DAY- 2 Today's Objective:

→ Operators

➔ Control statements

Operator in Java is a symbol that is used to perform operations.

Operator Type	Category	Precedence
Unary	postfix	*expr++ expr--*
	prefix	*++expr --expr +expr -expr ~ !*
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

//Example of Unary Operator:

```
public class OperatorExample{
public static void main(String args[]){
int x=10;
System.out.println(x++);//10 (11)
System.out.println(++x);//12
System.out.println(x--);//12 (11)
System.out.println(--x);//10  }}
```

```
        public class OperatorExample{
        public static void main(String args[]){
        int a=10;
        int b=10;
        System.out.println(a++ + ++a);//10+12=22
        System.out.println(b++ + b++);//10+11=21
//Java Arithmetic Operator Example
        public class OperatorExample{
        public static void main(String args[]){
        int a=10;
        int b=5;
        System.out.println(a+b);//15
        System.out.println(a-b);//5
        System.out.println(a*b);//50
        System.out.println(a/b);//2
        System.out.println(a%b);//0
        }}
                public class OperatorExample{ // BOMAS
                public static void main(String args[]){
                System.out.println(10*10/5+3-1*4/2);
                }}
//Left and Right operators
        public class OperatorExample{
        public static void main(String args[]){
        System.out.println(10<<2);//10*2^2=10*4=40
        System.out.println(10<<3);//10*2^3=10*8=80
        System.out.println(10>>2);//10/2^2=10/4=2
        System.out.println(20>>2);//20/2^2=20/4=5
// Logical AND, OR
    public class OperatorExample{
    public static void main(String args[]){
    int a=10;  int b=5;
    int c=20;  System.out.println(a<b&&a<c);//false && true = false
    System.out.println(a<b&a<c);//false & true = false
    System.out.println(a>b||a<c);//true || true = true
    System.out.println(a>b|a<c);//true | true = true
    }}
```

Relational Operators: Comparison (>, <, >=, <=, Instance of) and Equality (==, !=)

The **java instanceof operator** is used to test whether the object is an instance of the specified type (class or subclass or interface).

The instanceof in java is also known as type *comparison operator* because it compares the instance with type. It returns either true or false.

Example1:

```
class Simple1{
 public static void main(String args[]){
 Simple1 s=new Simple1();
 System.out.println(s instanceof Simple1);//true
```

Example2:

```
class Animal{}
class Dog1 extends Animal{//Dog inherits Animal

 public static void main(String args[]){
 Dog1 d=new Dog1();
 System.out.println(d instanceof Animal);//true
```

Java Control Statements:

Java provides three types of control flow statements.

1. Decision Making statements
    - if statements
    - switch statement
2. Loop statements
    - do while loop
    - while loop
    - for loop
    - for-each loop
3. Jump statements
    - break statement
    - continue statement

the "if" statement is used to evaluate a condition.

1. Simple if statement
2. if-else statement
3. if-else-if ladder
4. Nested if-statement

Syntax:

```
if(condition) { statement 1; //executes when condition is true   }
```

EX: // Simple If

```
public class Student {
public static void main(String[] args) {
int x = 10;
int y = 12;
if(x+y > 20) {
System.out.println("x + y is greater than 20");}}}
```

EX: // If-else

```
public class Student {
public static void main(String[] args) {
int x = 10;
int y = 12;
if(x+y < 10) {
System.out.println("x + y is less than      10");
}   else {
System.out.println("x + y is greater than 20");}}}
```

// Ternary Operator by instand of if-else-if

```
int s1=75, s2=80, s3=35;
String   result=s1>=35&&s2>=35&&s3>=35?"pass":"fail";
System.out.print(result);
```

EX: // If-else ladder

```
public class Student {
public static void main(String[] args) {
String city = "Delhi";
if(city == "Meerut") {
System.out.println("city is meerut");
}else if (city == "Noida") {
```

```
System.out.println("city is noida");
}else if(city == "Agra") {
System.out.println("city is agra");
}else {
System.out.println(city); } } }
```

// Nested If statements

```
public class Student {
public static void main(String[] args) {
String address = "Delhi, India";

if(address.endsWith("India")) {
if(address.contains("Meerut")) {
System.out.println("Your city is Meerut");
}else if(address.contains("Noida")) {
System.out.println("Your city is Noida");    }else {
System.out.println(address.split(",")[0]);    }
}else {    System.out.println("You are not living in India");
```

The switch statement is easier to use instead of if-else-if statements. It also enhances the readability of the program.

The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched.

Syntax : **switch** (expression){

```
case value1:
 statement1;
 break;
.
.
.
case valueN:
 statementN;
 break;
default:
 default statement;  }
```

Rules:

- The case variables can be int, short, byte, char, or enumeration. String type is also supported since version 7 of Java

- Cases cannot be duplicate
- Default statement is executed when any of the case doesn't match the value of expression. It is optional.
- Break statement terminates the switch block when the condition is satisfied. It is optional, if not used, next case is executed.
- While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value.

```
public class BreakExample {  // Break is a keyword break the loop statements
public static void main(String[] args) {
   //using for loop
   for(int i=1;i<=10;i++){
      if(i==5){
         //breaking the loop
         break;
      }
      System.out.println(i);  }}}
```

//switch examples:

```
public class SwitchExample {
public static void main(String[] args) {
   //Declaring a variable for switch expression
   int number=20;
   //Switch expression
   switch(number){
   //Case statements
   case 10: System.out.println("10");
   break;
   case 20: System.out.println("20");
   break;
   case 30: System.out.println("30");
   break;
   //Default case statement
   default:System.out.println("Not in 10, 20 or 30");
   }  }}
```

```java
//where we are printing month name for the given number
        public class SwitchMonthExample {
        public static void main(String[] args) {
            //Specifying month number
            int month=7;
            String monthString="";
            //Switch statement
            switch(month){
            //case statements within the switch block
            case 1: monthString="1 - January";
            break;
            case 2: monthString="2 - February";
            break;
            case 3: monthString="3 - March";
            break;
            case 4: monthString="4 - April";
            break;
            case 5: monthString="5 - May";
            break;
            case 6: monthString="6 - June";
            break;
            case 7: monthString="7 - July";
            break;
            case 8: monthString="8 - August";
            break;
            case 9: monthString="9 - September";
            break;
            case 10: monthString="10 - October";
            break;
            case 11: monthString="11 - November";
            break;
            case 12: monthString="12 - December";
            break;
            default:System.out.println("Invalid Month!");
            }
            //Printing month of the given number
            System.out.println(monthString);  }}
```

Loop Statements

In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true.

loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.

1. for loop
2. while loop
3. do-while loop

for loop: It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code.

We use the for loop only when we exactly know the number of times, we want to execute the block of code.

```
for(initialization, condition, increment/decrement) {
//block of statements
}
```

```
// for loop example
public class Calculattion {
public static void main(String[] args) {
// TODO Auto-generated method stub
int sum = 0;
for(int j = 1; j<=10; j++) {
sum = sum + j;
}
System.out.println("The sum of first 10 natural numbers is " + sum); }}
```

// **for each loop :** Java provides an enhanced for loop to traverse the data structures like array or collection.

```
for(data_type var : array_name/collection_name){
//statements
}
```

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
```

```
String[] names = {"Java","C","C++","Python","JavaScript"};
System.out.println("Printing the content of the array names:\n");
for(String name:names) {
System.out.println(name);
}
```

Java while loop: used to iterate over the number of statements multiple times. However, if we don't know the number of iterations in advance, it is recommended to use a while loop.

```
while(condition){
//looping statements
}
// Example:
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbers \n");
while(i<=10) {
System.out.println(i);
i = i + 2; } }
```

Do-while loop: checks the condition at the end of the loop after executing the loop statements. When the number of iteration is not known and we have to execute the loop at least once, we can use do-while loop.

```
do
{
//statements
} while (condition);
// Example
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbers \n");
do {
System.out.println(i);
i = i + 2;    }while(i<=10);    }}
```

PB-1: In fibonacci series, *next number is the sum of previous two numbers* for example 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 etc. The first two numbers of fibonacci series are 0 and

```
    // Fibonacci Series without using recursion

        class FibonacciExample1{
        public static void main(String args[])  {
         int n1=0,n2=1,n3,i,count=10;
         System.out.print(n1+" "+n2);//printing 0 and 1

         for(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed
          {
         n3=n1+n2;
         System.out.print(" "+n3);
         n1=n2;
         n2=n3;
        }}}
// Fibonacci Series without using recursion

        class FibonacciExample2{
         static int n1=0,n2=1,n3=0;
         static void printFibonacci(int count){
           if(count>0){
                n3 = n1 + n2;
                n1 = n2;
                n2 = n3;
                System.out.print(" "+n3);
                printFibonacci(count-1);      } }
         public static void main(String args[]){
          int count=10;
          System.out.print(n1+" "+n2);//printing 0 and 1
          printFibonacci(count-2);//n-2 because 2 numbers are already printed    } }
//Prime NO
        public class PrimeExample{
         public static void main(String args[]){
          int i,m=0,flag=0;
          int n=3;//it is the number to be checked
          m=n/2;
```

```
        if(n==0||n==1){
         System.out.println(n+" is not prime number");
        }else{
         for(i=2;i<=m;i++){
          if(n%i==0){
           System.out.println(n+" is not prime number");
           flag=1;
           break;
          }        }
         if(flag==0)  { System.out.println(n+" is prime number"); }
        }//end of else  } }
// prime no using method
        public class PrimeExample2{
        static void checkPrime(int n){
         int i,m=0,flag=0;
         m=n/2;
         if(n==0||n==1){
          System.out.println(n+" is not prime number");
         }else{
          for(i=2;i<=m;i++){
           if(n%i==0){
            System.out.println(n+" is not prime number");
            flag=1;
            break;          }        }
          if(flag==0)  { System.out.println(n+" is prime number"); }
         }//end of else   }
        public static void main(String args[]){
         checkPrime(1);
         checkPrime(3);
         checkPrime(17);
         checkPrime(20);  } }
// Palindraom Number or not
        class PalindromeExample{
         public static void main(String args[]){
          int r,sum=0,temp;
          int n=454;//It is the number variable to be checked for palindrome
```

```
        temp=n;
        while(n>0){
         r=n%10;  //getting remainder
         sum=(sum*10)+r;
         n=n/10;      }
        if(temp==sum)
         System.out.println("palindrome number ");
        else
         System.out.println("not palindrome");    } }
```

// Factorial number

```
        class FactorialExample{
         public static void main(String args[]){
         int i,fact=1;
         int number=5;//It is the number to calculate factorial
         for(i=1;i<=number;i++){
            fact=fact*i;      }
         System.out.println("Factorial of "+number+" is: "+fact);
```

Using Recursion

```
class FactorialExample2{
 static int factorial(int n){
  if (n == 0)
    return 1;
  else
    return(n * factorial(n-1));
 }
 public static void main(String args[]){
  int i,fact=1;
  int number=4;//It is the number to calculate factorial
  fact = factorial(number);
  System.out.println("Factorial of "+number+" is: "+fact); } }
```

Armstrong Number

3: 3^1 = **3**

153: $1^3 + 5^3 + 3^3 = 1 + 125 + 27 =$ **153**

125: $1^3 + 2^3 + 5^3 = 1 + 8 + 125 =$ **134 (Not an Armstrong Number)**