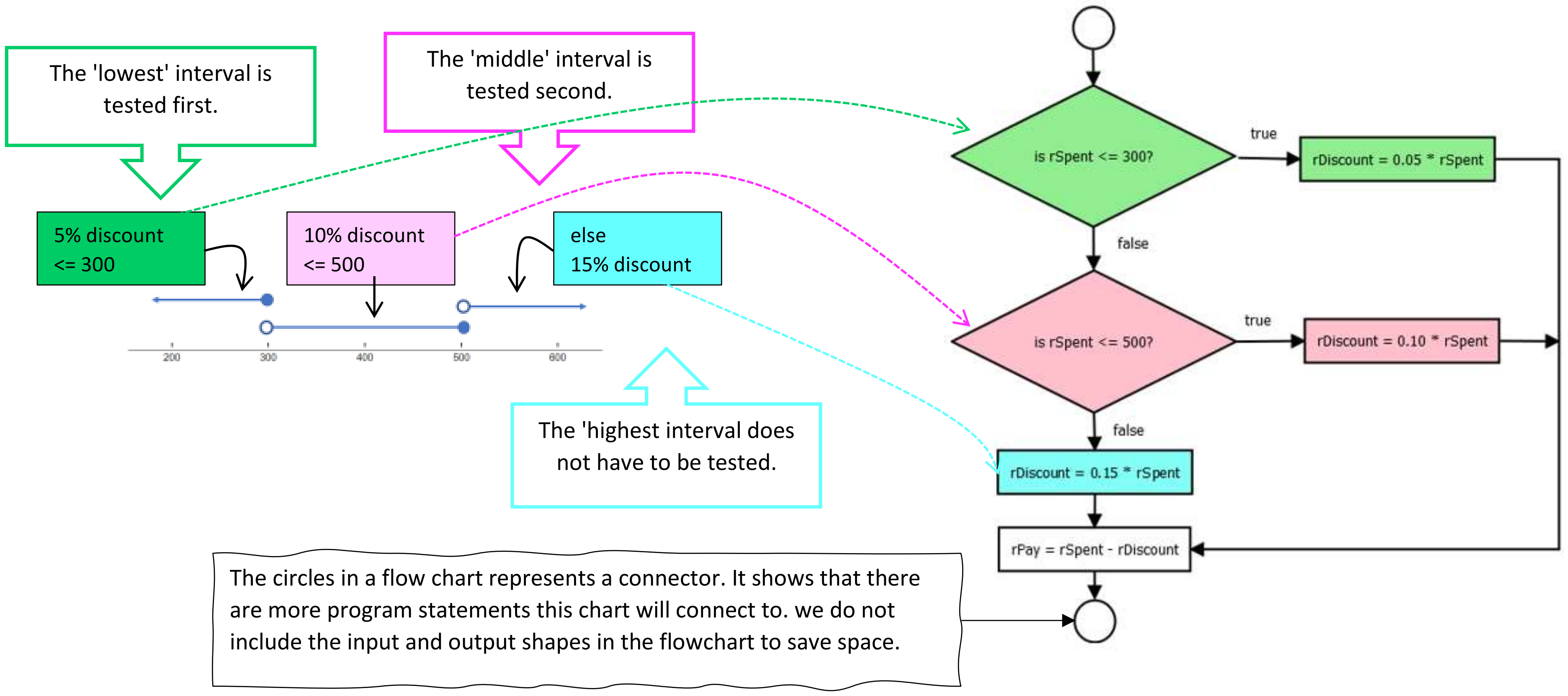
```
CalcDiscount
begin //start of the class
    main method
    begin //main method

        double rSpent, rDiscount, rPay
```

```
          display "What is the amount the customer spent? R"
          enter rSpent
          if rSpent > 500 then
               rDiscount = 0.15 * rSpent
          else
               if rSpent > 300 then
                    rDiscount = 0.10 * rSpent
               else
                    rDiscount = 0.05 * rSpent
               endif
          endif
          rPay = rSpent – rDiscount
          display "You spent R" + rSpent
          display "You get R" + rDiscount + " discount."
          display "Please pay R" + rPay
     end //main method
end //class
```

A flowchart for this solution can look as follows:


The 'middle' interval is tested second.
The 'highest' interval is tested first.
else
5% discount
10% discount
>300
15% discount
>500
200
300
400
500
600
The 'lowest' interval does not have to be tested.
Start
"What is the amount the customer spent? R"
enter rSpent
is rSpent > 500?
true
rDiscount = 0.15 * rSpent
false
is rSpent > 300?
true
rDiscount = 0.10 * rSpent
false
rDiscount = 0.05 * rSpent
rPay = rSpent - rDiscount
"You spent R" + rSpent
"You get R" + rDiscount + " discount"
"Please pay R" + rPay
End

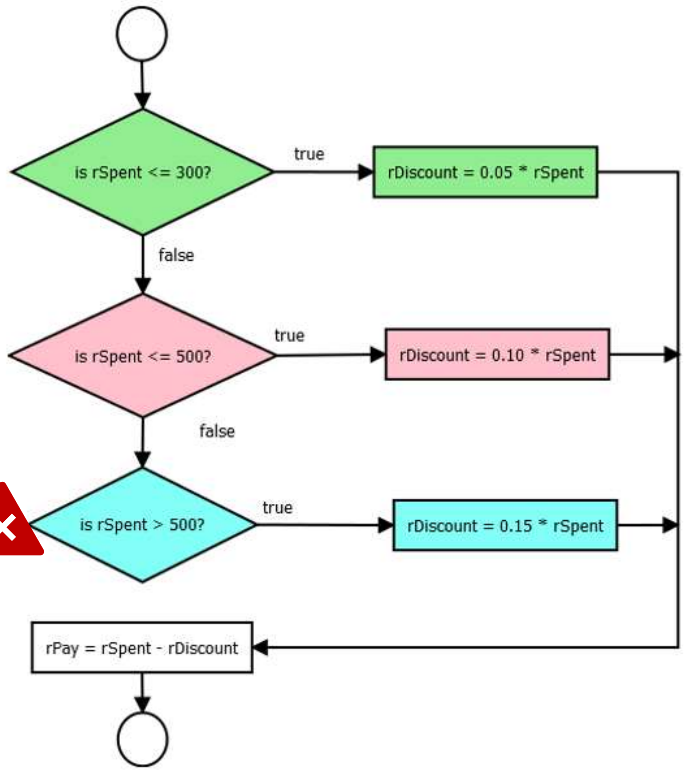
- The following solution is also correct:


The 'lowest' interval is tested first.
The 'middle' interval is tested second.
5% discount <= 300
10% discount <= 500
else 15% discount
200
300
400
500
600
The 'highest interval does not have to be tested.
The circles in a flow chart represents a connector. It shows that there are more program statements this chart will connect to. we do not include the input and output shapes in the flowchart to save space.
is rSpent <= 300?
true
rDiscount = 0.05 * rSpent
false
is rSpent <= 500?
true
rDiscount = 0.10 * rSpent
false
rDiscount = 0.15 * rSpent
rPay = rSpent - rDiscount

Important

- Notice how it helps you to look at the line graph's circles and dots to determine what logical operator to use in a relational expression. A dot indicates that a value is included (equals), a circle indicates greater than (>) or less than (<).

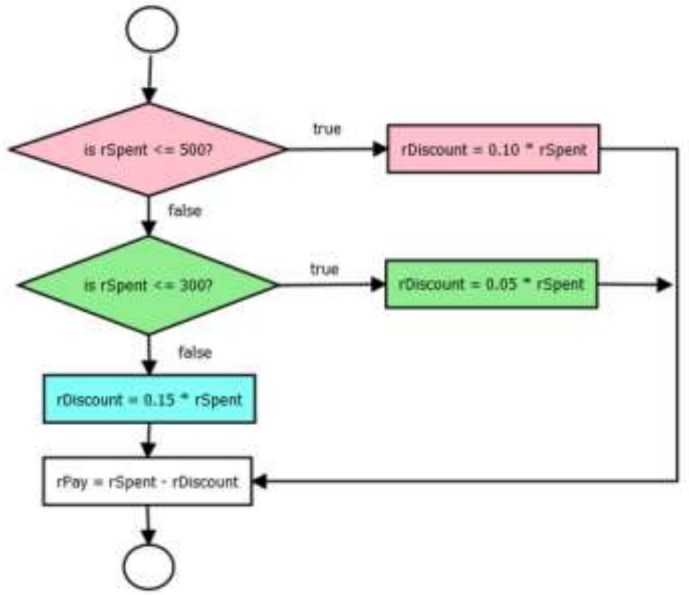
- There are three options: either 15% discount, 10% discount, or 5% discount. However, there are only two relational expressions. If the the customer does not qualify for 15% discount, and also not for 10% discount, then the only option left is to get 5% discount. It is not necessary to test for it again. Therefore, the following flowchart does not represent an efficient solution.



is rSpent <= 300?
true
rDiscount = 0.05 * rSpent
false
is rSpent <= 500?
true
rDiscount = 0.10 * rSpent
false
is rSpent > 500?
true
rDiscount = 0.15 * rSpent
rPay = rSpent - rDiscount

It is not necessary to test for rSpent <= 300. If rSpent is not > 500, and it is not > 300, then the only option left is that rSpent <=300. There is no need to test for it.

The order in which the intervals are tested is important. The following flowchart will not provide correct results:


is rSpent <= 500?
true
rDiscount = 0.10 * rSpent
false
is rSpent <= 300?
true
rDiscount = 0.05 * rSpent
false
rDiscount = 0.15 * rSpent
rPay = rSpent - rDiscount

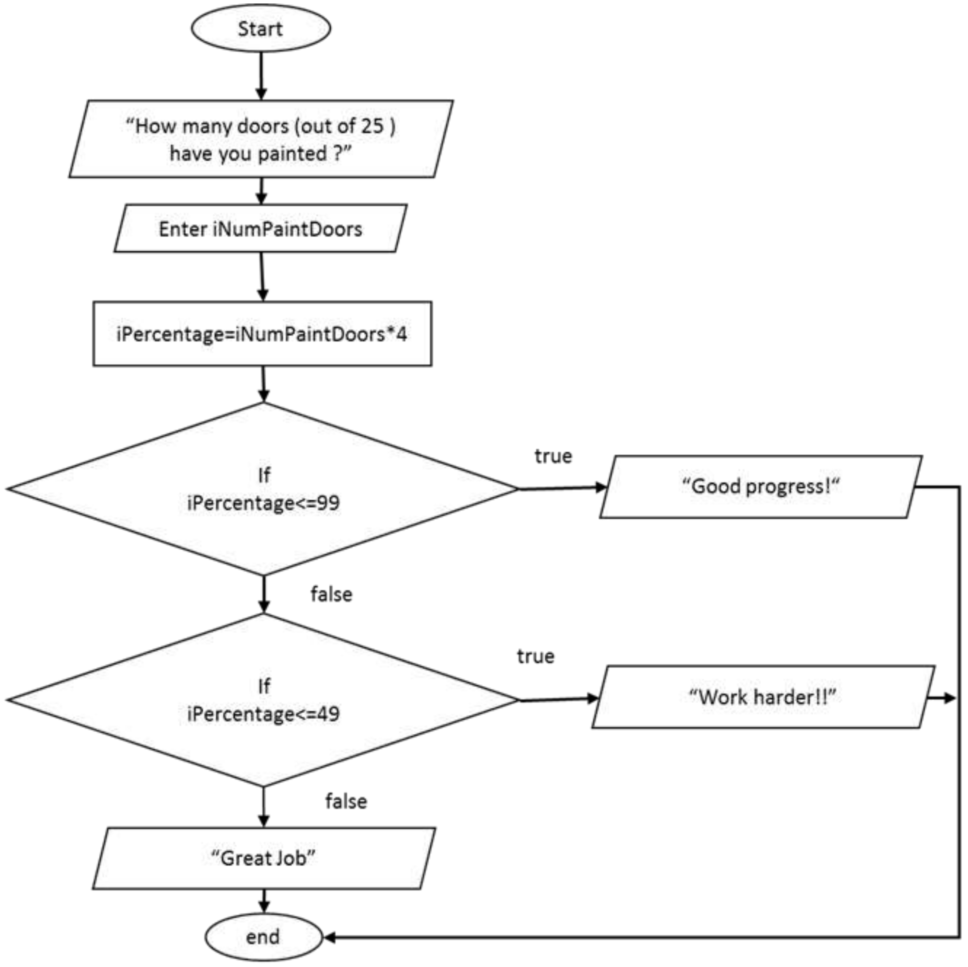
Activity 22: Test validity of algorithms with nested if .. else solutions

a) Determine the outcome for different values for the last flowchart provided. Use the following values as input: rSpent = 200, for the second trace table, use the value rSpent = 450, and for the last trace table use the value rSpent = 600.

b) The following flowchart attempt to monitor the progress of a handy man who is painting 25 doors of a building. The number of painted doors will be entered and then the algorithm must show his progress by displaying one of the following messages:

Percentage of completed doors	**Message**
0 - 49	work harder!
50 - 99	Good progress!
100	Great Job!!

- Use a trace table to test the algorithm. Test with 13 painted doors and 6 painted doors, demonstrate that the output of the flow chart will be incorrect
- Re-draw a correct flowchart that will produce correct outputs



c) Consider the following pseudocode

```
display "How many tickets do you want to buy? "
enter  iNumber
display "In which block must the seats be? – A, B or C"
enter cBlock
if cBlock = 'A' then
    begin
      ticketPrice = 150 * iNumber
    end
else
    begin
      if cBlock = 'B' then
         begin
            ticketPrice = 115 * iNumber
         end
      else
            if cBlock = 'C' then
                begin
                  ticketPrice = 55 * iNumber
                end
     end
if iNumber > 10 then
begin
    ticketPrice = ticketPrice * 0.95
end
rFinalAmount = ticketPrice * 1.15
display "The amount due is R" + rFinalAmount
```

- test the algorithm with the following data

code	Number of tickets
B	9
C	15

- What will happen if we do not include if cBlock = 'C'?

d) The following pseudo code tests the relationship status of the user. And displays the following message: “in relationship” when the answer is yes, “single” when the answer is no or “complicated” when the answer is Maybe. Test the algorithm when the answer is Yes and when the answer is No. Will the following pseudocode produce correct outputs? if no, provide a pseudocode that will produce correct outputs.

```
display "Are you in a relationship? <Y>es, <N>o, <M>aybe"
enter cAnswer
if cAnswer = 'Y' then
    begin
        if cAnswer = 'N' then
                begin
                  display "In a relationship"
                end
        else
                begin
                   Display "Single"
                end
   end
else
  begin
        display "It is complicated"
  end
```

Activity 23: Develop solutions using nested if..else statements

a) Write an algorithm to enter a value (rNumber) that is greater or equal to 1 and then perform the following actions depending on the value entered. Display the new value of rNumber. You have to check for values less than 1 which will then result in displaying a suitable message to indicate that an incorrect value was entered.

Value of rNumber	Action to take
1 – 20	Increase the value of rNumber by 5%
21 and above	Decrease the value of rNumber by 8%

b) The students in the residence have a choice of ordering one of 3 different meals at the following prices:

Meal type	Price of meal in Rand
A	15.00
B	18.50
C	24.20

A program is needed to enter the name of the student as well as the type of meal preferred and then display a message to indicate how much the student owes for the meal e.g. Tamsin must pay R18.50.

Study the pseudo code below.

i. Rewrite the pseudo code so that it uses a nested if-statement.
ii. Draw a flowchart for the solution using a nested if-statement.
iii. Write a Java program based on the algorithm. The program should make provision that a user may enter lowercase letters (a, b or c), and still provide correct output.

```
DetermineMealPrice
begin //start of the class
     main method
     begin //main method

          //declare variables
          String sName
          char cMealType
          double rPrice
          //get the input
          display "Name of the student:"
          enter sName
          display "Meal type, A, B or C:"
          enter cMealType

          // determine the price
          if cMealType =  'A'
               rPrice = 15.00
          endif
          if cMealType = 'B' then
               rPrice = 18.50
          endif
          if cMealType = 'C then
               rPrice = 24.20
          endif

           // show the price on the screen
          display sName + "  must pay  R" + rPrice
     end //main method
end //class
```

This is not a good solution. If the user enters a letter 'A', two more tests will be executed anyway, since it is three separate if-statements.

c) Employees working at the Good Fortune company have the option of joining a savings plan. There are 3 different plans available named Premium, Gold or Silver. The name of the employee, salary of the employee, the amount the employee has already saved, and the name of the chosen plan should be entered. An employee who chooses the Premium plan pays 8% of his salary towards the savings plan; the employee who chooses the Gold plan, pays 5% of his salary towards the savings plan while the employee who chooses Silver, pays R150 towards the savings plan. Determine the amount that must be paid to the savings plan and add it to the amount already saved. Display the name of the employee and the total amount saved.

Activity 24: Develop solutions using nested if..else statements

a) Study the following pseudocode and answer the following questions

```
Balance // The class name
begin //start of the class
      main method
      begin //main method
            //declare variables
                  double rBalance
                  double rTotWithdraw
                  double rWithdrawAmt
            // input
            display "Provide the current balance: "
            enter rBalance
            display "Enter the recent (previous) withdrawal: "
            enter rTotWithdraw
            display "Enter the amount to withdraw: "
            enter rWithdrawAmt
            // decision
            if rWithdrawAmt <= rBalance Then
              begin
                  if (rTotWithdraw + rWithdrawAmt) < 500 Then
                    begin
                       rBalance = rBalance - rWithdrawAmt
                       rTotWithdraw = rTotWithdraw + rWithdrawAmt
                    end
                  else
```

```
                              begin
                                 rWithdrawAmt = 0
                              end
                        end
                      else
                       begin
                             rWithdrawAmt = 0
                       end
                      display "The current amount withdrawn = " + rWithdrawAmt
                      display "The total recent withdrawals = " + rTotWithdraw
                      display "The new Balance = " + rBalance
                end //main method
          end //class
```

Using the trace table test the algorithm with the following values

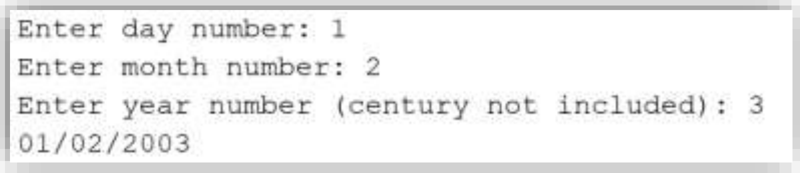
rBalance	rTotWithdraw	rWithdrawAmt
675.5	0.0	100.0
10.00	100.0	300.0
150.0	450.0	10.0

b) Everybody in the town will be attending the circus during the weekend. Do the complete planning and write an algorithm to determine the entrance fee for a particular person. Persons younger than 12 or persons 60 and older receive a 25% discount. The user must enter the normal entrance fee and the person's age in years as an integer value (the age cannot be a negative value – an error message should be displayed if it is). Calculate and display the entrance fee on the screen.

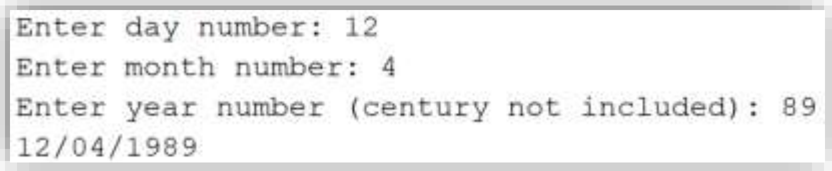
c) Enter a day number, month number and year (maximum 2 digits). If the year is not a value between 0 and 99, an error message should be displayed.

Create a string to display the date in the format dd/mm/yyyy. E.g. 06/01/1989.

If a year between 0 and 29 (both included) is used, the date of birth was in the 21st century, so you should add the digits '20' to the front of the year value. If the year value is between 30 and 99 (both included), the date of birth is in the 20th century, which means '19' should be added to the front of the year value. E.g. 20/06/45 must become 20/06/1945.The date 23/12/01 must become 23/12/2001.



```
Enter day number: 1
Enter month number: 2
Enter year number (century not included): 3
01/02/2003
```



```
Enter day number: 12
Enter month number: 4
Enter year number (century not included): 89
12/04/1989
```

Tip: You can use the following formatting pattern and code to ensure a single-digit day and month is displayed with a 0:

```
DecimalFormat formatter = new DecimalFormat ("00");
sDateConverted = "" + formatter.format(iDay) + "/" + formatter.format(iMonth) + "/" + iYear;
```

d) Write a program where the user can enter a year, e.g. 2003, and the program has to determine whether the year is a leap year or not. A leap year is determined as follows: If the year ends in 00 (e.g. 1900 or 2000), it is a leap year if it can be divided by 400 without a remainder. If it does not end in 00, it is a leap year if it can be divided by 4 without a remainder.

e) Create a program where the user will be asked to enter the number of SMS messages they sent in a month and the cost per SMS message. The user will then be required to indicate whether they are on a cell phone contract or not. The first 20 messages will be free of charge for a contract. The program must calculate the total cost of the SMS messages sent.

f) You need to design a program for a courier company to calculate the cost of delivering a parcel. The cost is calculated as follows:

- The basic cost is calculated as R 1.23 per kilogram + transport cost per kilometre.

  The transport cost per kilometre depends on the mode of transport:

  Road: R 0.15; Train: R 0.12; Air: R 0.36; Ship: R 0.25

- If the customer wishes to ensure the parcel, insurance cost is calculated as 11% of the basic cost.
- If the customer wishes to have the parcel delivered as a high priority parcel, priority cost is calculated as 15% of the basic cost.
- VAT must be calculated and added to the total cost (basic cost + insurance + priority). Round off the total cost to the nearest Rand.

The program only needs to report on the total cost. All other values should be displayed on the stage using monitors.

Use the following data to test your program:

Weight	**Kilometres**	**Mode of transport**	**Insurance?**	**High priority?**	**Total cost**
10	100	Road	Yes	No	R35
40.5	250	Train	Yes	Yes	R115
23.8	320	Air	No	Yes	R189
100	600	Ship	No	No	R311

g) The *Coffee Corner* restaurant needs a program to calculate and display a number of values:

Input to the program:

- The amount on the bill.
- An indication whether the customer is a pensioner or not.
- An indication whether the customer presented a club card or not.

The following calculations need to be done:

- Provide a 5% discount on the bill if the customer is a pensioner.
- Calculate the waiter's tip of 10%.
- Calculate the total due by the customer.
- Calculate the loyalty points if the customer presents a club card, according to the following rules:
    - If the total due (excluding the tip) exceeds R850 points, the customer earns 50 points per Rand.
    - If the total due (excluding the tip) is between R150 and R850 (both values included), the customer earns 20 points per Rand.
    - If the total due (excluding the tip) is less than R150, the customer earns 10 points per Rand

Display the waiter's tip, the discount (if any), the total amount due, as well as the number of loyalty points earned. Round the amount due to two digits.

h) The organisers of a singing competition use a grading system to see whether a contestant is through to the final round:

- Judges as well as the public can cast votes.

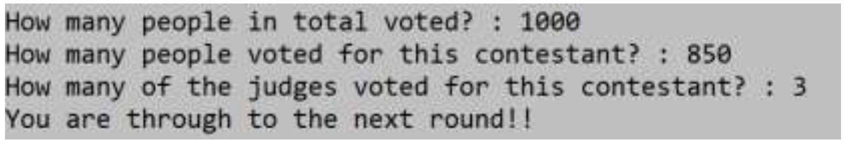
A score out of 100 must be calculated for each contestant using the public votes, the number of judges' votes and bonus points. The score is calculated as follows:

- The number of votes from the public is converted to a number out of 60.
- The number of votes from the judges (out of the 4) is converted to a number out of 30.
- They score 10 additional bonus points if they receive more than 75% of the public votes **or** more than 75% of the judges' votes.
- The three values must be added up to create the final score.

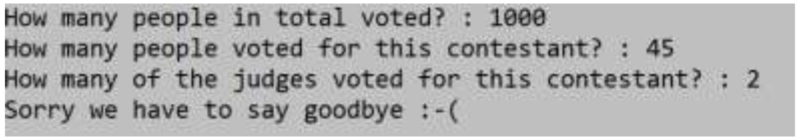
- To qualify to be in the final round, a contestant must have a score of at least 75 **and** have at least two judges' votes.
- Input to the program:
  Number of persons from the public who voted in total; Number of persons from the public who voted for this contestant; Number of judges who voted for this contestant

Create a flowchart for this program. The program must ask all the necessary values (input) from the user, then calculate and display the final score.

Use the flowchart and create a Java program.



```
How many people in total voted? : 1000
How many people voted for this contestant? : 850
How many of the judges voted for this contestant? : 3
You are through to the next round!!
```



```
How many people in total voted? : 1000
How many people voted for this contestant? : 45
How many of the judges voted for this contestant? : 2
Sorry we have to say goodbye :-(
```

Activity 25: Knowledge checklist

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

	✓ / ?
Between	
Boolean expression	
Boolean operator	

Case sensitive	
Comparison operator	
Compound condition	
Conditional	
Conditional Operator / Ternary operator	
Conditional statement	
Different	
Efficient programming	
Exclude	
Flowchart	
Include	
Instalment	
Interval notation	
Logical expression	
Logical operator	
Lowercase	
Number line representation	
Relational expression	
Relational operator > < >= <= == !=	

Sequential	
Set builder notation	
Similar	
Ternary	
Uppercase	

Activity 26: Skills checklist

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

	✓ / ?
Design an algorithm in pseudo code or using a flow chart to solve a basic mathematical problem, requiring selection control structures.	
Convert algorithms to Java and vice versa.	
Predict output and correct errors of provided algorithms using a trace table.	
Design programs to solve problems that require decisions and have up to three possible outcomes.	
Create relational as well as Boolean expressions.	
Apply rules for precedence and associativity in solving relational and Boolean expressions.	
Write logical expressions to determine whether a value falls within a certain range.	
Write code to compare String and char data.	
Write code to ignore case in character comparisons.	

4.19 Using switch statements

When you need to make decisions in a program, another structure is available in Java that can usually perform the same task as if .. else statements, but that some programmers find easier to write. This structure is called the *switch* statement. It is a type of control structure that is known as a *case structure*.

Most if .. else statements can be re-written as a switch statement. Let us look at the following scenario: We want to write a program where the user can enter the day of the week, and the program should return the name of the day. For example, day 1 is Sunday, day 4 is Wednesday. A Java program to do this can look as follows:

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

          System.out.print("Enter day number: ");
          iDayNumber = keyboard.nextInt();
          if (iDayNumber == 1) sDayName = "Sunday";
          else if (iDayNumber == 2) sDayName = "Monday";
          else if (iDayNumber == 3) sDayName = "Tuesday";
          else if (iDayNumber == 4) sDayName = "Wednesday";
          else if (iDayNumber == 5) sDayName = "Thursday";
          else if (iDayNumber == 6) sDayName = "Friday";
          else if (iDayNumber == 7) sDayName = "Saturday";
          System.out.println("Day number " + iDayNumber + " is " + sDayName);
     } //end main method
} //end class
```

Note how the nested if statements can be formatted to be displayed on fewer lines.

This program can be written using a switch statement.

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

        System.out.print("Enter day number: ");
        iDayNumber = keyboard.nextInt();
        switch (iDayNumber)
        { //beginning of switch statement
            case 1: sDayName = "Sunday"; break;
            case 2: sDayName = "Monday"; break;
            case 3: sDayName = "Tuesday"; break;
            case 4: sDayName = "Wednesday"; break;
            case 5: sDayName = "Thursday"; break;
            case 6: sDayName = "Friday"; break;
            case 7: sDayName = "Saturday"; break;
            default:  sDayName = "";
        } //end of switch statement
        System.out.println("Day number " + iDayNumber + " is " + sDayName);
    } //end main method
} //end class
```

Switch keyword

Switch expression – it is compulsory to use the parentheses.

Case labels

Note:

- The switch expression is inside parentheses.
- Each case label is followed by a colon :
- Each statement to be executed ends with a semi-colon ;

1. The switch keyword must be followed by a *switch expression* (the variable or expression that must be evaluated) in parentheses.
2. One or more *case labels* follows that represent the possible values of the switch expression.
3. When the switch expression matches the value specified by the case label, the statements following the case label will be executed.
4. Case labels can be coded in any sequence.

- Keyword
- Switch expression
- Case labels

5. A *break* statement must be included after the statements for each case label. Each break statement terminates the switch statement – in other words the control of the program will jump to the first statement following the switch statement.
6. Without break statements the statements in switch blocks *fall through.* That means *a*ll statements after the matching case label are executed in sequence, regardless of the expression of subsequent case labels, until a break statement or the end of the switch statement is encountered. (Oracle, 2019)
7. The default label is an optional label that identifies the statement to be executed if none of the case labels are executed.
8. The switch expression may be of type char, byte, short, int or String object. (Note – you cannot use a long data type in a switch expression.)
9. Strings in switch statements are case-sensitive.

- Optional

4.19.1 Converting if .. else statements to a switch statement

Deciding whether to use if .. else statements or a switch statement is based on readability and the expression that the statement is testing. An if .. else statement can test expressions based on ranges of values or conditions, whereas a switch statement tests expressions based only on a single char, byte, short, int or String object. (Oracle, 2019)

Here are more examples of if .. else statements and the equivalent switch statements.

Example 1:

Employees receive a bonus at the end of the year based on the number of days they were absent. The following two programs give equivalent results.

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

      int iDaysAbsent;
      double rBonus;

      System.out.print("Enter number of days absent: ");
      iDaysAbsent = keyboard.nextInt();
```

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

      int iDaysAbsent;
      double rBonus;

      System.out.print("Enter number of days absent: ");
      iDaysAbsent = keyboard.nextInt();
      if (iDaysAbsent == 0)
          rBonus = 500;
      else
          if (iDaysAbsent ==1)
              rBonus = 250;
          else
              if (iDaysAbsent == 2)
                  rBonus = 100;
              else
                  rBonus = 0 ;

      System.out.print("Your bonus is "+formatter.format(rBonus));
   } //end main method
} //end class
```

```
      switch (iDaysAbsent)
      {
         case 0: rBonus = 500; break;
         case 1: rBonus = 250; break;
         case 2: rBonus = 100; break;
         default: rBonus = 0;
      }
      System.out.print("Your bonus is " + formatter.format(rBonus));
   } //end main method
} //end class
```

Example 2:

This example demonstrates how a string value can be used as a switch expression.

A student should enter the name of a module. The program will then display the lab the student should go to.

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

           String sModuleName, sMessage;

           System.out.print("Enter module code (e.g. PPAF05D): ");
           sModuleName = keyboard.nextLine();
           sModuleName = sModuleName.toUpperCase();

           if (sModuleName.equals("PPAF05D"))
                 sMessage = sModuleName + " Go to 10-LG44";
           else if (sModuleName.equals("COHF05D"))
                 sMessage = sModuleName + " Go to 10-LG15";
           else if (sModuleName.equals("CFAF05D"))
```

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

     String sModuleName, sMessage;

     System.out.print("Enter module code (e.g. PPAF05D): ");
     sModuleName = keyboard.nextLine();
     sModuleName = sModuleName.toUpperCase();

     switch (sModuleName)
     {
      case "PPAF05D":
         sMessage = sModuleName + " Go to 10-LG44"; break;
      case "COHF05D":
```

All letters were converted to uppercase, therefore the case labels contain only capital letters.

```
                  sMessage = sModuleName + " Go to 10-140";

            else if (sModuleName.equals("COEF05X"))
                  sMessage = sModuleName + " Go to 10-170";
            else sMessage = "Module " + sModuleName + " is not valid";
            System.out.print(sMessage);
      } //end main method
} //end class
```
	```
	         sMessage = sModuleName + " Go to 10-LG15"; break;
	      case "CFAF05D":
	         sMessage = sModuleName + " Go to 10-140"; break;
	      case "COEF05X":
	         sMessage = sModuleName + " Go to 10-170"; break;
	      default:
	         sMessage = "Module " + sModuleName + " is not valid";
	      }
	      System.out.print(sMessage);
	   } //end main method
	} //end class
	```

Activity 27: Convert if .. else statements to switch statements

Convert the following if..else statements to switch statements

Java code using Nested if	Java code using Switch
```
a)
public class Preferences
{ public static void main (String [] args)
     { //start main
          Scanner keyboard = new Scanner(System.in);
          System.out.println("Provide your preference:  <Vegan>,
                              <Vegetarian>,<Prescatarian>");
          String sPreference = keyboard.nextLine();
          sPreference= sPreference.toLowerCase();
          if (sPreference.equals("vegan"))
               System.out.println("strictly vegetables!");
          else if(sPreference.equals("vegetarian"))
               System.out.println("Vegetables and eggs and milk!");
          else if (sPreference.equals("prescatarian"))
               System.out.println("Vegetables and sea food");
          else
                System.out.println("Invalid choice, you can eat whatever you
like");
     }
```

```
}
```

```
b)
import java.util.Scanner;

public class Coffee
{ public static void main (String [] args)

     { //start main
          Scanner keyboard = new Scanner(System.in);
          System.out.println("Coffee sizes: 1=small 2=medium 3=large 4=extra-large");
          System.out.println("Please enter your selection <1>,<2>,<3> or <4>");

          int iNumber = keyboard.nextInt();

          double rPrice=0;
          String sChoice;
          if (iNumber==1)
               {sChoice="small";
               rPrice=10;}

          else if (iNumber==2)
               {sChoice="medium";
               rPrice=12.5;}
          else if (iNumber==3)
               {sChoice="Large";
               rPrice=18;}
          else if (iNumber==4)
               {sChoice="extra-Large";
               rPrice=20;}
          else
               {sChoice=" An invalid selection. Please select 1, 2, 3 or 4";}

          System.out.println("You chose:"+sChoice+". You must pay R" + rPrice);
     }
}
```

4.19.2 Pseudo code and flowcharts for switch statements

A switch statement can be represented as pseudo code or using a flow chart. The pseudo code for the switch statement of Example 1 can look as follows:

```
switch (iDaysAbsent)
     {
          case 0: rBonus = 500; break;
          case 1: rBonus = 250; break;
          case 2: rBonus = 100; break;
          default: rBonus = 0;
     } //switch
```

All programming languages have case structures which let you code actions for different cases. In Java it is called a switch statement, in other languages key words such as case, select or inspect are used. (Wikipedia, 2020)

Because you are writing pseudo code for a Java program, it is acceptable to use the same syntax for your pseudo code. It is also acceptable to leave out symbols relating to Java syntax.

```
case based on iDaysAbsent
     begin
          case 0
               rBonus = 500
               break
          case 1
               rBonus = 250
               break
          case 2
               rBonus = 100
               break
          default
                rBonus = 0
     end case
```

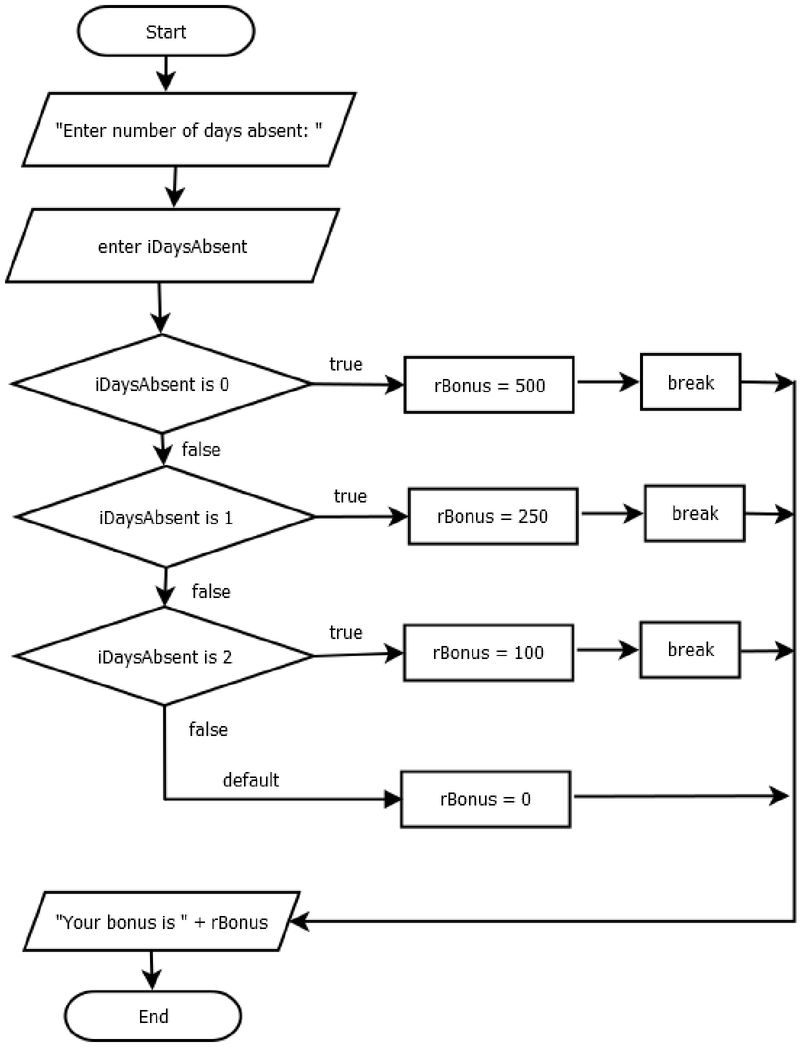
You can also write:

switch based on iDaysAbsent

You can also write:

end switch

The flowchart for the program for Example 1 looks as follows:



Activity 28: Create flowcharts for switch statements

Draw a flow chart for each of the following pseudo codes and problem statements.

a) Pseudo code 1

```
display "Enter an operator(+, - , * , / or % ): "
enter cOperator
double rAnswer
display "Enter two integers: "
enter iNumberOne
enter iNumberTwo

case based on cOperator
     begin
          case '+'
               rAnswer= iNumberOne+iNumberTwo
               display "The sum was calculated and the answer is " + rAnswer
               break
          case '-'
               rAnswer= iNumberOne-iNumberTwo
               display "The difference was calculated and the answer is " + rAnswer
                break
          case '/'
               rAnswer= iNumberOne/iNumberTwo
               display "The quotient was calculated and the answer is " + rAnswer
               break
          case '*'
               rAnswer= iNumberOne * iNumberTwo
               display "The product was calculated and the answer is " + rAnswer
               break
          case '%'
               rAnswer = iNumberOne / iNumberTwo
               display "The remainder was calculated and the answer is " + rAnswer
               break
          default
               display "Error, you chose the wrong operator."
     end case
```

b) Pseudo code 2

```
display "Enter your course (<1>: Telecom, <2>: mechatronics, <3>: industrial) "
enter iChoice
double rAverage
display "Enter math percentage: "
enter iMath
display "Enter electronics percentage: "
enter iElect
display "Enter project percentage: "
enter iProject

case based on iChoice
    begin
        case 1
            rAverage = (iMath+iProject)/2.0
            message = " Telecom "
            break
        case 2
            rAverage = (iMath+iElec)/2.0
            message = " Mechatronics "
            break
        case 3
            rAverage = (iElec+iProject)/2.0
            message = " Industrial "
            break

        default
            message = " an invalid "
            rAverage=0
    end case
    display "You chose" + message + "course and your average is " + rAverage
```

c) Many young students enquire about the different levels of emergency workers they may strive to qualify for. There are 5 different levels of emergency personnel based on the duration of their studies.

   a. A basic ambulance attendant (BAA) will have a minimum of 160 hours of lectures and practical simulations.
   b. An ambulance emergency assistant (AEA) has to complete a four-month course held at a recognized training facility.
   c. A critical care assistant (CCA) has to successfully complete a nine-month course at the Ambulance Training College's.
   d. A National Diploma- EMC (NEMC): has to complete a three-year course at a Technicon & practical year at a recognized training service.
   e. A Bachelor of Technology-EMC(BEMC) is a four-year professional degree in Emergency Medical Care (Bachelor Emergency Medical Care).

   Draw a flow chart that asks the user their choice between BAA, AEA, CCA, NEMC and BEMC. Display the minimum time their studies will take based on their choice (use switch statement).

d) To make a small cup of burger sauce one needs tomato sauce and mustard sauce in the ratio of 3:1. One person can buy up to 3 cups. If one cup of a burger sauce is 40 ml. ask the user how many cups they would like to buy and tell them the amount of tomato sauce and the amount of mustard that will be required for their order.

Activity 29: Write programs using a switch statement

Write Java programs for the following scenarios. You can decide which planning you want to use – pseudo code or a flowchart.

a) Write a program where the user can enter a number between 1 and 5 (both included). If a number out of range is entered, an error message should be displayed. For each number, the name of the fraction with the number as denominator should be displayed.

1	Whole
2	Half
3	Third
4	Quarter
5	Fifth
any other number	Invalid number

b) Write a java program that will allow the user to enter the height and the base, and ask the user what shape (triangle, parallelogram or trapezium) they would like to build and calculate the area of the shape they choose.

- Triangle area: ½ base x height
- Parallelogram: base x height
- If they choose the trapezium, the user needs to ask for the length of the small base. Then calculate the area of the trapezium: ½ (small base + base) x height

If another shape is chosen, an error message should be displayed

c) A single dose of Panado paediatric syrup is given to infants and children for the relief of mild to moderate pain and fever. Considering the information in the following table, write a Java program that will ask the user the category their child belongs to and display the amount of Panado that should be given to the child as a single dose. If they child belongs to category <0>, the user must enter the weight of the infant. Display an error message when a wrong category is entered.

Age	**Category**	**Dosage in mg** (*minimum dosage is considered*)
2-3 months	0	10 mg/kg
3 months to 1 year	1	60 mg
1 year to 5 years	2	120 mg
6 years to 12 years	3	240 mg
	Any other category	Invalid category

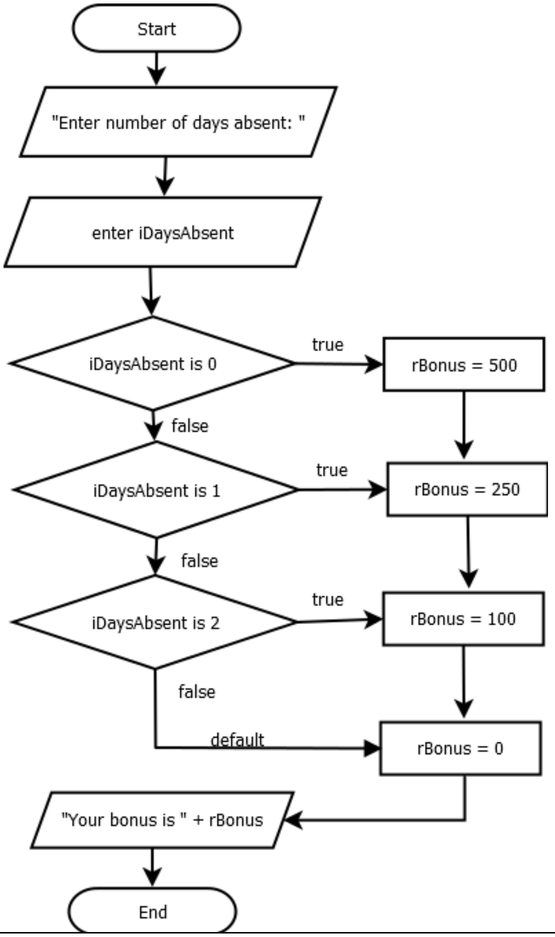
4.19.3 Using fall through in switch statements

If you forget to add break statements at the end of the statements for each case label, you may get unwanted results. Let us look at such a scenario. We are going to omit the break statements for the program in Example 1 (to calculate a person's bonus).

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

          int iDaysAbsent;
          double rBonus;

          System.out.print("Enter number of days absent: ");
          iDaysAbsent = keyboard.nextInt();

          switch (iDaysAbsent)
          {
               case 0: rBonus = 500;
               case 1: rBonus = 250;
               case 2: rBonus = 100;
               default: rBonus = 0;
          }
          System.out.print("Your bonus is " + formatter.format(rBonus));
     } //end main method
} //end class
```

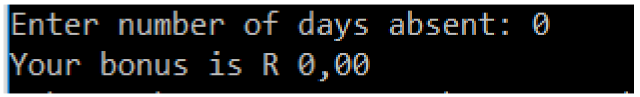
No break statements!

The flowchart for this program will look as follows:

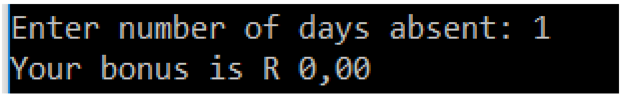


No break statements!

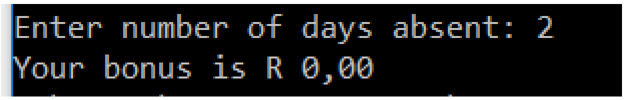
Here are examples of the output for this program:



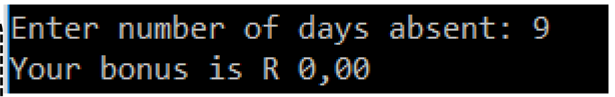
```
Enter number of days absent: 0
Your bonus is R 0,00
```



```
Enter number of days absent: 1
Your bonus is R 0,00
```



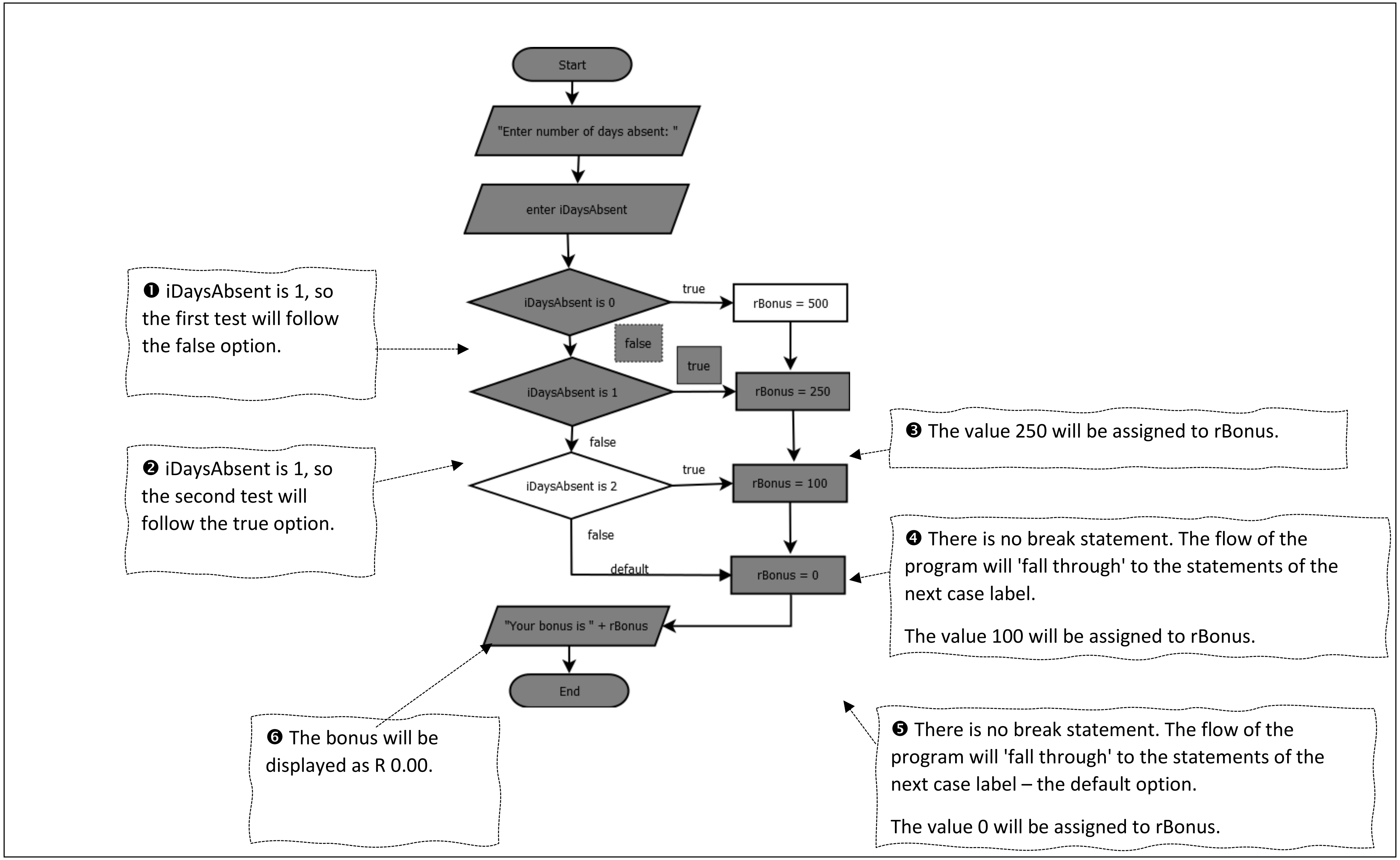
```
Enter number of days absent: 2
Your bonus is R 0,00
```



```
Enter number of days absent: 9
Your bonus is R 0,00
```

The value of rBonus is always 0, regardless of the number of days absent.

The output obtained from the program is not what you would like. In the next flowchart we show you what the flow of the program is when the value of iDaysAbsent is 1.



❶ iDaysAbsent is 1, so the first test will follow the false option.

❷ iDaysAbsent is 1, so the second test will follow the true option.

❸ The value 250 will be assigned to rBonus.

❹ There is no break statement. The flow of the program will 'fall through' to the statements of the next case label.

The value 100 will be assigned to rBonus.

❺ There is no break statement. The flow of the program will 'fall through' to the statements of the next case label – the default option.

The value 0 will be assigned to rBonus.

❻ The bonus will be displayed as R 0.00.

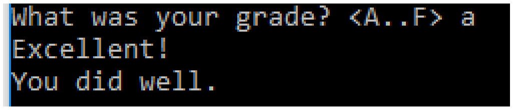
Even though the previous program's output was not what you wanted, the fall through of the switch statement can be used to obtain valid output for certain programs. Let us look at the following program (adapted from (Richard, 2018):

Example 1 of using a switch with fall through

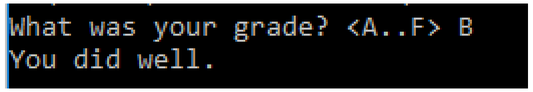
```
import java.util.Scanner;

public class FallThroughGrades
{ //start class
     public static void main(String[] args)
     { //start main method
       Scanner keyboard = new Scanner(System.in);
       char cGrade;
       System.out.print("What was your grade? <A..F> ");
       cGrade = keyboard.next().charAt(0);
       switch(Character.toUpperCase(cGrade))
       {
         case 'A' :
            System.out.println("Excellent!");
         case 'B' :
         case 'C' :
            System.out.println("You did well.");
            break;
         case 'D' :
         case 'E' :
            System.out.println("You passed");
            break;
         case 'F' :
            System.out.println("Better try again");
            break;
         default :
            System.out.println("Invalid grade");
      } //end switch
     } //end main method
} //end class
```

A message should be displayed based on the grade a student obtained. This is a summary of the message(s) for each symbol.

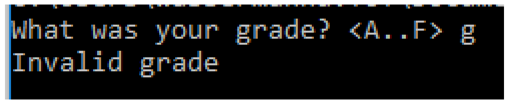
'A'	Excellent! You did well.
'B' and 'C'	You did well.
'D' and 'E'	You passed
'F'	Better try again
any other symbol	Invalid grade



```
What was your grade? <A..F> a
Excellent!
You did well.
```

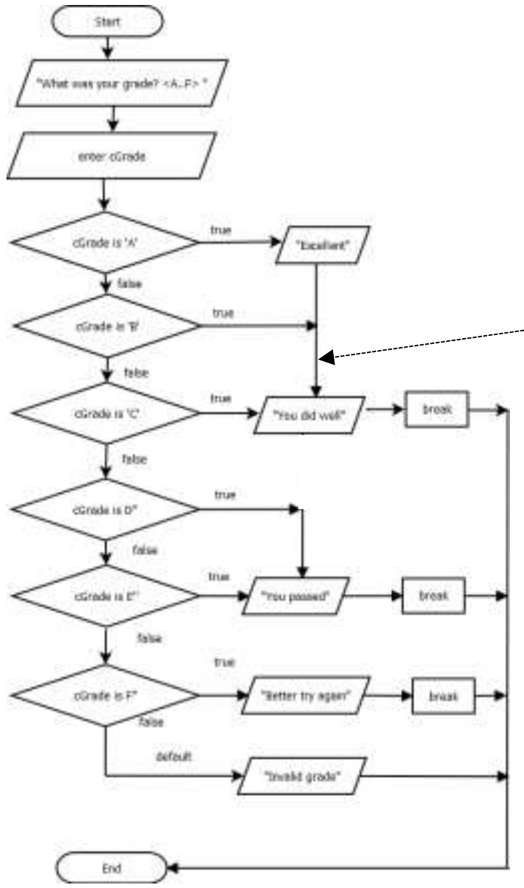


```
What was your grade? <A..F> B
You did well.
```



```
What was your grade? <A..F> g
Invalid grade
```

The flowchart for this program will look as follows:



When the value of cGrade is 'A', notice how the flow of the program 'falls through' to the display statement for the case label where cGrade is 'C'.

Revision

Before we look at the next example, we want to do revision of the algorithm to determine whether a year is a leap year or not. In the next example, we are going to write a program using a switch statement with fall through to calculate the number of days in a month for a specific year. February has 28 days, but in a leap year it has 29 days. So, we need to determine whether a year is a leap year or not before being able to say how many days February had in that year. We want you to focus on using the switch with fall through, and not worry about determining whether the year is a leap year or not. Therefore – look at the following versions of the algorithm.

Version 1 to determine whether a year is a leap year or not

```
import java.util.Scanner;
public class LeapYearV1
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iYear;
        System.out.print("Enter the year: ");
        iYear = keyboard.nextInt();

        if (((iYear % 4 == 0) && (iYear % 100!= 0)) || (iYear % 400 == 0))
            System.out.print(iYear + " is a leap year");
        else
            System.out.print(iYear + " is NOT a leap year");
    } //end main method
} //end class
```

Version 2 to determine whether a year is a leap year or not

```
import java.util.Scanner;
public class LeapYearV2
{ //start class
    public static void main(String[] args)
    { //start main method
```

```
        Scanner keyboard = new Scanner(System.in);
        int iYear;
        System.out.print("Enter the year: ");
        iYear = keyboard.nextInt();
        if (iYear % 400 == 0) System.out.print(iYear + " is a leap year");
        else
        {
            if (iYear % 100 == 0) System.out.print(iYear + " is NOT a leap year");
            else
            {
                if (iYear % 4 == 0)
                    System.out.print(iYear + " is a leap year");
                else
                    System.out.print(iYear + " is NOT a leap year");
            } //end else
        } //end else

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

Note: The braces { } are not necessary in the else-blocks of the nested if .. else statement. It was added to improve readability.

Version 3 to determine whether a year is a leap year or not

```
import java.util.Scanner;
public class LeapYearV3
{ //start class
    public static void main(String[] args)
    { //start main method
        Scanner keyboard = new Scanner(System.in);
        int iYear;
        System.out.print("Enter the year: ");
        iYear = keyboard.nextInt();
```

```
          if (iYear % 100 == 0) //year ends in 00
          {
               if (iYear % 400 ==0 ) System.out.print(iYear + " is a leap year");
               else System.out.print(iYear + " is NOT a leap year");
          }
          else //year does not end in 00
          {
               if (iYear % 4 == 0) System.out.print(iYear + " is a leap year");
               else System.out.print(iYear + " is NOT a leap year");
          }

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

Version 4 to determine whether a year is a leap year or not

```
import java.util.Scanner;
public class LeapYearV4
{ //start class
     public static void main(String[] args)
     { //start main method
          Scanner keyboard = new Scanner(System.in);
          int iYear;
          boolean bLeap;
          System.out.print("Enter the year: ");
          iYear = keyboard.nextInt();

          bLeap = (iYear % 100 == 0) ? (iYear % 400 == 0) : (iYear % 4 == 0);

          if (bLeap) System.out.print(iYear + " is a leap year");
          else System.out.print(iYear + " is NOT a leap year");
     } //end main method
} //end class
```

This version is based on version 3 and uses a conditional (ternary) operator.

Example 2 of using a switch with fall through

The following program will read a year number, and a month number, then display the number of days for the month in that year.

```
import java.util.Scanner;
public class DaysInMonthForYear
{ //start class
     public static void main(String[] args)
     { //start main method
          Scanner keyboard = new Scanner(System.in);
          int iYear, iMonthNum = 0, iNumDays = 0;
          System.out.print("Enter the year: ");
          iYear = keyboard.nextInt();
          System.out.print("Enter the month number <1 - 12>: ");
          iMonthNum = keyboard.nextInt();

          switch (iMonthNum)
          {
            case 1: case 3: case 5:
            case 7: case 8: case 10:
            case 12:
                iNumDays = 31;
                break;
            case 4: case 6:
            case 9: case 11:
                iNumDays = 30;
                break;
            case 2:
                if (((iYear % 4 == 0) && !(iYear % 100 == 0)) || (iYear % 400 == 0))
                    iNumDays = 29;
                else
                    iNumDays = 28;
                break;
            default:
                System.out.println("Invalid month.");
          } //end switch
```

This is similar to:

```
if (iMonthNum = 1) || (iMonthNum = 3) || (iMonthNum = 5) ||
   (iMonthNum = 7) || (iMonthNum = 8) || (iMonthNum = 10) ) ||
   (iMonthNum = 12)
```

Fall through.

This is similar to:

```
if (iMonthNum = 4) || (iMonthNum = 6) || (iMonthNum = 9) ||
   (iMonthNum = 11)
```

```
          System.out.println("Month number " + iMonthNum + " in year " + iYear + " has " + iNumDays + " days.");

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

Let us create a trace table to determine the output of this algorithm if iYear = 2000 and iMonthNum = 2.

Instruction	iYear	iMonthNum	iNumdays	Value of switch expression	Outcome of if	Output
display						Enter the year:
enter	2000					
display						Enter the month number <1 - 12>:
enter		2				
switch				2		
if					true*	
assign			29			
display						Month number 2 in year 2000 has 29 days.

*Evaluation of the Boolean expression of the if .. else statement:

```
  (((iYear % 4 == 0) && !(iYear % 100 == 0)) || (iYear % 400 == 0))
= (((2000 % 4 == 0) && !(2000 % 100 == 0)) || (2000 % 400 == 0))
= (((0 == 0) && !(0 == 0)) || (0 == 0))
= ((true && !(true)) || true)
= ((true && false) || true)
= (false || true)
= true
```

Activity 30: Use switch statements with fall through

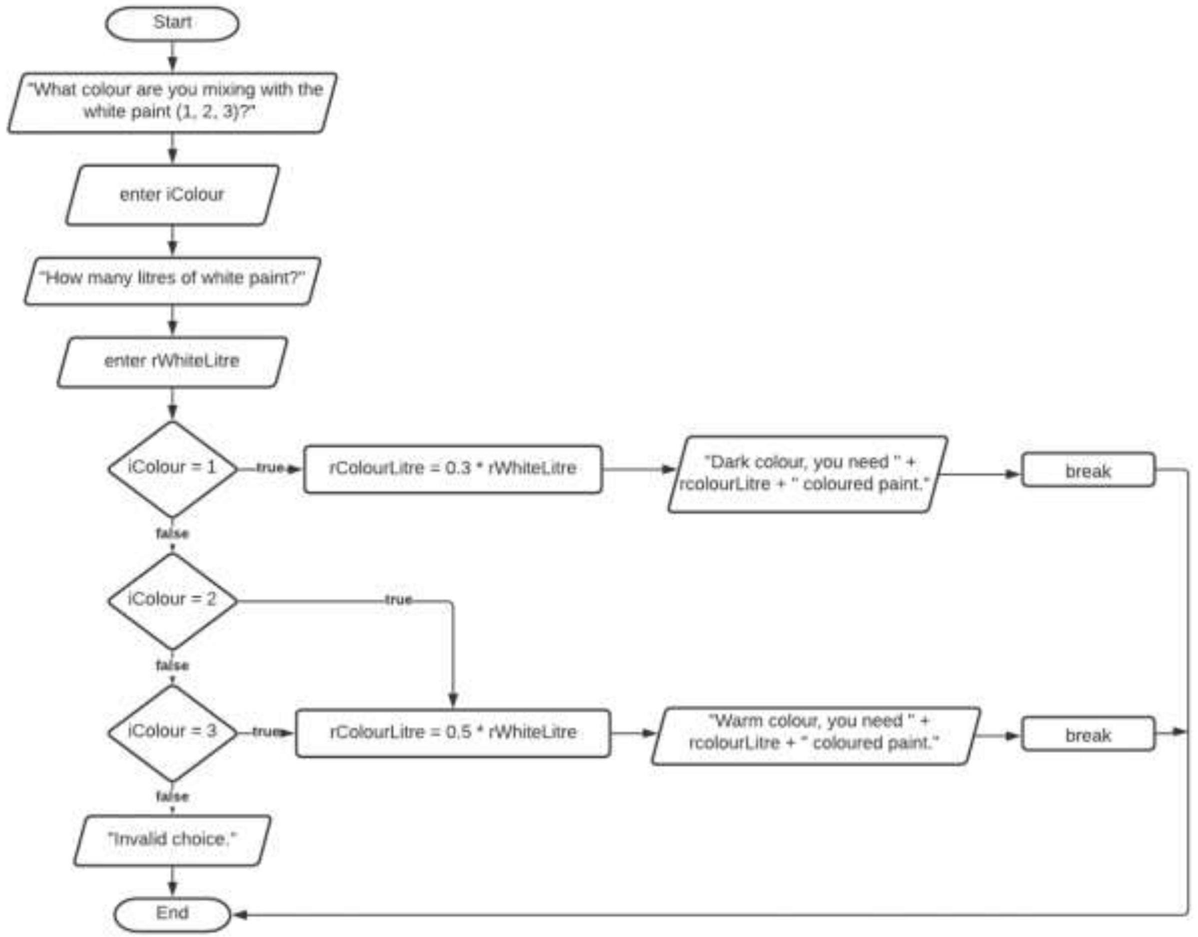
a) Draw a trace table to predict the output for Example 2 of using a switch with fall through when iYear is 1965 and iMonth = 7.

b) Draw a flowchart for the following problem statements and write a Java program. Each student must choose an extracurricular activity; the activity will take a certain percentage of their free time every week. They will enter their preferred extracurricular activity, and the number of free hours they have in a week. The program will calculate and display how much time the user will spend on that activity in a year. We can assume there will be 52.14 weeks in a year. Test your algorithm with a trace table (activity: dance, free hours per: 7hrs)

Activities	**Percentage of free time per week**
dance	40 % of your free hours
soccer	55%of your free hours
choir	40% of your free hours
Basketball	55 % of your free hours
karate	35% of your free hours

c) Considering the following flowchart.

- Predict the output using a trace table, test with iColour=2 and rWhiteLitre=25, and iColour=1, rWhiteLitre=15.
- Write the Java code for the following flowchart.


Start
"What colour are you mixing with the white paint (1, 2, 3)?"
enter iColour
"How many litres of white paint?"
enter rWhiteLitre
iColour = 1
true
rColourLitre = 0.3 * rWhiteLitre
"Dark colour, you need " + rcolourLitre + " coloured paint."
break
false
iColour = 2
true
false
iColour = 3
true
rColourLitre = 0.5 * rWhiteLitre
"Warm colour, you need " + rcolourLitre + " coloured paint."
break
false
"Invalid choice."
End

d) Predict the following output using a trace table (iMonth=8, iMonth=4)

```
Seasons// The class name
begin //start of the class
       main method
       begin //main method
              // declare variables
              int iMonth
              String sMessage
              // inputs
                display "Enter the month <1 - 12>: "
                enter iMonth
                case based on sMonth
                      begin
                           case 12: case 1: case 2:
                                sMessage = "Summer"
                                break
                           case 3: case 4: case 5:
                                sMessage = "Autumn"
                                break
                           case 6: case 7: case 8:
                                sMessage = "Winter"
                                break
                           case 9: case 10: case 11:
                                sMessage = "Spring"
                                break

                       default
                          sMessage = "Invalid Month"
                end case
               display sMessage
     end //main method
end //class
```

Activity 31: Knowledge checklist

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

	✓ / ?
Case structure	
Switch statement	
Case labels	
Switch expression	
Break statement	
Fall through	

Activity 32: Skills checklist

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

	✓ / ?
Convert if .. else statements to a switch statement	
Design an algorithm requiring a switch statement in pseudo code or using a flow chart.	
Determine whether a switch statement or a nested if .. else statement will be the most suitable structure for a specific scenario, taking the data type of the value(s) needed in the logical or switch expression into account.	
Convert algorithms (in pseudo code and using a flowchart) of solutions where a switch statement is required to Java and vice versa.	
Explain why a break statement is necessary in certain scenarios where a switch statement is used.	

Use fall through switch statements in a Java program.	
Predict output and correct errors of provided algorithms with switch statements using a trace table.	

Reflection

1. What was the purpose of the lesson content you are discussing?
2. What did you learn that will advance your future learning?
3. What new knowledge and skills did you learn today? (Tip: To help you to answer this question – if someone asks you tonight – “So, what did you learn in class today?” – what will you answer them? Your answers can, for example, start with “I am now able to….I managed to … I learned how to ….)
4. What did you figure out on your own?
5. What is still challenging or confusing for you to get your mind around?
6. What did you find most challenging today?
7. Are you now confident that you have mastered that difficult concept / skill?
8. If not – what are you going to do to make it work?
9. How was your thinking extended or pushed today?
10. How are the knowledge and skills you learned today connected to what you already knew?
11. Did you discover a connection today that you were not aware of previously?
12. What did you enjoy doing and why?
13. What do you think could be done differently in today’s lesson?
14. What did you discover that may make someone else’s life easier in future classes?

(Sackson, 2011)