CONDITIONAL AND LOOPS

Condition:- It provide check for the statement.

1. If-else statement → Used to check the condition, it checks the Boolean condition True or False.
   Syntax :-

```
if (boolean expression True or false){
    //Body
} else{
    // Do this
}
```

Example:-

```
public class IfElse {
    public static void main(String[] args) {
        int salary = 25400;
        if (salary> 10000) {
            salary = salary + 2000;
        }
        else {
            salary = salary + 1000;
        }

        System.out.println(salary);

    }
}
```

Output :- 27400

2. Multiple if-else statement

→ It executes one condition from multiple statements.

Syntax :-

```
if (condition 1){
    // code to be executed if condition 1 is true
} else if (condition 2) {
    // code to be executed if condition 2 is true
} else if (condition 3){
    // code to be executed if condition 3 is true
} else {
    // code to be executed if all conditions are false
}
```

Example :-

```
public class MultipleIfElse {
    public static void main(String[] args) {
        int salary = 25400;
        if (salary<= 10000) {
            salary +=1000;
        }
        else if (salary <= 20000) {
            salary += 2000;
        }
        else {
            salary += 3000;
        }
        System.out.println(salary);

    }
}
```

Output :- 28400

Loop → Loops are used to iterate a part of program several times.

1. for loop :- It is generally used when we know how many times loop will iterate.

   Syntax :-

```
for (initialization; condition; increment/decrement){
    // body
}
```

Example:- print numbers from 1 to 5

```
public class forloop {
    public static void main(String[] args) {
        for (int num=1;num<=5;num+=1){
            System.out.println(num);
        }
    }
}
```

Output :- 1
2
3
4
5

Example 2 :- print numbers from 1 to n

```
import java.util.Scanner;
public class forloop {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        for (int num=1;num<=n;num+=1){
            System.out.print(num + " ");
        }
    }
}
```

Input : 6

Output :- 1 2 3 4 5 6

2. While Loop :- It is used when we don't know how many time the loop will iterate.

   Syntax :-

```
while (condition){
    // code to be executed
    // increment/decrement
}
```

   Example :-

```
public class whileloop {
     public static void main(String[] args) {
         int num = 1;
         while (num <=5){
             System.out.println(num);
             num += 1;
         }
     }
}
```

   Output :- 1
   2
   3
   4
   5

3. do while loop :- It is used when we want to execute our statement at least one time.
   → It is called exit control loop because it checks the condition after execution of statement.
   Syntax :-

```
do{
    // code to be executed
    // update statement -> increment/decrement
}while (condition);
```

Example :-

```
public class doWhileloop {
    public static void main(String[] args) {
       int n = 1;
       do{
           System.out.println(n);
           n++;
       } while(n<=5);
    }
}
```

Output :- 1
2
3
4
5

While Loop	Do while loop
→ used when no. of iteration is not fixed	→ used when we want to execute the statement at least ones
→ Entry controlled loop	→ Exit controlled loop
→ no semicolon required at the end of while (condition)	→ semicolon is required at the end of while (condition)

■ Program to find largest of three numbers.

"Take 3 integer input from keyboard, Find the largest numbers among them ".

Approach -1 :-

```
import java.util.Scanner;
public class LrgestOfThree {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = in.nextInt();
        int b = in.nextInt();
        int c = in.nextInt();

        int max = a;
        if(b>max){
            max = b;
        }
        if (c > max){
            max = c;
        }
        System.out.println(max);
    }
}
```

Approach – 2:-

```
import java.util.Scanner;
public class LrgestOfThree {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = in.nextInt();
        int b = in.nextInt();
        int c = in.nextInt();

        int max = 0;
        if(a > b){
            max = a;
        } else {
```

```
            max = b;
        }
        if (c > max){
            max = c;
        }
        System.out.println(max);
    }
}
```

Approach 3 :-

Using Math.max :- Math is a class present in java.lang package and max is a function present in it which takes two number as an argument and return maximum out of them.

```
import java.util.Scanner;
public class LrgestOfThree {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = in.nextInt();
        int b = in.nextInt();
        int c = in.nextInt();

        int max = Math.max(c,Math.max(a,b));
        System.out.println(max);
    }
}
```

Input :- 3 6 5

Output :- 6

■ Alphabet case check

" Take an input character from keyboard and check weather it is Upper case alphabet or lower case alphabet"

```
import java.util.Scanner;
public class AlphabetCaseCheck {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        char ch = in.next().trim().charAt(0);
        if (ch > 'a' && ch <= 'z'){
            System.out.println("Lowercase");
        }
        else {
            System.out.println("Uppercase");
        }
    }
}
```

Input :- a

Output :- Lowercase

Input :- Z

Output :- Uppercase

■ Fibonacci Numbers :- a series of numbers in which each number (*Fibonacci number*) is the sum of the two preceding numbers.

Ex :- 0,1,1,2,3,5,8,13...

→ **Find the nth Fibonacci number.**

" Given three input a, b, n a is the starting number of Fibonacci series and b is the next number after a, n is an number to find the nth Fibonacci number"

```
import java.util.Scanner;
public class FibonacciNumbers{
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        int n = in.nextInt();
        int a = in.nextInt();
        int b = in.nextInt();
        int count = 2;

        while(count <= n){
            int temp = b;
            b = b+a;
            a = temp;
            count++;
        }
        System.out.println(b);
    }
}
```

Input :- 0 1 7

Output :- 8.

- **Counting occurrence :-**

  " Input two numbers, find that hoe many times second number digit is present in first number"

  Ex :- first number = 14458

  Second number = 4

  Output = 2, because 4 is present 2 times in first number.

```
import java.util.Scanner;

public class CountingOccurence {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int count = 0;
        int Fn = in.nextInt();
        int Sn = in.nextInt();
        while (Fn>0){
            int rem = Fn % 10;
            if (rem == Sn){
                count++;
            }
            Fn = Fn/10;
        }
        System.out.println(count);
    }
}
```

Input :- 45535 5

Output :- 3

■ Reverse a number

" A number I input from the keyboard and Show the output as Reverse of that number "

Example :- Input :- 12345

Output :- 54321

```java
import java.util.Scanner;
public class ReverseANumber {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int num = in.nextInt();
        int ans = 0;
        while(num > 0){
            int rem = num % 10;
            num /= 10;
            ans = ans * 10 + rem;
        }
        System.out.println(ans);
    }
}
```

Input :- 458792

Output :- 297854

Calculator Program

```
import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // Take input from user till user does not press X or x
        int ans = 0;
        while (true) {
            // take the operator as input
            System.out.print("Enter the operator: ");
            char op = in.next().trim().charAt(0);

            if (op == '+' || op == '-
' || op == '*' || op == '/' || op == '%') {
                // input two numbers
                System.out.print("Enter two numbers: ");
                int num1 = in.nextInt();
                int num2 = in.nextInt();

                if (op == '+') {
                    ans = num1 + num2;
                }
                if (op == '-') {
                    ans = num1 - num2;
                }
                if (op == '*') {
                    ans = num1 * num2;
                }
                if (op == '/') {
                    if (num2 != 0) {
                        ans = num1 / num2;
                    }
                }
                if (op == '%') {
                    ans = num1 % num2;
                }
            } else if (op == 'x' || op == 'X') {
                break;
            } else {
                System.out.println("Invalid operation!!");
            }
            System.out.println(ans);
        }
    }
}
```

Input and output:-

```
Enter the operator: +
Enter two numbers: 86  94
180
Enter the operator: -
Enter two numbers: 75
12
63
Enter the operator: *
Enter two numbers: 12   3
36
Enter the operator: /
Enter two numbers: 70    5
14
Enter the operator: x
```

6/8/21 If-else conditions
Loops → while & for & do-while

7/8/21 Switch Statements + Nested case in Java.

• Switch Statements :

```
switch (expression) {
    case one:
        // code block
        break;
    case two:
        // code block
        break;
    default:
        // code block
}
```

break; → terminate the sequence

default: → default will execute when none of above does.
→ if default is not at end put break after it.

→ if break is not used then it will continue with other cases.

→ duplicate cases not allowed.

eg:
```
case one:
    // code block
    break;
case one:
    // code block
    break;
```

✗ not allowed.

New Syntax:

```
switch (expression) {
        case one -> //do this ;
        case two -> // do this;
        default -> // do this;
}
```

* `x.equals("word")` → here equals only checks value not reference.

`x == "word"` → here it checks reference

- **Nested Switch Case:**

```
switch (expression) {
        case one:
            // code block
            break;
        case two:
            switch (expression) {
                case one:
                    // code block
                    break;
                case two:
                    // code block
                    break;
                default;
                    // code block
            }
            break;
        default:
            // code block
}
```