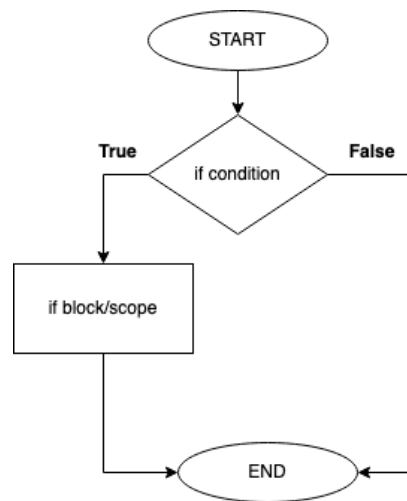


CHAPTER 7: DECISION MAKING

7.1 "If" SINGLE SELECTION STATEMENT

We are now going to look at conditional statements or branching. Sometimes we need to produce outputs based on conditions, for example, if it rains today then I will take an umbrella. In this example, taking an umbrella is dependent on the condition: rain. A simple flowchart is given below:



Here, we can see that an "if" block is executed only when the if condition holds true.

A sample code structure is given below:

```

public class Sample1{
    public static void main(String [] args){
        if (condition) {
            True
            block
        }
    }
}
  
```

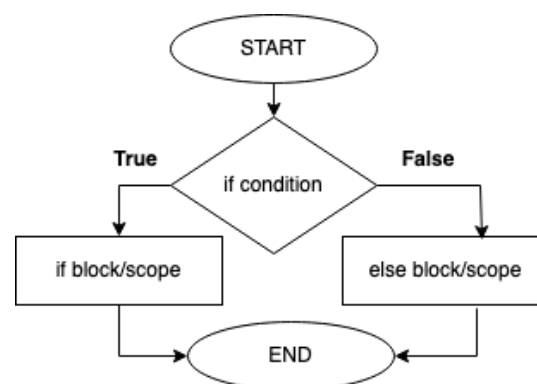
Here, we can see that the keyword "if" is followed by a "condition" and a pair of curly braces { }. In the next line, we can see leading whitespaces followed by a block of code. A block/scope of code is a collection of lines of codes. For example, the "True block" inside the curly braces may contain multiple lines of codes which will only be executed when the "if condition" is True. The leading whitespaces are called *indentation* which separates a block/scope of code from the rest of the lines of code. Maintaining indentation is crucial in many programming languages. Let's see another example. Here we want to print "num1 is greater" if the first number is greater.

<u>Flowchart</u>	<u>Code</u>
	<pre> public class Sample2{ public static void main(String [] args){ int num1 = 9; </pre>

<pre> graph TD START([START]) --> num1[num1 = 9] num1 --> num2[num2 = 4] num2 --> cond{num1 > num2} cond -- True --> print[/PRINT "num1 is greater"/] print --> END([END]) cond -- False --> END </pre>	<pre> int num2 = 4; if (num1 > num2){ System.out.println("num1 is greater"); } </pre>
<p>Output: num1 greater</p>	<p>Explanation: num1 variable holds the value 9 and num2 variable holds the value 4 so in the if condition we are checking if num1 (9) is greater than num2 (4) or not, since 9 is greater than 4, the condition yields True and the if scope is executed. Therefore, "num1 is greater" is printed. Note that if num1 was any value smaller than num2 then the if scope would not be executed and nothing would be printed.</p>

7.2 "If - Else" DOUBLE SELECTION STATEMENT

Sometimes we might need to execute a block of code when the if condition yields False. For example, if it rains today then I will stay home, else I will go out. In this example, there are two possibilities which depend on the condition: rain. If the condition is True i.e. if it rains then the outcome is staying home but if the condition is False then the outcome will be going out. A simple flowchart is given below:



Here, we can see that an "if" block is executed only when the if condition holds true and the "else" block is executed only when the if condition holds false.

A sample code structure is given below:

```

public class Sample3{
    public static void main (String []
args){
        if (condition) {
            True block
        }
        else {
            False block
        }
    }
}

```

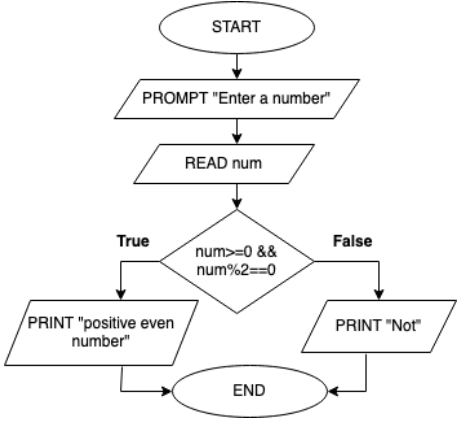
Here, we can see the keyword “else” followed by a pair of curly braces { } which indicate the scope of the else block separated by an indentation. The else block of code is only executed when the if condition yields false. Notice that we do not need to write any conditional statement beside the “else” keyword.

Let’s see the previous example again. Now, we want to print “num1 is greater” if the first number is more than the second number or print “num2 is greater” otherwise.

Flowchart	Code
<pre> graph TD START([START]) --> N1[num1 = 7] N1 --> N2[num2 = 11] N2 --> D{num1 > num2} D -- True --> P1[/PRINT "num1 is greater"/] D -- False --> P2[/PRINT "num2 is greater"/] P1 --> END([END]) P2 --> END </pre>	<pre> public class Sample4{ public static void main(String [] args){ int num1 = 7; int num2 = 11; if (num1 > num2){ System.out.println("num1 is greater"); } else{ System.out.println("num2 is greater"); } } } </pre>
<p>Output: num2 greater</p>	<p>Explanation: num1 variable holds the value 7 and num2 variable holds the value 11 so in the if condition we are checking if num1 (7) is greater than num2 (11) or not, since 7 is NOT greater than 11, the condition yields False and the else scope is executed. Therefore, “num2 is greater” is printed. Note that if num1 was any value larger than num2 then only the if scope would be executed and “num1 is greater” would be printed.</p>

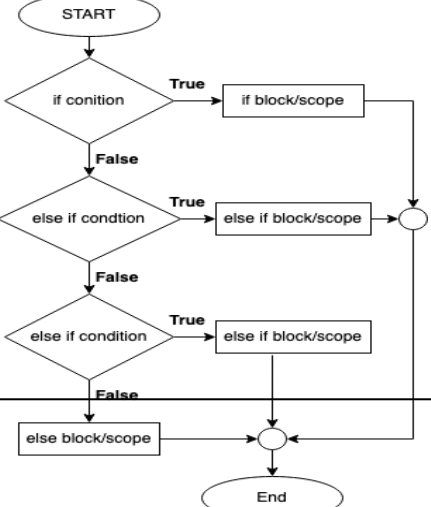
We can use Logical Operators within if conditions as well whenever there are multiple conditions that need to be checked at a time.

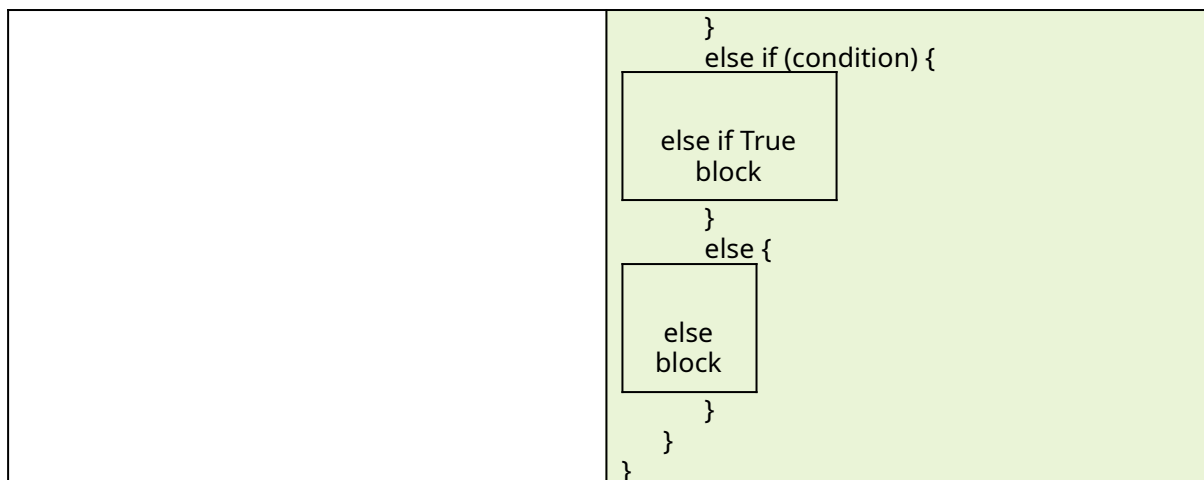
Let's see an example where we need to take an integer input from the user and determine if the number is a positive even number or not.

Flowchart	Code
 <pre> graph TD START([START]) --> PROMPT[/PROMPT "Enter a number"/] PROMPT --> READ[/READ num/] READ --> DEC{num >= 0 && num % 2 == 0} DEC -- True --> PRINT1[/PRINT "positive even number"/] DEC -- False --> PRINT2[/PRINT "Not"/] PRINT1 --> END([END]) PRINT2 --> END </pre>	<pre> import java.util.Scanner; public class Sample5{ public static void main(String [] args){ Scanner sc = new Scanner(System.in); int num= sc.nextInt(); if (num>=0 && num%2==0){ System.out.println("positive even number"); } else{ System.out.println("not"); } } } </pre>
Input: 18	Output: positive even number Explanation: The user input is 18 and in the if condition it is checked $18 \geq 0$ which is True and $18 \% 2 == 0$ which is also True. Since True and True yields True so the condition is True overall. Hence, "positive even number" is printed.

7.3 "If - ELSE IF - ELSE" MULTIPLE SELECTION STATEMENT

Sometimes we might need to execute different blocks of code depending on multiple conditions. These types of problems can be solved using chained conditions i.e. (if.... else if.... else if.... else). These conditions follow a top-down approach during execution. First, the if condition is checked and if it yields False, then the next else if condition is checked and if it yields False then the next else if condition is checked and so on. If none of the else if conditions are satisfied then the final else block would be automatically executed. One if block may be followed by multiple else if blocks and a final else block. Among these several conditions, only one condition block is executed when it yields True and so the rest of the bottom conditional statements in the chain will not be checked further. A simple flowchart and sample code structure is given below:

Flowchart	Code
 <pre> graph TD START([START]) --> IF{if condition} IF -- True --> IF_BLOCK[if block/scope] IF -- False --> ELSE_IF1{else if condition} ELSE_IF1 -- True --> ELSE_IF1_BLOCK[else if block/scope] ELSE_IF1 -- False --> ELSE_IF2{else if condition} ELSE_IF2 -- True --> ELSE_IF2_BLOCK[else if block/scope] ELSE_IF2 -- False --> ELSE_BLOCK[else block/scope] IF_BLOCK --> MERGE(()) ELSE_IF1_BLOCK --> MERGE ELSE_IF2_BLOCK --> MERGE ELSE_BLOCK --> MERGE MERGE --> END([End]) </pre>	<pre> public class Sample6{ public static void main(String [] args){ if (condition) { if True block } else if (condition) { else if True block } } } </pre>

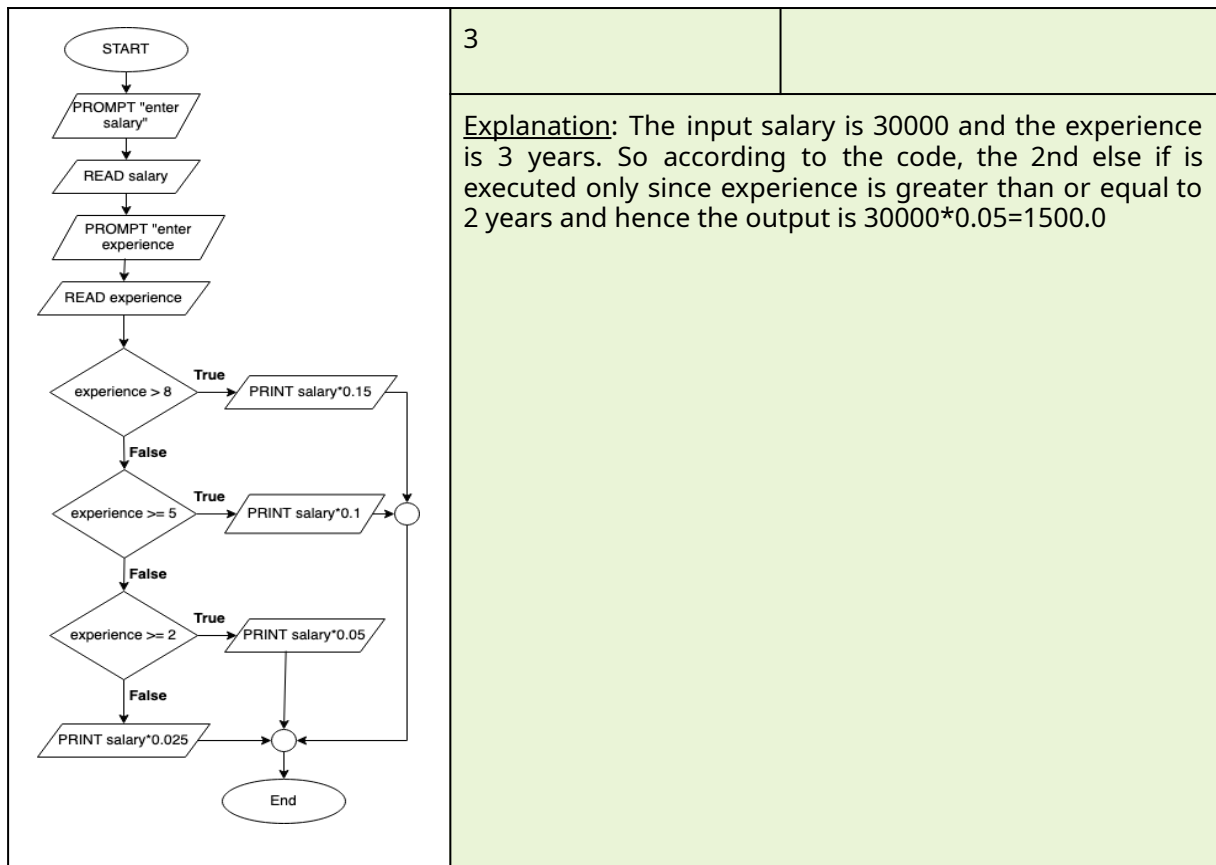


In the sample code it can be seen that conditional statements are placed after “else if” keywords. The else block will only be executed if none of the above if or else if conditions yield True.

Remember that, a single “if” block can exist on its own without the help of any “else if” block or “else” block. Also, an “if” block followed by one or multiple “else if” blocks can also exist without the help of any “else” block. But, only “else” blocks or “else if” blocks cannot stand along without any initial “if” block.

Let’s see an example. We will take the salary and work experience of an employee as an input and calculate the bonus based on their experience. If the work experience is over 8 years, then 15% bonus is awarded, if the work experience is 5 years - 8 years then 10% bonus is awarded, if the work experience is 2 years - 4 years then 5% bonus is awarded and work experience below 2 years will be awarded with 2.5% bonus.

<u>Flowchart</u>	<u>Code</u>	
	<pre> import java.util.Scanner; public class Sample7 { public static void main(String [] args) { Scanner sc = new Scanner(System.in); int salary = sc.nextInt(); int experience = sc.nextInt(); if(experience > 8){ System.out.println(salary*0.15); } else if(experience >= 5){ System.out.println(salary*0.1); } else if(experience >= 2){ System.out.println(salary*0.05); } else{ System.out.println(salary*0.025); } } } </pre>	
	Sample Input: 30000	Output: 1500.0

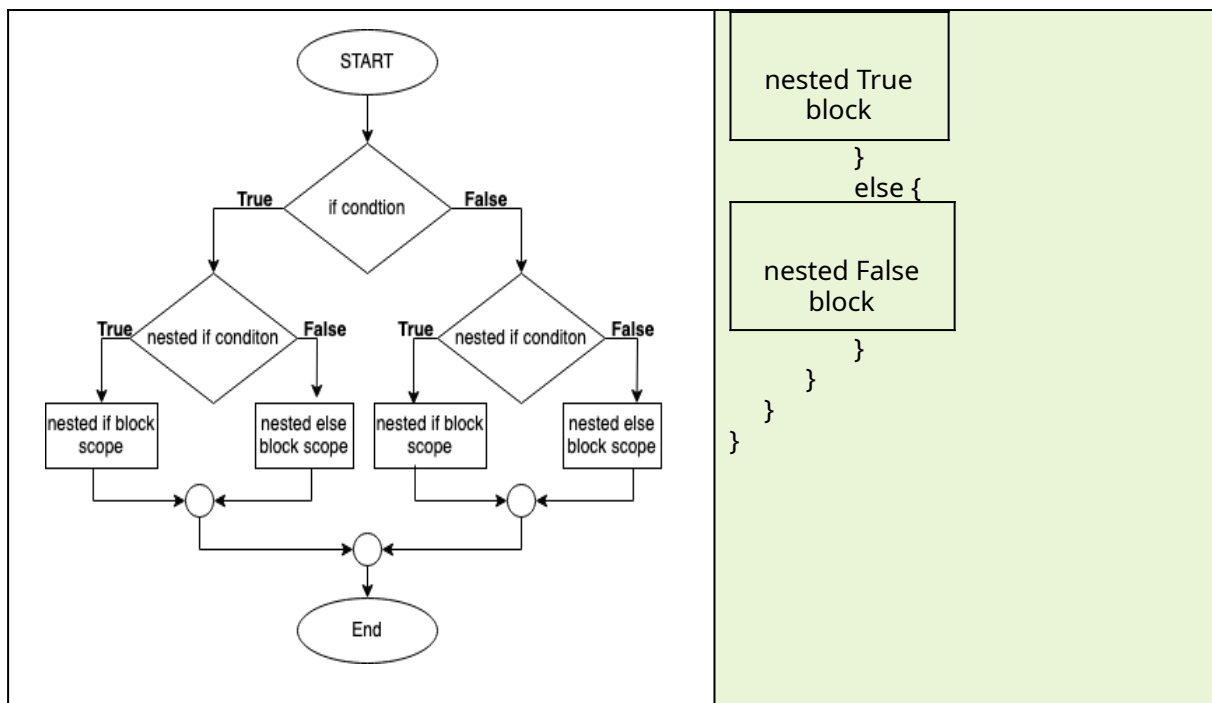


7.4 NESTED DECISION MAKING

Multiple if...else if...else blocks can be put inside an if block and/or an else if block and/or an else block. These blocks would be called nested blocks. The inner code blocks would only be executed if the outer conditional block yields True. In these programs, indentation is crucial to distinguish between nested code blocks.

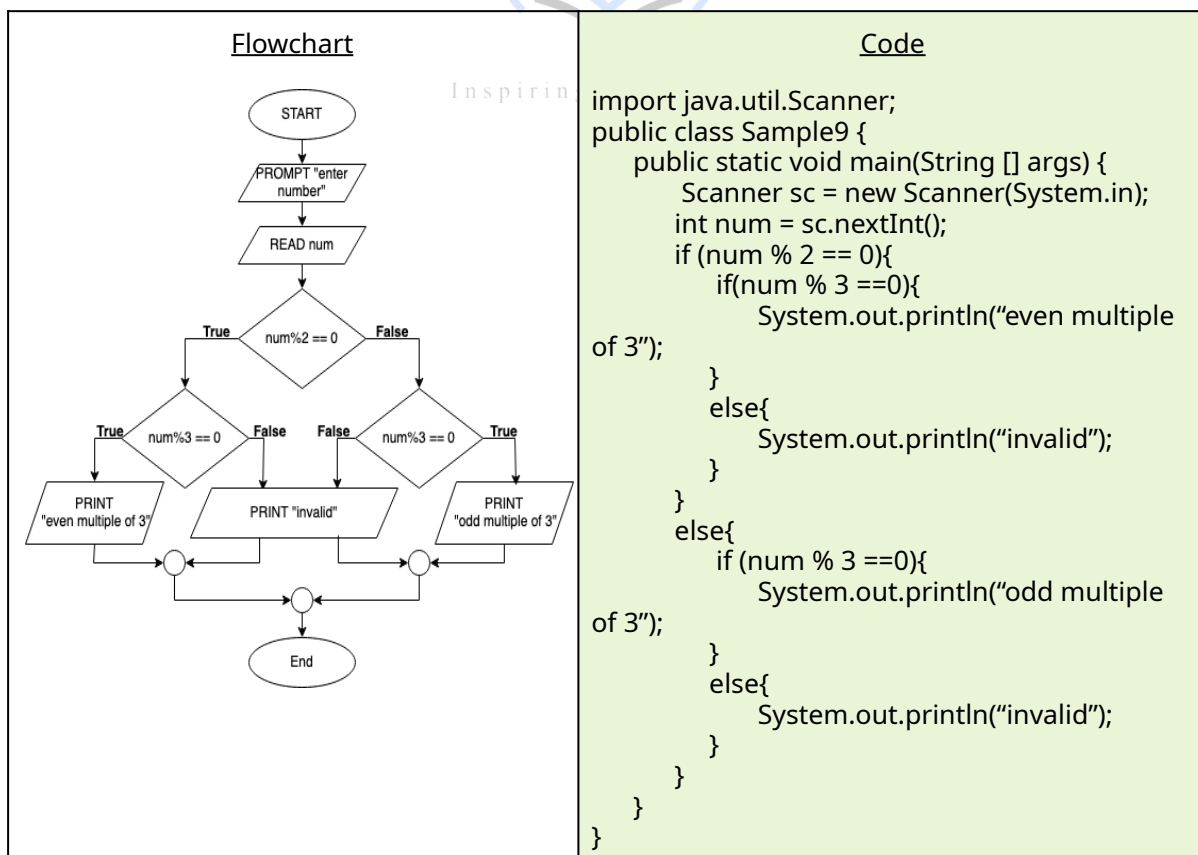
A simple flowchart and sample code structure is given below:

Flowchart	Code
	<pre> public class Sample8{ public static void main (String [] args){ if (condition) { if (condition) { nested True block } else { nested False block } } else { if (condition) { </pre>



In the sample code, it can be seen that the nested if/else blocks follow their own indentation. Remember, the nested blocks also follow the same convention as any basic if...else if...else block that we learned in the above sections.

Let's see an example. We want to print "even multiple of 3" if a number is an even number and a multiple of 3, "odd multiple of 3" if the number is an odd number and a multiple of 3 and "invalid" otherwise. [Note: this problem can also be solved using "and" logical operator and "if...else if...else" blocks. Try it out on your own.]



	Input: 33	Output: odd multiple of 3
	Explanation: The input is 33 for which $33\%2$ is not equal to 0 so it goes to the else block and then inside the else block 33 is again checked for which $33\%3$ is equal to 0 and so "odd multiple of 3" is printed.	

7.5 SWITCH CASE CONDITIONING

Another way of handling the decision making process is called **switch statement** or **switch case conditioning**. It can be used as an alternative to writing many *if ... else if ... else* statements. Here, the syntax is more clear and easy to write & understand.

<p>Syntax</p> <pre>switch(expression){ case value1 : // Code Statements break; // optional case value2 : // Code Statements break; // optional . . . case valueN : // Code Statements break; // optional default: // optional // Default Code Statements }</pre>	<ul style="list-style-type: none"> • There can be one or multiple case values for a switch expression. The expression is evaluated once and compared with the values of each case. • The case values must be of switch expression type only; variables are not allowed. Also, the case values need to be unique. In case of duplicate values, it renders an error. • If there is a match, the associated block of code is executed. For example, if expression matches with value1, the code of case value1 is executed. Similarly, the code of case value2 is executed, if expression matches with value2 and so on. • The optional break statement is used inside the cases to terminate a statement sequence. If it is omitted, execution will continue on into the next case. • If there is no match, the code of the optional default case is executed. It can appear anywhere inside the switch block. In case, if it is not at the end, then a break statement must be kept after the default statement to omit the execution of the next case statement.
---	---

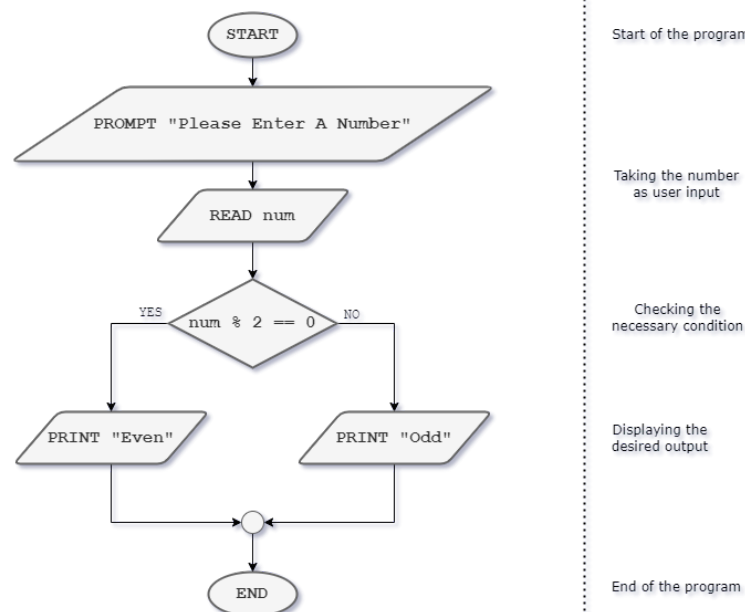
In summary, the switch statement evaluates an expression and compares it to case values of the same type. When a match is found, the associated block of code is executed. The break statement is used to exit the switch block, and the default statement is executed if there is no match.

Let's see an example.

7.6 FLOWCHARTS

Now, let's look at some relevant examples through flowcharts.

- ✓ **Scenario 1:** Let's design a flowchart of a program that will take a number as user input and determine whether the number is even or odd. [*Hint: If you try to divide any even number in this universe by 2, you will always end up with a remainder value 0 (zero).*]

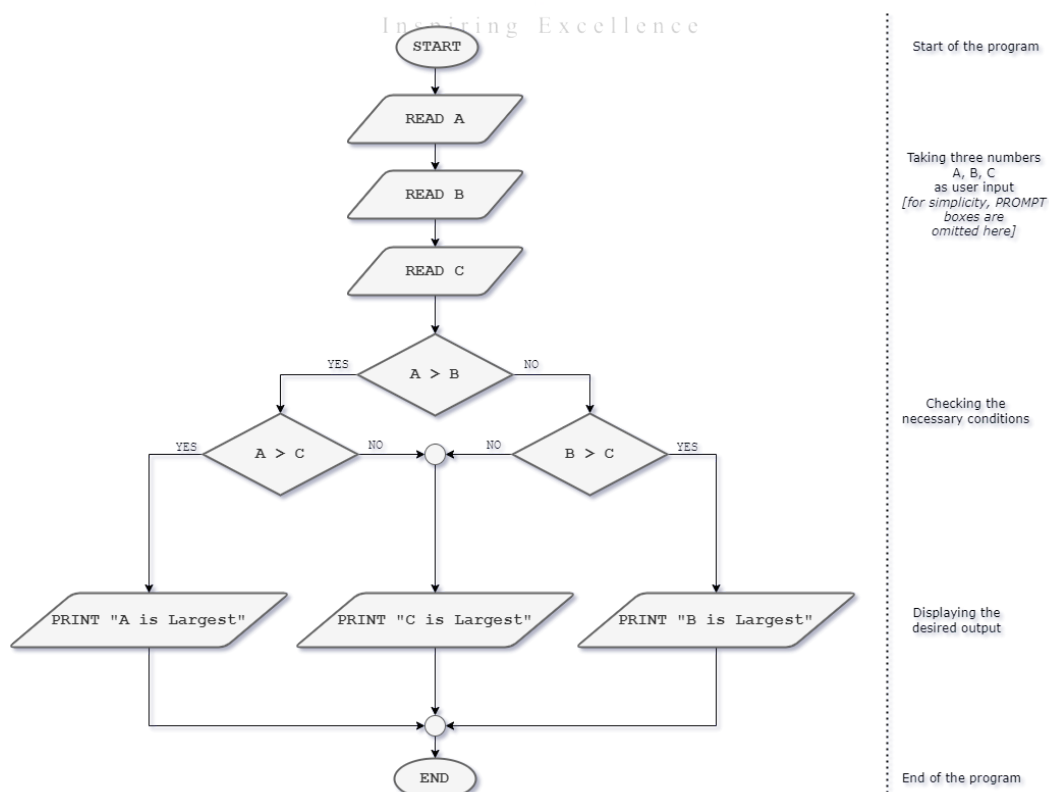


- ✓ **Scenario 2:** Let's consider three numbers A, B and C. We need to find out which number is the largest among these three and draw a flowchart for that. Let's assume, for this problem all the values of A, B & C are unique; that means there will be no duplicate values.

Before going to design the flowchart, let's break down the problem and formulate the necessary conditions first.

- If A is greater than B and A is greater than C, then definitely A will be the largest.
- If A is not greater than B (which means B is greater than A) and B is greater than C, then B will be the largest.
- If the above two conditions are not fulfilled, then obviously C will be the largest.

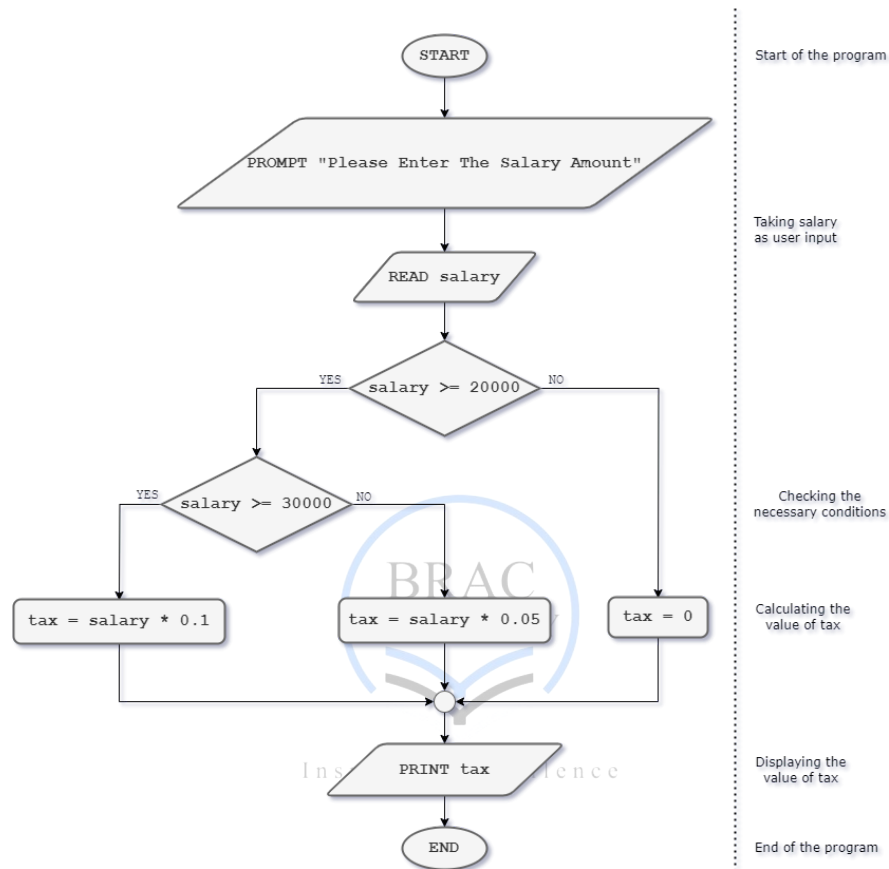
Now, let's look at the flowchart below.



✓ **Scenario 3:** Let's calculate some taxes again. Suppose, you need to calculate the tax amount an employee needs to be paid based on his/her salary. Here are the requirements:

- If the salary of the employee is below 20,000 Taka, then no tax.
- If the salary is 30,000 Taka or above, then the tax will be 10% of the monthly salary.
- If the salary is 20,000 Taka or above and less than 30,000 Taka, then the tax will be 5% of the monthly salary.

Let's design the flowchart of this problem.



7.7 WORKSHEET

- A. Consider the following code. Suppose, the desired output should be: "A is greater than 100". What is the problem in this code? How can you overcome this problem? *[There can be multiple ways to solve the issues.]*

```
class SimpleCode {
    public static void main(String[] args) {
        int a = 120;
        if (a > 10) {
            System.out.println("A is greater than
10");
        }
        else if (a > 100){
            System.out.println("A is greater than
100");
        }
        else {
            System.out.println("A is less than 10");
        }
    }
}
```

- B. Suppose, you are dealing with 3 conditions. You are trying to implement these conditions using if → else if → else if blocks. So, clearly the "else" block is missing. But the code will not give any errors. So, what do you think of omitting this "else" block? In which cases you can apply this technique?
- C. Consider the following incomplete code.
- At least how many lines of "Hello" will be printed? Explain briefly.
 - Maximum how many lines of "Hello" will be printed? Explain briefly.

```
if [condition]
    System.out.println("Hello");
if [condition]
    System.out.println("Hello");
else
    System.out.println("Hello");
if [condition]
    System.out.println("Hello");
else if [condition]
    System.out.println("Hello");
else
    System.out.println("Hello");
if [condition]
    System.out.println("Hello");
```

- D. Suppose, there are two boolean variables a and b. The value of a is set to true and the value of b is set to false. What will be the result of the following expression? Briefly explain.

```
!(a && false) && (a || !b) || (b && !a)
```

- E. Design the flowchart and also write the Java code for the following problems.
- Take a number from the user and find out if the number is positive or negative or zero.
 - Take a number as user input and calculate the absolute value of the number. *[For example, absolute value of 5 is 5 and absolute value of -5 is 5.]*
 - Find out the largest number among four numbers taken from the user.
 - Find out the second largest number / second smallest number among three numbers taken from the user. *[You are not allowed to use a loop.]*

- F. Suppose, on normal days, a worker usually works a maximum of 8 hours daily. On some special days, after 8 hours of work, overtime is counted. Suppose,
- For the first 8 hours, the worker gets 200 taka hourly.
 - For the next 2 hours of overtime, the worker will receive 250 taka hourly.
 - For another 2 hours of overtime, the worker will receive 300 taka hourly.

Your first task is to write a Java program that takes the daily work hour of that worker as input. Then calculate the amount the worker will receive based on the criteria mentioned above. If the daily work hour is zero or negative or greater than 12, then display "Invalid".

Sample Input	Sample Output	Explanation
7	1400	$7 \times 200 = 1400$
9	1850	$(8 \times 200) + (1 \times 250) = 1850$
12	2700	$(8 \times 200) + (2 \times 250) + (2 \times 300) = 2700$
-4	Invalid	Invalid case, because the input is negative.

- G. Let's assume, these are the grading criteria of a particular course.

Marks Range	Grade	Comment
80 - 100	A	Excellent
70 - 79	B	Good
60 - 69	C	Satisfactory
50 - 59	D	Okay
0 - 50	F	Fail

Your task is to write a Java code where you need to take a student's obtained marks as user input. Then based on the above criteria, display the grade the student got and also display the corresponding comment from the table. If the user enters a number that is outside the valid range 0-100, then display the line "Invalid Input".

- H. Write a Java program that takes an integer number as user input and print "Yes" if the number is divisible by: *(For each subproblem, there will be separate codes)*
- either 3 or 5
 - 3 and 5 both
 - 3 but not 5
 - 5 but not 3
 - 3 or 5 but not both
 - neither 3 nor 5

Print "No" otherwise.