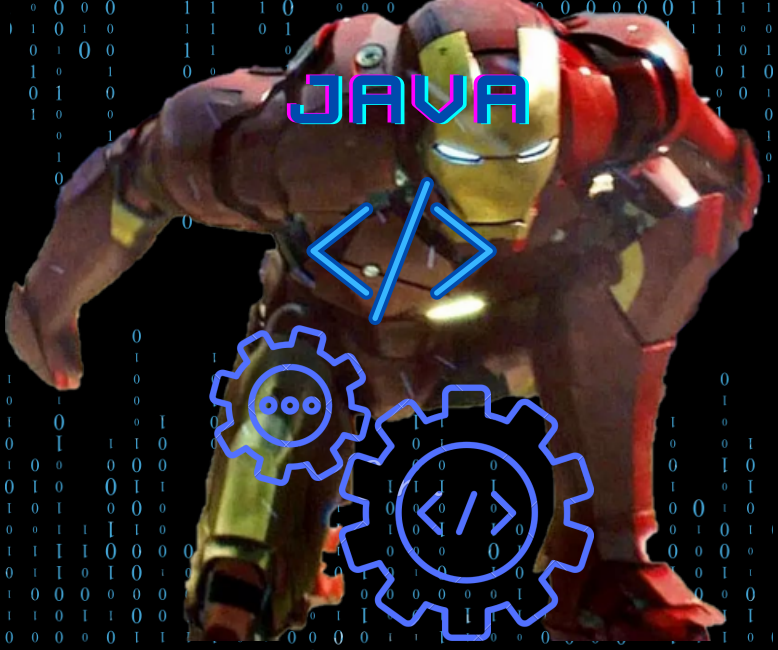
2023

Java

JAVA BASIC

OBSTRON

JAVA



(FIRST PART)

T.I.M. HAMIM

Index:

1. Introduction to Java
2. Variables and Data Types
3. Output in Java
4. Input in Java
5. Operator and expression
6. Control statement
7. Functions/Methods
8. math class
9. static keyword
10. wrapper class
11. file of Java
12. Hamim

13.Hamim

14.Hamim

15.Hamim

16.Hamim

17.Hamim

18.Hamim

19.Hamim

20.Hamim

21.Hamim

22.Hamim

23.Hamim

24.Hamim

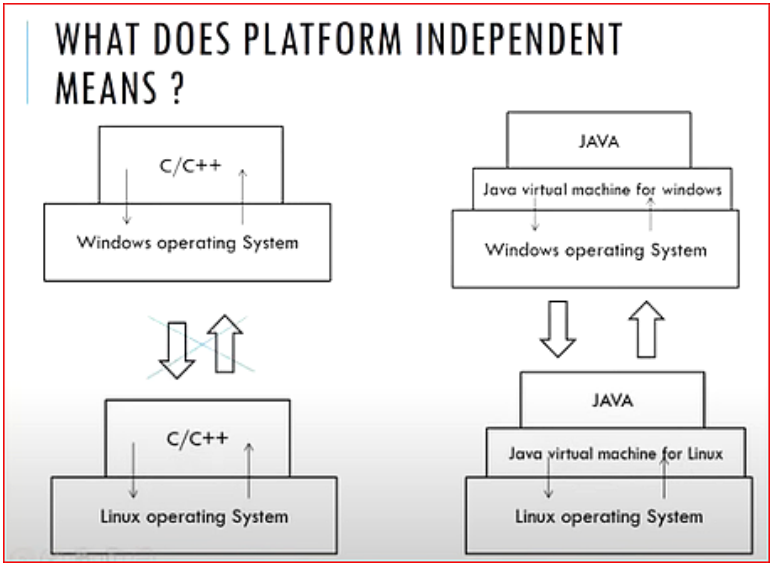
1. Introduction to Java

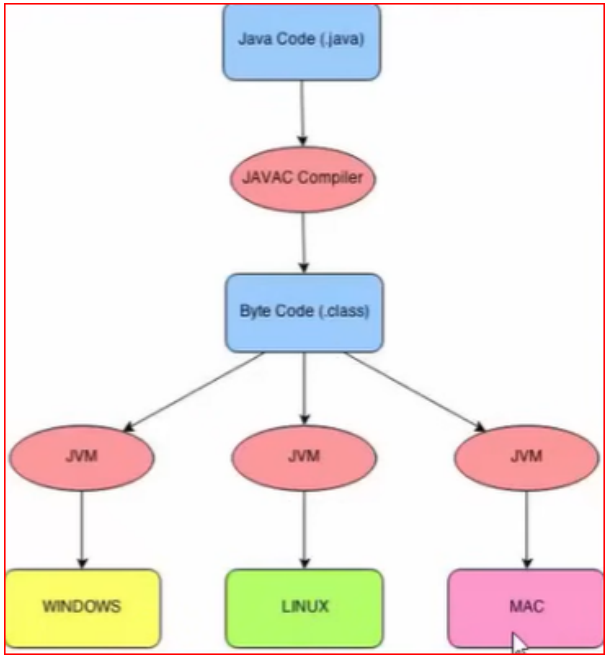
What is Java :

- Java is a high level programming language originally developed by Sun microsystem but currently owned by Oracle.
- Java = Purely object oriented

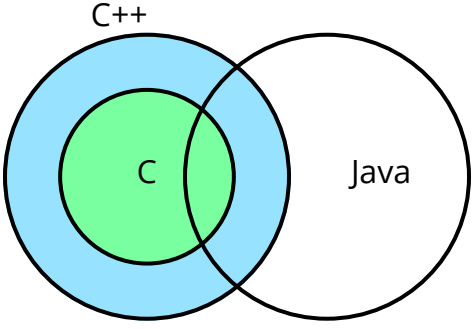
Features of Java :

- Platform independent (WORA)
- Object Oriented
- Support web based application
- Robust
- Secure
- Multi-threaded





Overlap of C, C++ and Java :

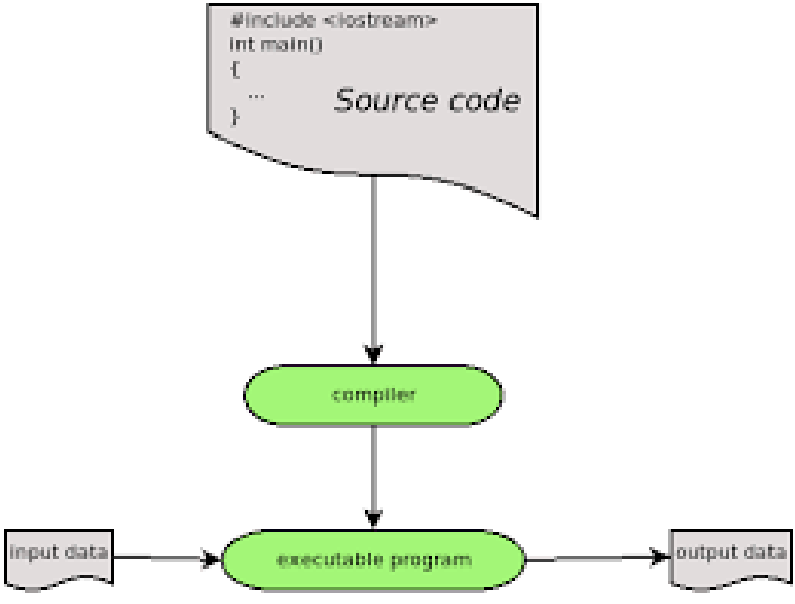


Compiler vs Interpreter :

Compiler and Interpreter are two different ways to translate a program from programming or scripting language to machine language.

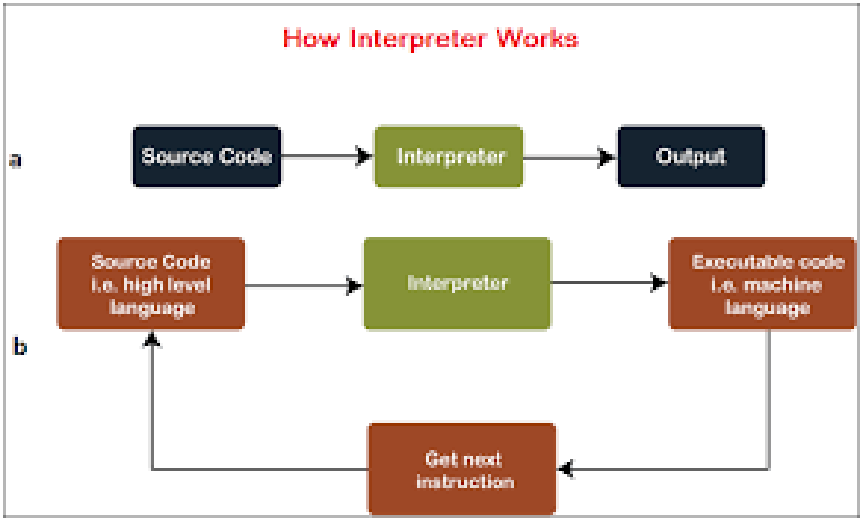
Compiler :

A compiler takes entire program and converts it into object code which is typically stored in a file. The object code is also referred as binary code and can be directly executed by the machine after linking. Examples of compiled programming languages are C and C++.



Interpreter :

An Interpreter directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code. Examples of interpreted languages are Perl, Python and Matlab.

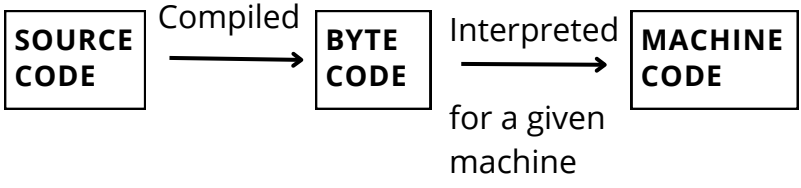


Differences between Interpreter and Compiler:

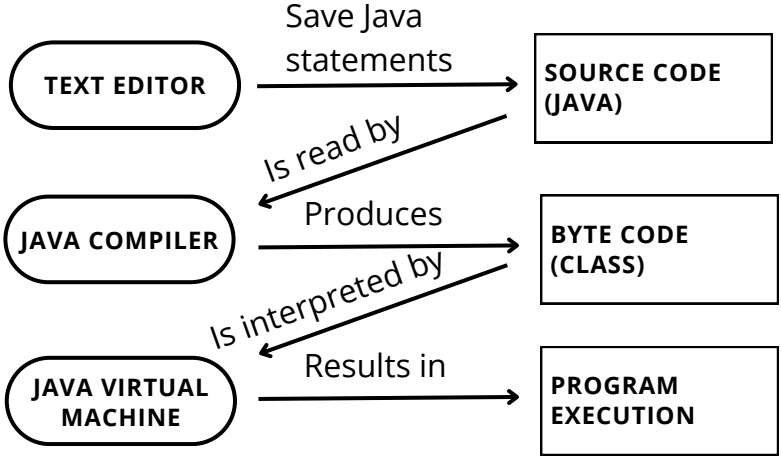
• Interpreter translates just one statement of the program at a time into machine code.	• Compiler scans the entire program and translates the whole of it into machine code at once.
• An interpreter takes very less time to analyze the source code. However, the overall time to execute the process is much slower.	• A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
• An interpreter does not generate an intermediary code. Hence, an interpreter is highly efficient in terms of its memory.	• A compiler always generates an intermediary object code. It will need further linking. Hence more memory is needed.
• Keeps translating the program continuously till the first error is confronted. If any error is spotted, it stops working and hence debugging becomes easy.	• A compiler generates the error message only after it scans the complete program and hence debugging is relatively harder while working with a compiler.
• Interpreters are used by programming languages like Ruby and Python for example.	• Compliers are used by programming languages like C and C++ for example.

How Java works?

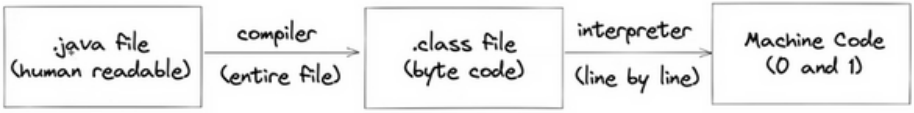
- Java is compiled into the bytecode and then it is interpreted to machine code.



1. Edit
2. Compile
3. Load
4. Verify
5. Execute

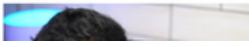


How Java code executes



this is the source code

- this code will not directly run on a system
- we need JVM to run this
- Reason why Java is platform independent

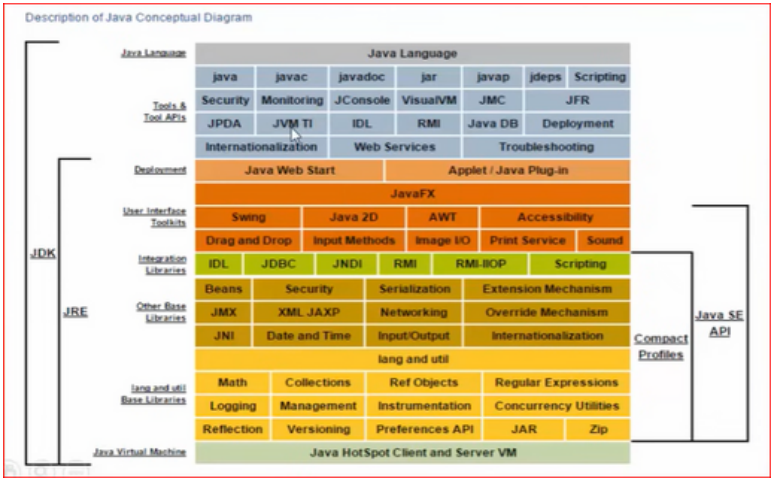


Required Software:

- JDK -> Java Development Kit
- IDE -> Netbenas / Eclipse / JDeveloper

JDK:

- The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

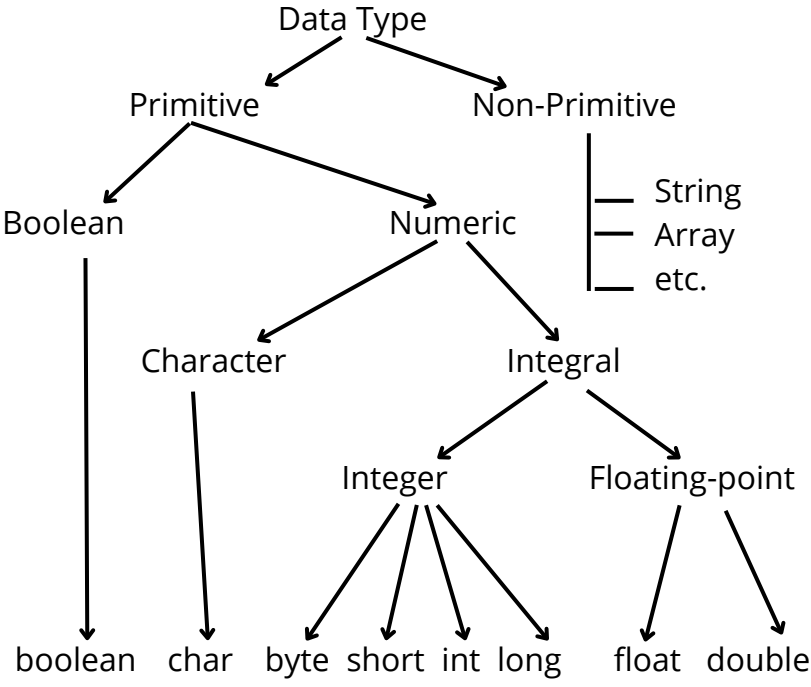


2. Variables and Data Types

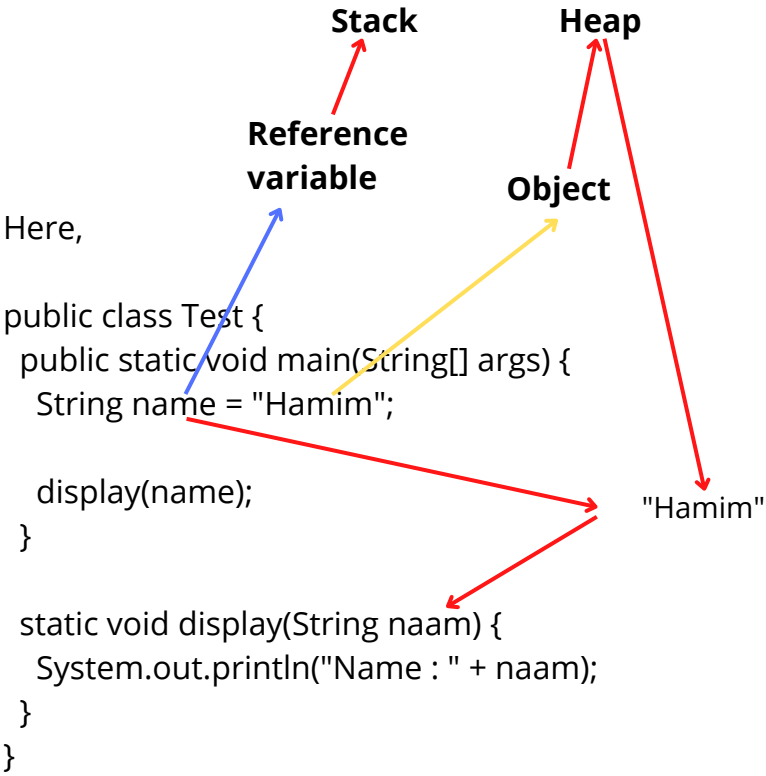
The variable's character can be (A to Z), (a to z), (0 to 9), (_), and ($) sign. And remember one thing that we can't use digit before variables.

Example:
Hamim_Talukder, hamim$, hamim1

Data type in Java:



- Primitive data type variables are stored in the stack memory section.
- Non-Premitive data type variables are stored in the heap memory section.
- Primitive and Non-Primitive data type's variables value, also called object, are stored in the heap memory section.
- All types of variables are called reference variables.



Type Name	Description	Size	Range
boolean	true or false	1 bit	{true, false}
char	Unicode Character	2 bytes	\u0000 to \uFFFF
byte	Signed Integer	1 byte	-128 to 127
short	Signed Integer	2 bytes	-32768 to 32767
int	Signed Integer	4 bytes	-2147483648 to 2147483647
long	Signed Integer	8 bytes	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	4 bytes	±1.4E-45 to ±3.4028235E+38
double	IEEE 754 floating point	8 bytes	±4.9E-324 to ±1.7976931348623157E+308

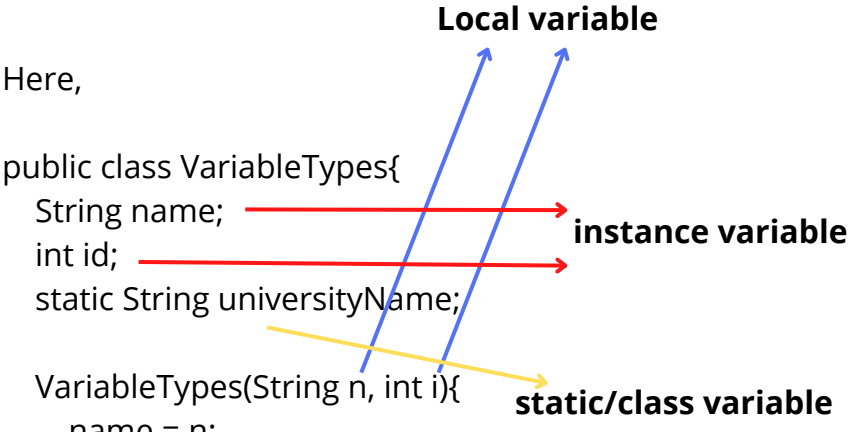
Type Name	Sample Declaration & Initialization
boolean	boolean myBool = true;
char	char myChar = 'a';
byte	int myInt = 100;
short	short myShort = 1000;
int	int myInt = 100000;
long	long myLong = 0;
float	float myFloat = 10.0f;
double	double myDouble = 20.0;

Here,
1 byte = 8 bit,
1 bit = 0 and 1 = 2^1,
2 byte = 16 bit = 2^32,
signed = +value/-value,
unsigned = +value.

Variable types :

There are 3 types of variables in Java :

1. Local
2. instance
3. Class/static



1. **Local variable :**

A variable that is declared inside the method is called local variable.

Local variable is declared inside method, constructor or in a block.

2.**instance variable :**

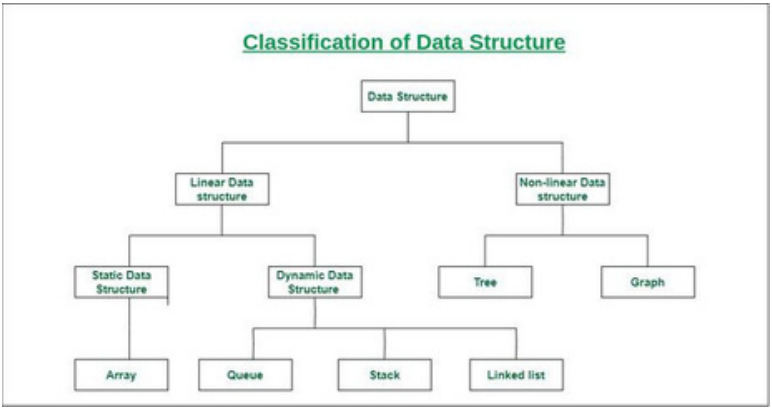
A variable that is declared inside the class but outside any method that is called instance variable.

It is not declared as static.

3.**static/class variable :**

A variable that is declared as static is called static or class variable.

It can't be declared as local variable.


Classification of Data Structure
Data Structure
Linear Data structure
Non-linear Data structure
Static Data Structure
Dynamic Data Structure
Tree
Graph
Array
Queue
Stack
Linked list

keywords in Java:

Java Keyword			
abstract	double	int	strictfp
assert	else	interface	super
boolean	enum	long	switch
break	extends	native	synchronized
byte	final	new	this
case	finally	package	throw
catch	float	private	throws
char	for	protected	transient
class	if	public	try
continue	implements	return	void
default	import	short	volatile
do	instanceof	static	while

Escape Sequence:

It is a specilal character followed by backslash.

Escape Sequence	Meaning
\a	Alarm or Beep
\b	Backspace
\f	Form feed
\n	New line
\t	Tab (Horizontal)
\v	Tab (Vertical)
\r	Carriage return
\'	Single quote
\"	Double quote
****	Backslash
\?	Question Mark
\ooo	Octal number
\xhh	Hexadecimal number

3. Output in Java

- **print():**
- **println():**
- **printf():**

print() :

print() is a method which work is to print/show the output of the program.

- **Make a program that will print your name.**

```
public class java_1
{
   public static void main(String a[])
   {
      System.out.print("Hamim Talukdaar");
   }
}
```

Output :
Hamim Talukdaar

Here,
System is already a created class, and System.out is a standard output object. It is used to display something on screen. All methods/funcions have a first bracket after the method's name. All methods start from main() method.

- **Make a program that will print your name and phone number.**

```
public class java_1
{
    public static void main(String a[])
    {
        System.out.print("Hamim Talukdaar");
        System.out.print("O1731767273");
    }
}
```

Output :

Hamim TalukdaarO1731767273

println() :

We use ln after print method (println) to make a new line. println() is also a method of System class as like print() method.

- **What will be the output of the program given bellow.**

```
public class java_1
{
   public static void main(String a[])
   {
      System.out.println("Hamim Talukdaar");
      System.out.print("O1731767273");
   }
}
```

Output :
Hamim Talukdaar
O1731767273

- **What will be the output of the program given bellow.**

```
public class java_1 {
   public static void main(String a[])
   {
      System.out.println("Hamim Talukdaar");
      System.out.println("O1731767273");
      System.out.print("BSC in CSE");
   }
}
```

Output :
Hamim Talukdaar
O1731767273
BSC in CSE

- **What will be the output of the program given bellow.**

```
public class Hamim_Data
{
   public static void main(String[] args)
   {
      System.out.println("Hamim Talukdaar
\n01731672737 \nBSC in CSE");
   }
}
```

Output :
Hamim Talukdaar
01731672737
BSC in CSE

- **What will be the output of the program given bellow.**

```
public class java_1
{
   public static void main(String[] args)
   {
      System.out.println("1 \t 2");
   }
}
```

Output:

```
1       2
```

- **What will be the output of the program given bellow.**

```
public class java_1
{
   public static void main(String[] args)
   {
      System.out.println("1 \t 2");
      System.out.println("3 \t 4");
   }
}
```

Output:

```
1       2
3       4
```

- **What will be the output of the program given bellow.**

```
public class java_1
{
   public static void main(String[] args)
   {
      System.out.println(" \"Java programming\" ");
   }
}
```

Output:

"Java programming"

- **What will be the output of the program given bellow.**

```
package basicjava;

public class HamimData {
   public static void main(String[] args)
   {
      int number = 12;
      System.out.print(number);
   }
}
```

Output:

12

- **What will be the output of the program given bellow.**

```
package basicjava;

public class HamimData {
   public static void main(String[] args)
   {
      boolean b;
      char c;
      short s;
      int i;
      float f;
      double d;

      b = true;
      System.out.println("b = "+b);
      c = 'a';
      System.out.println("c = "+c);
   }
}
```

Output:

```
b = true
c = a
```

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class HamimData {
   public static void main(String[] args)
   {
      boolean b = true;
      char c = 'a';
      short s;
      int i;
      float f;
      double d;

      System.out.println("b = "+b);
      System.out.println("c = "+c);
   }
}
```

Output:

```
b = true
c = a
```

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    System.out.println('a' + 'b');
    System.out.println("a" + "b");
    System.out.println('a' +3);
    System.out.println((char) ('a' + 3));
    System.out.println("a" + 1);
    System.out.println("Hamim " + new ArrayList<>
());
  }
}
```

Output:

```
195
ab
100
d
a1
Hamim []
```

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      String series = "";
      for(int i=0; i<26; i++){
         char ch = (char) ('a' + i);
         System.out.println(ch);
      }
   }
}
```

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
  public static void main(String[] args) {
    String series = "";
    for(int i=0; i<26; i++){
      char ch = (char) ('a' + i);
      series = series + ch;
    }

    System.out.println(series);
  }
}
```

Output:

abcdefghijklmnopqrstuvwxyz

- **What is the output of the following Java program fragment:**

```
public class Test {
  public static void main(String[] args) {
    StringBuilder builder = new StringBuilder();
    for(int i=0; i<26; i++){
      char ch = (char) ('a' + i);
      builder.append(ch);
    }

    System.out.println(builder);
  }
}
```

Output:
abcdefghijklmnopqrstuvwxyz

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class HamimData {
   public static void main(String[] args)
   {
      boolean b = true;
      char c = 'a';
      short s = 32677;
      int i = 11700;
      float f = 12.50f;
      double d = 132.12;

      System.out.println("b = "+b);
      System.out.println("c = "+c);
      System.out.println("s = "+s);
      System.out.println("i = "+i);
      System.out.println("f = "+f);
      System.out.println("d = "+d);
   }
}
```

Output:

```
b = true
c = a
s = 32677
i = 11700
f = 12.5
d = 132.12
```

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      char[] str1 = "Hamim".toCharArray();
      System.out.println(str1);
   }
}
```

or,

```
public class Test {
   public static void main(String[] args) {
      char str1[] = "Hamim".toCharArray();
      System.out.println(str1);
   }
}
```

Output:
Hamim

printf() :

printf() is a function which work is to print/show the output of the program.

Format specifier in Java:

Format Specifier	Usual Variable Type	Display as
%c	char	single character
%d	int	signed integer
%f	float or double	signed decimal
%e	float or double	exponential format
%b	boolean	
%d	short	
%0	int	unsigned octal value
%u	int	unsigned integer
%x	int	unsigned hex value
%ld	int	long decimal integer
%s	array of char	sequence of characters

We need to use a format specifier when we will use printf() statement.

- **Make a program that will print your name.**

```
public class Test{
   public static void main(String[] args) {
      System.out.printf("I am Hamim");
   }
}
```

Output:
I am Hamim

- **Make a program that will show sum of 50 and 60.**

```
public class Test {
   public static void main(String[] args) {
      int a;
      int b;
      int sum;
      a = 50;
      b = 60;
      sum = a + b;
      System.out.printf("Sum is %d", sum);
   }
}
```

or,

```
public class Test {
   public static void main(String[] args) {
      int a = 50;
      int b = 60;
      int sum;
      sum = a + b;
      System.out.printf("Sum is %d", sum);
   }
}
```

or,

```
public class Test {
   public static void main(String[] args) {
      int a=50,b=60,sum;
      sum = a+b;
      sum = a + b;
      System.out.printf("Sum is %d", sum);
   }
}
```

or,

```
public class Test {
   public static void main(String[] args) {
      int a = 50, b = 60;
      System.out.printf("Sum is %d", a + b);
   }
}
```

or,

```
public class Test {
   public static void main(String[] args) {
      int a = 50, b = 60;
      System.out.printf("%d+%d=%d", a, b, a + b);
   }
}
```

Output:
Sum is 110

- **What will be the output of the program given bellow?**

```
public class Test {
   public static void main(String[] args) {
      int x, y;
      x = 1;
      y = x;
      x = 2;
      System.out.printf("%d", y);
   }
}
```

Output:
1

Here,
The output will be 1. Now think about why it happend ? The reason is y=x is not a equation in program and here we are just assigning the value of y variable.

- **What will be the output of the program given bellow.**

```
public class Test {
  public static void main(String[] args) {
    int a = 277;
    System.out.printf("a = %d\n", a);
    System.out.printf("a = %5d\n", a);
    System.out.printf("a = %05d\n\n", a);

    float b = 277.1f, c = 277f, d = 277.10f, e = 277.0f;
    System.out.printf("b = %f\n", b);
    System.out.printf("b = %09.3f\n", b);
    System.out.printf("b = %9.3f\n", b);
    System.out.printf("b = %9f\n", b);
    System.out.printf("c = %f\n", c);
    System.out.printf("c = %09.3f\n", c);
    System.out.printf("c = %9f\n", c);
    System.out.printf("d = %09.3f\n", d);
    System.out.printf("e = %9f\n", e);
  }
}
```

Output:

```
a = 277
a =   277
a = 00277

b = 277.100006
b = 00277.100
b =   277.100
b = 277.100006
c = 277.000000
c = 00277.000
c = 277.000000
d = 00277.100
e = 277.000000
```

- **Range of data type :**

Type	size	Range	Format
unsigned char	1 byte	0 to 255	%c
signed char or char	1 byte	-128 to +127	%c
unsigned int	2 byte	0 to 65535	%u
signed int or int	2 byte	-32,768 to +32767	%d/%i
unsigned short int	2 byte	0 to 65535	%u
signed short int or short int	2 byte	-32,768 to +32767	%d/%i
unsigned long int	4 byte	0 to +4,294,967,295	
signed long int or long int	4 byte	-2,147,483,648 to +2,147,483,647	
long double	10 byte	3.4E-4932 to 1.1E+4932	%ld
double	8 byte	1.7E-308 to 1.7E+308	%f
float	4 byte	3.4E-38 to 3.4E+38	%f

Here,
1 byte = 8 bit
1 bit = 0 and 1
2 byte = 16 bit = 2^16
4 byte = 32 bit = 2^32
signed = +value/-value
unsigned = +value

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class HamimData {
   public static void main(String[] args)
   {
      boolean b = true;
      char c = 'a';
      short s = 32677;
      int i = 11700;
      float f = 12.50f;
      double d = 132.12;

      System.out.printf("boolean b = %b\n",b);
      System.out.printf("Character c =  %c\n",c);
      System.out.printf("Short s = %d\n",s);
      System.out.printf("Integer i = %d\n",i);
      System.out.printf("Float f = %.2f\n",f);
      System.out.printf("Double d = %.2f\n",d);
   }
}
```

Output:

```
boolean b = true
Character c =  a
Short s = 32677
Integer i = 11700
Float f = 12.50
Double d = 132.12
```

Here,

We used printf() statement that's why we used the format specifier just like we did in C language.

4. Input in Java

Scanner is a class that is used to take input in Java language.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class HamimData
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      int number;

      System.out.println("Enter any number : ");
      number = input.nextInt();
      System.out.println("Number = "+number);
   }
}
```

Output:

```
Enter any number :
120
Number = 120
```

Here,
We must have to import/include Scanner class location (import java.util.Scanner;) after the packegen name. Scanner class is included in util package. In the program, we used Scanner class "input" object/variable to take the input and we used nextInt() for taking integer type value.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class HamimData
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      int number;

      String name = "Hamim Talukdaar";
      System.out.println("Name is : "+name);
   }
}
```

Output:
Name is : Hamim Talukdaar

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class HamimData
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      int number;

      String name;
      System.out.print("Enter your name : ");
      name = input.next();
      System.out.println("My name is "+name);
   }
}
```

Output:

```
Enter your name : Hamim Talukder
My name is Hamim
```

Here,
We can see that next() method can not read strings character after space that's why our program does not print Talukder after Hamim.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class HamimData
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      int number;

      String name;
      System.out.print("Enter your name : ");
      name = input.nextLine();
      System.out.println("My name is "+name);
   }
}
```

Output:

```
Enter your name : Hamim Talukder
My name is Hamim Talukder
```

Here,
We used nextLine() method that's why it printed the whole strings.

- **Make a program where you can enter your name as input after doing this the program will show your name.**

```
package basicjava;
import java.util.Scanner;

public class HamimData {
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      int number;

      String name;
      System.out.print("Enter your name : ");
      name = input.nextLine();
      System.out.printf("My name is %s",name);
   }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class HamimData {
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      int number;
      String name;
      System.out.print("Enter your name : ");
      name = input.nextLine();
      System.out.println("My name is "+name);
   }
}
```

- **Make a program where you can enter a double number as input after doing this the program will print the number.**

```
package basicjava;
import java.util.Scanner;

public class HamimData
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      double num1;
      System.out.print("Enter any double number : ");
      num1 = input.nextDouble();
      System.out.println(num1);
   }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class HamimData
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner (System.in);
      double num1;
      System.out.print("Enter any double number : ");
      num1 = input.nextDouble();
```

```
      System.out.printf("The double number is
%.2f",num1);
   }
}
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class Triangle{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int x = 25;
      int y;
      String  s;
      System.out.print("How many times :");
      y = scan.nextInt();
      s = scan.nextLine();
      System.out.print("Enter your name :");
      s = scan.nextLine();
      for (int i = 1; i <= y; i++) {
         System.out.println("Name : "+s);
      }
   }
}
```

Output:

How many times :3
Enter your name :Hamim Talukder
Name : Hamim Talukder
Name : Hamim Talukder
Name : Hamim Talukder

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      char[] str1 = "Hamim".toCharArray();
      System.out.println(str1);
   }
}
```

or,

```
public class Test {
   public static void main(String[] args) {
      char str1[] = "Hamim".toCharArray();
      System.out.println(str1);
   }
}
```

Output:
Hamim

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      char str1[];
      System.out.print("Enter your name : ");
      str1 = scan.nextLine().toCharArray();
      System.out.println(str1);
   }
}
```

Output:

```
Enter your name : Hamim Talukder
Hamim Talukder
```

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      char str1[];
      System.out.print("Enter your name : ");
      str1 = scan.nextLine().toCharArray();
      System.out.println("Your name is "+ new String(str1));
   }
}
```

Output:

```
Enter your name : Hamim Talukder
Your name is Hamim Talukder
```

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      char[] str1;
      System.out.print("Enter your name: ");
      str1 = scan.nextLine().toCharArray();

      System.out.print("Your name is ");
      for (int i = 0; i < str1.length; i++) {
         System.out.print(str1[i]);
      }
   }
}
```

Output:
Enter your name : Hamim Talukder
Your name is Hamim Talukder

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      char[] str1;
      System.out.print("Enter your name: ");
      str1 = scan.nextLine().toCharArray();

      System.out.print("Your name is ");
      for (char ch : str1) {
         System.out.print(ch);
      }
   }
}
```

Output:
Enter your name : Hamim Talukder
Your name is Hamim Talukder

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    char str1[];
    System.out.print("Enter your name : ");
    str1 = scan.next().toCharArray();
    System.out.println(str1);
  }
}
```

Output:

```
Enter your name : Hamim Talukder
Hamim
```

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        char ch;
        System.out.print("Enter a letter : ");
        ch = scan.next().charAt(0);
        System.out.println("Letter is : "+ch);
    }
}
```

Output:
Enter a letter : h
Letter is : h

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      char str1[], ch;
      System.out.print("Enter your name : ");
      str1 = scan.nextLine().toCharArray();
      System.out.println(str1);
      System.out.print("Enter a letter : ");
      ch = scan.next().charAt(0);
      System.out.println("Letter is : "+ch);
   }
}
```

Output:
Enter your name : Hamim Talukder
Hamim Talukder
Enter a letter : H
Letter is : H

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int num1;
    double num2;
    float num3;
    char ch;
    System.out.print("Enter an integer number : ");
    num1 = input.nextInt();
    System.out.println("Entered integer number is :
"+num1);
    System.out.print("Enter a double number : ");
    num2 = input.nextDouble();
    System.out.println("Entered double number is :
"+num2);
    System.out.print("Enter a float number : ");
    num3 = input.nextFloat();
    System.out.println("Entered float number is :
"+num3);
    System.out.print("Enter a character number : ");
    ch = input.next().charAt(0);
    System.out.println("Entered character is : "+ch);
    input.close();
  }
}
```

Output:

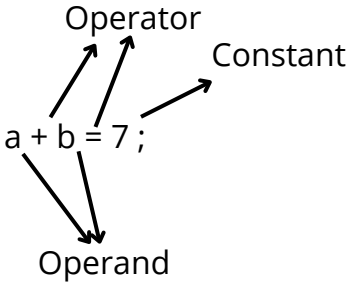
```
Enter an integer number : 12
Entered integer number is : 12
Enter a double number : 7.092
Entered double number is : 7.092
Enter a float number : 3.5
Entered float number is : 3.5
Enter a character number : H
Entered character is : H
```

5. operator and expression

Java provides many types of operators which can be used according to the need. They are classified based on the functionality they provide. Some of the types are:

1. Arithmetic Operators
2. Unary Operators
3. Assignment Operator
4. Relational Operators
5. Logical Operators
6. Ternary Operator
7. Bitwise Operators
8. Shift Operators
9. instance of operator

Example:



Here,
a, and b are Operand,
+ and = are Operator,
7 is a Constant.

- Consider the expression **A + B * 5** , where,

+ , *	are **operators**,
A, B	are **variables**,
5	is **constant**,
A, B and 5	are called **operand**, and
A + B * 5	is an **expression**.

`(a+b) *c`	Operator is `*`, operands are `(a+b)` and `c`
`(a+b)`	Operator is `()`, operand is `a+b`
`a+b`	Operator is `+`, operands are `a` and `b`

Why we are learning these things:

15 * 10 ⟶ Arithmetic operator

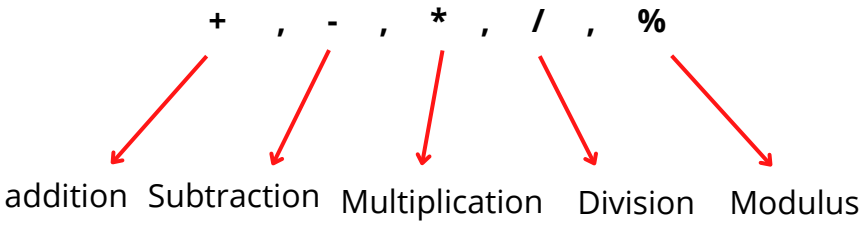
300000000 >= 500000000 Relational operator

if ( && ) Logical operator

Types of operators in C:

NAME OF OPERATORS	OPERATORS
ARITHMETIC OPERATORS	**+ , - , * , / , %**
INCREMENT/DECREMENT OPERATORS	**++ , --**
RELATIONAL OPERATORS	**== , != , <= , >= , < , >**
LOGICAL OPERATORS	**&& , || , !**
BITWISE OPERATORS	**& , ^ , | , ~ , >> , <<**
ASSIGNMENT OPERATORS	**= , += , -= , *= , /= , %= , <<= , >>= , &= , ^= , |=**
OTHER OPERATORS	?: , & , * , SIZEOF()

ARITHMETIC OPERATORS



All are binary operators → means two operands are required to perform the operation.

For example: A + B

Op1 Op2

Operator Precedence and Associativity:

PRECEDENCE ↓	OPERATORS	ASSOCIATIVITY
HIGHEST	+ , - , * , / , %	LEFT TO RIGHT
LOWEST	+ , -	LEFT TO RIGHT

Note: Associativity is used only when two or more operators are of same precedence.

For example: **+ , -**

Same precedence therefore we use associativity

+ , - | left to right

Coding Example:

```
package basicjava;

public class JavaCode{
   public static void main(String[] args)
   {
      int a=2,b=3,c=4,d=5;
      System.out.printf("a*b/c=%d\n",a*b/c);
      System.out.printf("a+b-c=%d\n",a+b-c);
      System.out.printf("a+b*d-c%%a=%d",a+b*d/b-
c%a);
   }
}
```

Output:
a*b/c=1
a+b-c=1
a+b*d-c%a=7

Here:
a+b*d/b-c%a
= a+(b*d)/b-(c%a)
= 2+(3*5)/3-(4%2)
= 2+15/3-0
= 2+5-0
= 7-0
= 7

Increment and Decrement operators:

Increment operator is used to increment the value of a variable by one. Similarly, decrement operator is used to decrement the value of a variable by one.

Increment	Decrement
int a=5	int a=5;
a++;	a-- ;
a=6	a=4

a++; is same as a = a + 1;
a-- ; is same as a = a - 1;

Both are unary operators.
- because they are applied on single operand.

a++; ✓ a ++ a; ✗

Pre-increment operator	post-increment operator
++a;	a++;

Pre-decrement operator	post-decrement operator
--a;	a--;

(a+b)++; error!
++(a+b); error!

error: lvalue required as increment operand

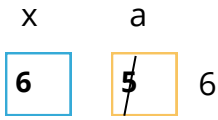
Compiler is expecting a variable as an increment operand but we are providing an expression (a+b) which dose not have the capability to store data.

Question: What is the difference between pre-increment and post-increment operator or pre-decrement and post-decrement operator?

Pre - means first increment/decrement then assign it to another variable.

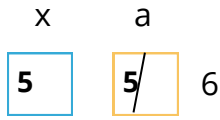
Post - means first assign it to another variable then increment/decrement.

x = ++a;



x = 6

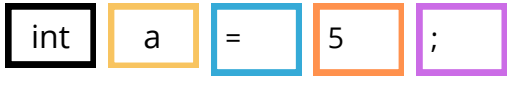
x = a++;



x = 5

Token Generation:

- Lexical analysis is the first phase in the compilation process.
- Lexical analyzer(scanner) scans the whole source program and when it finds the meaningful sequence of characters(lexemes) then it converts it into a token.
- Token: Lexemes mapped into token-name and attribute-value.
  Example: int → <keyword, int>
- It always matches the longest character sequence.

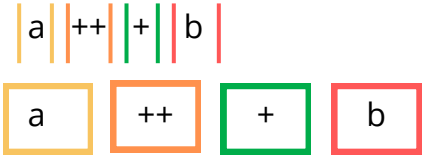


- **What is the output of the following Java program fragment:**

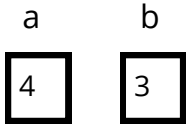
```
package basicjava;
public class JavaCode{
   public static void main(String[] args)
   {
      int a=4,b=3;
      System.out.printf("%d\n",a+++b);
   }
}
```

Output:
7

Here,



a++ + b
= 4 + 3
= 7



Post-increment/ decrement in context of equation:
First use the value in the equation and then increment the value.

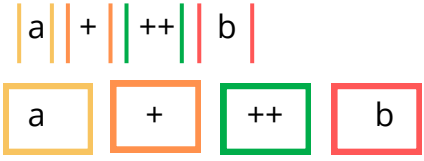
Pre-increment/ decrement in context of equation:
First increment the value and then use in the equation after completion of the equation.

- **What is the output of the following Java program fragment:**

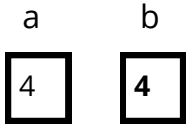
```
package basicjava;
public class JavaCode{
   public static void main(String[] args)
   {
      int a=4,b=3;
      System.out.printf("%d\n",a + ++b);
   }
}
```

Output:
8

Here,



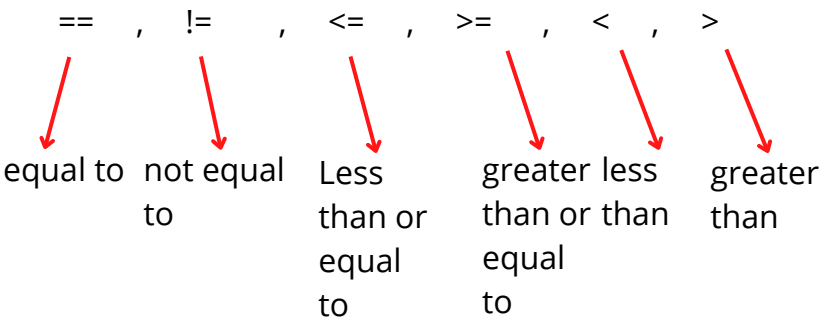
a+ ++ b
= 4 + 4
= 8



Post-increment/ decrement in context of equation:
First use the value in the equation and then increment the value.

Pre-increment/ decrement in context of equation:
First increment the value and then use in the equation after completion of the equation.

Relation operation:



Used for comparing two values

- all relational operators will return either true or False.

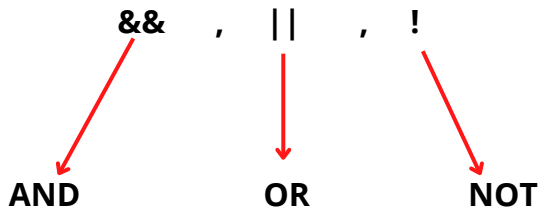
4==5 is equivalent to is 4 == 5 ?
Answer: False
4!=5 is equivalent to is 4!==5 ?
Answer: True

- **What is the output of the following C program fragment:**

```
package basicjava;

public class JavaCode {

    public static void main(String[] args) {
        int a = 3, b = 4;
        if (b >= a) {
            System.out.printf("Bingo! You are in");
        } else {
            System.out.printf("OOPS! You are out");
        }
    }
}
```

Output:
Bingo! You are in

Logical operators:



- && and || are used to combine two conditions. && - returns TRUE when all the conditions under consideration are true and returns FALSE when any one or more than one condition is false.

For example:

- **What is the output of the following C program fragment:**

```
package basicjava;
public class JavaCode {

    public static void main(String[] args) {
        int a = 5;
        if (a == 5 && a != 6 && a <= 56 && a > 4) {
            System.out.printf("Welcome to this beautiful
world of          operators");
        }
    }
}
```

Output:
Welcome to this beautiful world of operators

- || - returns TRUE when one or more than one condition under consideration is true and returns FALSE when all conditions are false.

For example:

- **What is the output of the following Java program fragment:**

```
package basicjava;
public class JavaCode {

    public static void main(String[] args) {
        int a = 5;
        if (a == 5 || a != 6 || a <= 56 || a > 4) {
            System.out.printf("Welcome to this beautiful
world of operators");
        }
    }
}
```

Output:
Welcome to this beautiful world of operators

- ! operator is used to complement the condition under consideration.
  ! - returns TRUE when condition is FALSE and returns FALSE and False when condition is TRUE.

For example:

- **What is the output of the following Java program fragment:**

```
package basicjava;
public class JavaCode {

    public static void main(String[] args) {
        int a = 5;
        if (a == 5 || a != 6 || a <= 56 || a > 4) {
            System.out.printf("Welcome to this beautiful
world of operators");
        }
    }
}
```

Output:
Welcome to this beautiful world of operators

Concept of short circuit in logical operators:

Short circuit in case of &&: Simply means if there is a condition anywhere in the expression that returns false, then the rest of the conditions after that will not be evaluated.
For example:

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class JavaCode {

   public static void main(String[] args) {
      int a = 5, b = 3;
      boolean incr;
      incr = (a < b) && (b++ > 0);
      System.out.printf("%b\n", incr);
      System.out.printf("%d\n", b);
   }
}
```

Output:
false
3

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class JavaCode {

   public static void main(String[] args) {
      int a = 5, b = 3;
      boolean incr;
      incr = (a > b) && (b++>0);
      System.out.printf("%b\n", incr);
      System.out.printf("%d\n", b);
   }
}
```

Output:
true
4

Short circuit in case of ||: Simply means if there is a condition anywhere in the expression that returns True, then the rest of the conditions after that will not be evaluated.
For example:

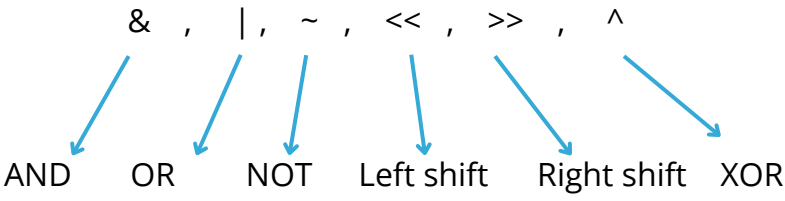
- **What is the output of the following Java program fragment:**

```
package basicjava;

public class JavaCode {

    public static void main(String[] args) {
        int a = 5, b = 3;
        boolean incr;
        incr = (a > b) || (b++ > 0);
        System.out.printf("%b\n", incr);
        System.out.printf("%d\n", b);
    }
}
```

Output:
true
3

Bitwise operators:

As name suggests - it dose bitwise manipulation.



BIT BY BIT

Bitwise AND (&) operator:

- It takes two bits at a time and perform AND operation.
- And (&) is binary operator. It takes two numbers and perform bitwise AND.
- Result of AND is 1 when both bits are 1.

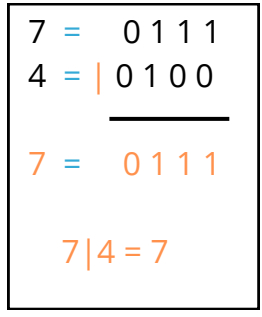
```
7 =    0 1 1 1
4 = &  0 1 0 0
-----------------
4 =    0 1 0 0

   7&4 = 4
```

Truth Table

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise OR (|) operator:

- It takes two bits at a time and perform OR operation.
- OR (|) is binary operator. It takes two numbers and perform bitwise OR.
- Result of OR is 0 when both bits are 0.

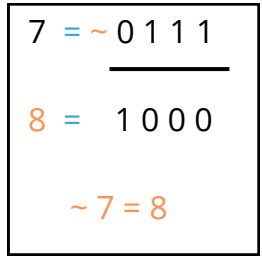


Truth Table

A	B	A&B
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise NOT (~) operator:

- NOT is a unary operator.
- It's job is to complement each bit one by one.
- Result of NOT is 0 when bit is 1 and 1 when bit is 0.



Truth Table

A	~A
0	1
1	0

Difference between bitwise and logical operators:

```
package basicjava;

public class JavaCode {

    public static void main(String[] args) {
        char x = 1, y = 2;
        /* x=1(0000 0001),
           y=2(0000 0010)
           1&2=0(0000 0000)
  1&&2 = TRUE && TRUE = TRUE =1 */
        if (0<(x & y)) {
            System.out.printf("Result of x&y is 1\n");
        }
        if (x>0 && y>0) {
            System.out.printf("Result of x&&y is 1\n");
        }
    }
}
```

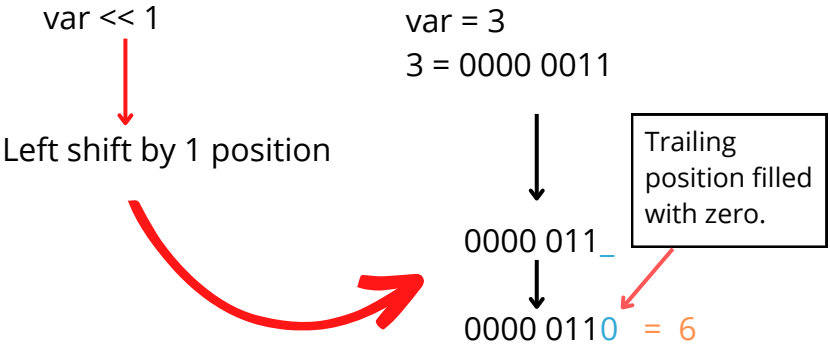
output:
Result of x&&y is 1

Left shift operator:

First operand << Second operand

Whose bits get left shifted

- How left shift works?



Important points:

Left shifting is equivalent to multiplication by 2^(right Operator) .

Example: var = 3;

var << 1 Output: 6 [3 x 2^1]

var << 4 Output: 48 [3 x 2^4]

Important points:

- When bits are shifted left then trailing positions are filled with zeros.

char = 1
byte = 8 bits

```
package basicjava;

public class JavaCode {

    public static void main(String[] args) {
        char var = 3; // 3 in binary = 0000
0011
        System.out.printf("%d\n", var << 1);
    }
}
```

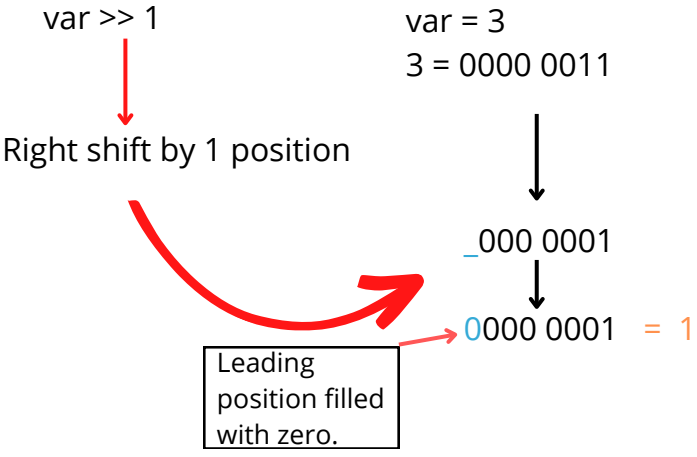
output:
6

Right shift operator:

First operand >> Second operand

Whose bits get right shifted

- How right shift works?



Important points:

Right shifting is equivalent to division by 2^(right Operator) .

Example:

var = 3;
var >> 1 Output: 1 [3 / 2^1]

var = 32
var >> 4 Output: 2 [32 / 2^4]

Important points:

- When bits are shifted right then leading positions are filled with zeros.

```
package basicjava;

public class JavaCode {

   public static void main(String[] args) {
      char var = 3; // 3 in binary = 0000
0011
      System.out.printf("%d\n", var >> 1);
   }
}
```

output:
1

Bitwise XOR operator:

X - OR

Inclusive OR:

- Either A is 1 or B is 1 or Both are 1, then the output is 1.
- Including BOTH.

Exclusive OR:

- Either A is 1 or B is 1 then the output is 1 but when both A and B are 1 then Output is 0.
- Excluding BOTH

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Only difference

- Bitwise XOR(^) is binary operator. It takes two numbers and perform bitwise XOR.
- Result of XOR is 1 when two bits are different otherwise the result is 0.

```
7 =    0 1 1 1
4 = ^  0 1 0 0
       -------
7 =    0 0 1 1

7^4 = 3
```

Truth Table

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

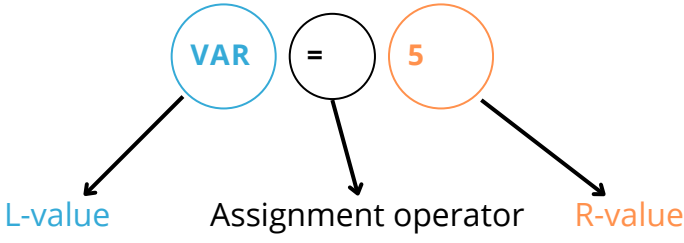
- **What is the output of the following C program fragment:**

```
package basicjava;
public class JavaCode {
   public static void main(String[] args) {
      int a = 4, b = 3;
      a = a ^ b;
      b = a ^ b;
      a = a ^ b;
      System.out.printf("After XOR, a=%d and
b=%d\n", a, b);
   }
}
```

Output:
After XOR, a=3 and b=4

Assignment operator:

Values to a variable can be assigned using assignment operator.
Requires two values - L-value and R-value.
This operator copies R-value to L-value.



Shorthand assignment operators:

+=	First addition than assignment
-=	First subtraction than assignment
*=	First multiplication than assignment
/=	First division than assignment

Example: a += 1 is equivalent to a = a +1

Similar concept for other shorthand assignment operators as well.

- **What is the output of the following Java program fragment:**

```
package basicjava;
public class JavaCode {
   public static void main(String[] args) {
      int a = 7;
      a ^= 5;
      System.out.printf("%d", a += 3);
   }
}
```

Output:

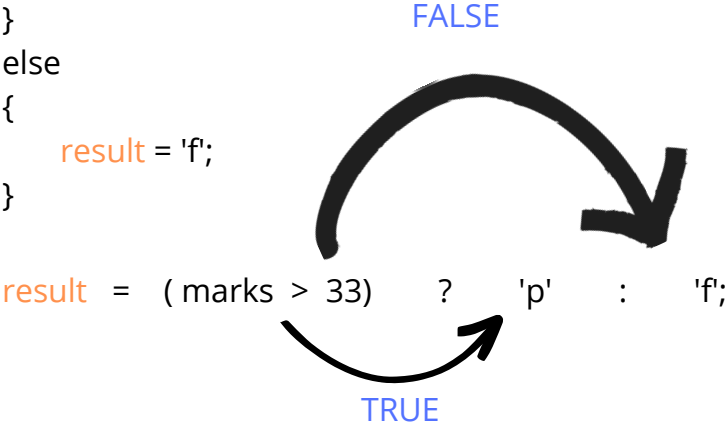
5

Conditional operator:

Look and feel: ? :

```
char result;
int marks;
if(marks > 33)
{
      result = 'p';
}
else
{
     result = 'f';
}
```

```
char result;
int marks;
result = (marks > 33) ? 'p' :  'f';
```



(marks > 33) is a boolean expression, therefore it will return either true or false.

(marks > 33) ? 'p ' : 'f' is a conditional expression, which is after all expression, therefore it is an r-value and result is l-value.

Ternary operator: Ternary operator is a shorthand version of the if-else statement. It has three operands and hence the name ternary.

The general format is:

condition **?** if true **:** if false

Syntax:

```
int x = 10;
in t y =5;

int large = (x>y) ? x : y;
```

The above statement means that if the condition evaluates to true, then execute the statements after the '?' else execute the statements after the ':'.
Example:

```
public class operators {
    public static void main(String[] args)
    {
        int a = 20, b = 10, c = 30, result;

        // result holds max of three
        // numbers
        result
            = ((a > b) ? (a > c) ? a : c : (b > c) ? b : c);
        System.out.println("Max of three numbers = "
                           + result);
    }
}
```

Output:
Max of three numbers = 30

- **What will be the output of the program given bellow.**

```
public class Test {
   public static void main(String[] args) {
      int var, y = 7;
      if (y < 10) {
         var = 30;
      } else {
         var = 40;
      }
      System.out.println(var);
   }
}
```

Here,
The output will be 30.
Now we will do this program using ternary operator.

or,

```
public class Test {
   public static void main(String[] args) {
      int var, y = 7;
      var = (y < 10) ? 30 : 40;
      System.out.println(var);
   }
}
```

Quick facts checklist:

- Conditional operator is the only ternary operator available in the list of operators in Java language.
- As in Expression1 ? Expression2 : Expression3 , expression1 is the boolean expression. If we simply write 0 instead of some boolean expression than that simply means FALSE and therefore Expression3 will getevaluated.

Example:

```
int result;
result = 0 ? 2 : 1
```

result

Instanceof operator: The instance of the operator is used for type checking. It can be used to test if an object is an instance of a class, a subclass, or an interface. General format-

object instance of class/subclass/interface

Example:

```
// Java program to illustrate
// instance of operator
class operators {
    public static void main(String[] args)
    {

        Person obj1 = new Person();
        Person obj2 = new Boy();

        // As obj is of type person, it is not an
        // instance of Boy or interface
        System.out.println("obj1 instanceof Person: "
                           + (obj1 instanceof Person));
 System.out.println("obj1 instanceof Boy: "
 + (obj1 instanceof Boy));
 System.out.println("obj1 instanceof MyInterface: "
 + (obj1 instanceof MyInterface));
```

```
        // Since obj2 is of type boy,
        // whose parent class is person
        // and it implements the interface Myinterface
        // it is instance of all of these classes
        System.out.println("obj2 instanceof Person: "
                        + (obj2 instanceof Person));
        System.out.println("obj2 instanceof Boy: "
                        + (obj2 instanceof Boy));
        System.out.println("obj2 instanceof
MyInterface: "
                        + (obj2 instanceof MyInterface));
    }
}

class Person {
}

class Boy extends Person implements MyInterface {
}

interface MyInterface {
}
```

Output

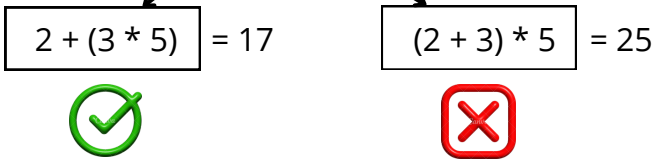
```
obj1 instanceof Person: true
obj1 instanceof Boy: false
obj1 instanceof MyInterface: false
obj2 instanceof Person: true
obj2 instanceof Boy: true
obj2 instanceof MyInterface: true
```

Precedence and Associativity of Operators::

Precedence of operators come into picture when in an expression we need to decide which operator will be evaluated first. Operator with higher precedence will be evaluated first.

Example: 2 + 3 * 5

Precedence of multiplication is greater than addition.



Associativity of operators come into picture when precedence of operators are some need to decide which operator will be evaluated first.

Example: 10 / 2 * 5
Left to right: (10 / 2) * 5 = 25
Right to left: 10 / (2 * 5) = 1

Associativity can be either :

1. Left to right
2. Right to left

Precedence and associative rules are used when dealing with hybrid equations involving more than one type of operator. In such cases, these rules determine which part of the equation to consider first, as there can be many different valuations for the same equation. The below table depicts the precedence of operators in decreasing order as magnitude, with the top representing the highest precedence and the bottom showing the lowest precedence.

Operators	Associativity	Type
++ --	Right to left	Unary postfix
++ -- + - ! (type)	Right to left	Unary prefix
/ * %	Left to right	Multiplicative
+ -	Left to right	Additive
< <= > >=	Left to right	Relational
== !==	Left to right	Equality
&	Left to right	Boolean Logical AND
^	Left to right	Boolean Logical Exclusive OR
|	Left to right	Boolean Logical Inclusive OR
&&	Left to right	Conditional AND
||	Left to right	Conditional OR
?:	Right to left	Conditional
= += -= *= /= %=	Right to left	Assignment

- The following table lists the precedence and associativity of Java operators.

Operators are listed top to bottom, in descending precedence.

Precedence	Operator	Description	Associativity
1	`( )`	Function call	
	`[ ]`	Array subscripting	
	`.`	Structure and union member access	
	`->`	Structure and union member access through pointer	
2	`++ --`	Prefix increment and decrement	Right-to-left
	`+ -`	Unary plus and minus	
	`! ~`	Logical NOT and bitwise NOT	
	`(type)`	Type cast	
	`*`	Indirection (dereference)	
	`&`	Address-of	
	`sizeof`	Size-of	
3	`* / %`	Multiplication, division, and remainder	Left-to-right
4	`+ -`	Addition and subtraction	
5	`<< >>`	Bitwise left shift and right shift	
6	`< <=`	For relational operators < and ≤ respectively	
	`> >=`	For relational operators > and ≥ respectively	
7	`== !=`	For relational = and ≠ respectively	
8	`&`	Bitwise AND	
9	`^`	Bitwise XOR (exclusive or)	
10	`|`	Bitwise OR (inclusive or)	
11	`&&`	Logical AND	
12	`||`	Logical OR	
13	`?:`	Ternary conditional	Right-to-Left
14	`=`	Simple assignment	
	`+= -=`	Assignment by sum and difference	
	`*= /= %=`	Assignment by product, quotient, and remainder	
	`<<= >>=`	Assignment by bitwise left shift and right shift	
	`&= ^= |=`	Assignment by bitwise AND, XOR, and OR	
15	`,`	Comma	Left-to-right

Interesting Questions about Operators :

1. Precedence and Associativity: There is often confusion when it comes to hybrid equations which are equations having multiple operators. The problem is which part to solve first. There is a golden rule to follow in these situations. If the operators have different precedence, solve the higher precedence first. If they have the same precedence, solve according to associativity, that is, either from right to left or from left to right. The explanation of the below program is well written in comments within the program itself.
Example:

```
public class operators {
   public static void main(String[] args)
   {
      int a = 20, b = 10, c = 0, d = 20, e = 40, f = 30;

      // precedence rules for arithmetic operators.
      // (* = / = %) > (+ = -)
      // prints a+(b/d)
      System.out.println("a+b/d = " + (a + b / d));

      // if same precedence then associative
      // rules are followed.
      // e/f -> b*d -> a+(b*d) -> a+(b*d)-(e/f)
      System.out.println("a+b*d-e/f = "
                         + (a + b * d - e / f));
   }
}
```

Output
a+b/d = 20
a+b*d-e/f = 219

2. Be a Compiler: Compiler in our systems uses a lex tool to match the greatest match when generating tokens. This creates a bit of a problem if overlooked. For example, consider the statement a=b+++c; too many of the readers might seem to create a compiler error. But this statement is absolutely correct as the token created by lex are a, =, b, ++, +, c. Therefore, this statement has a similar effect of first assigning b+c to a and then incrementing b. Similarly, a=b+++++c; would generate an error as tokens generated are a, =, b, ++, ++, +, c. which is actually an error as there is no operand after the second unary operand.
Example:

```
public class operators {
   public static void main(String[] args)
   {
      int a = 20, b = 10, c = 0;

      // a=b+++c is compiled as
      // b++ +c
      // a=b+c then b=b+1
      a = b++ + c;
      System.out.println("Value of a(b+c), "
                  + " b(b+1), c = " + a + ", " + b
                  + ", " + c);
```

```
        // a=b+++++c is compiled as
        // b++ ++ +c
        // which gives error.
        // a=b+++++c;
        // System.out.println(b+++++c);
    }
}
```

Output
Value of a(b+c), b(b+1), c = 10, 11, 0

3. Using + over (): When using + operator inside system.out.println() make sure to do addition using parenthesis. If we write something before doing addition, then string addition takes place, that is, associativity of addition is left to right, and hence integers are added to a string first producing a string, and string objects concatenate when using +. Therefore it can create unwanted results.
Example:

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

        int x = 5, y = 8;

        // concatenates x and y as
        // first x is added to "concatenation (x+y) = "
        // producing "concatenation (x+y) = 5"
        // and then 8 is further concatenated.
        System.out.println("Concatenation (x+y)= " + x + y);
```

```
        // addition of x and y
         System.out.println("Addition (x+y) = " + (x + y));
    }
}
```

Output

```
Concatenation (x+y)= 58
Addition (x+y) = 13
```

- **Make a program that will show the sum of 20 and 10.**

```
package basicjava;

public class ArithmeticDemo {
   public static void main(String[] args)
   {
      int num1, num2, result;
      num1 = 20;
      num2 = 10;
      result = num1 + num2;
      System.out.println("Sum = "+result);
   }
}
```

or,

```
package basicjava;

public class JavaCode {
   public static void main(String[] args)
   {
      int num1, num2, result;
      num1 = 20;
      num2 = 10;
      System.out.println("Sum = "+(num1+num2));
   }
}
```

Output:
Sum = 30

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class ArithmeticDemo {
   public static void main(String[] args)
   {
      int num1, num2, result;
      num1 = 20;
      num2 = 10;
      result = num1 + num2;
      System.out.println("Sum = "+result);
      result = num1 - num2;
     System.out.println("Sub = "+result);
     result = num1 * num2;
     System.out.println("Multi = "+result);
     result = num1 / num2;
     System.out.println("Divi = "+result);
     result = num1 % num2;
    System.out.println("Reminder = "+result);
  }
}
```

Output:
Sum = 30
Sub = 10
Multi = 200
Divi = 2
Reminder = 0

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class ArithmeticDemo {
   public static void main(String[] args)
   {
      Scanner inputan = new Scanner(System.in);
      int num1, num2, result;
      System.out.print("Enter first number : ");
      num1 = inputan.nextInt();
      System.out.print("Enter second number : ");
      num2 = inputan.nextInt();
      result = num1 + num2;
      System.out.println("Sum = "+result);
      result = num1 - num2;
      System.out.println("Sub = "+result);
      result = num1 * num2;
      System.out.println("Multi = "+result);
      result = num1 / num2;
      System.out.println("Div = "+result);
      result = num1 % num2;
      System.out.println("Reminder = "+result);
   }
}
```

Output:
Enter first number : 20
Enter second number : 10

Sum = 30
Sub = 10
Multi = 200
Div = 2
Reminder = 0

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class HamimData{
   public static void main(String[] args)
   {
      int x = 3;
      int y = 2;
      x+=y; // x = x + y = 5
      System.out.println("x = "+x);
   }
}
```

Output:
x = 5

- **What is the output of the following Java program fragment:**

```
package basicjava;

public class HamimData{
   public static void main(String[] args)
   {
      int x = 3;
      int y = 2;
      x+=y; // x = x + y = 5
      System.out.println("x = "+x);
      x-=y; // x = x - y = 3
      System.out.println("x = "+x);
      x*=y; // x = x * y = 6
      System.out.println("x = "+x);
      x/=y;// = x / y = 3
      System.out.println("x = "+x);
      x%=y;//x = x % y = 1
      System.out.println("x = "+x);
   }
}
```

Output:
x = 5
x = 3
x = 6
x = 3
x = 1

- **Make a program that will take the height and width of a triangle and show its area.**

```
package SecondPackage;
import java.util.Scanner;

public class Halum {
   public static void main(String[] args) {
      Scanner scanf = new Scanner(System.in);
      System.out.printf("Enter height : ");
      double height = scanf.nextDouble();
      System.out.printf("Enter base : ");
      double base = scanf.nextDouble();
      System.out.println("Area of triangle :
"+.5*height*base);
      scanf.close();
   }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class Triangle{
   public static void main(String[] args)
   {
      double base, height, area;
      Scanner input = new Scanner(System.in);
      System.out.print("Enter the height : ");
      height = input.nextDouble();
      System.out.print("Enter the base : ");
```

```
        base = input.nextDouble();
        area = .5*height*base;
        System.out.printf("The area of triangle :
%.2f",area);
    }
}
```

Output:
Enter the height : 6
Enter the base : 3
The area of triangle : 9.00

- **Make a program that will take the radius of a circle and show its area.**

```
package basicjava;
import java.util.Scanner;

public class Triangle{
    public static void main(String[] args)
    {
        double radius, area;
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the radius : ");
        radius = input.nextDouble();
        area = 3.1416*radius*radius;
        System.out.printf("The area of circle :
%.2f",area);
    }
}
```

Output:
Enter the radius : 3
The area of circle : 28.27

- **Make a program that will take the temperature in celsius and convert it into Fahrenheit.**

```
package basicjava;
import java.util.Scanner;

public class Triangle{
   public static void main(String[] args)
   {
      double celsius, fahrenheit;
      Scanner input = new Scanner(System.in);
      System.out.print("Enter the tempreture in
celsius : ");
      celsius = input.nextDouble();
      fahrenheit = 1.8*celsius+32;
      System.out.println("Fahrenheit : "+fahrenheit);
   }
}
```

```
Enter the tempreture in celsius : 15
Fahrenheit : 59.0
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class Triangle{
   public static void main(String[] args)
   {
      int x = -10;
      int result;
      result = +x; // Unary + operator
      System.out.println("result = "+result);
      result = -x; // Unary - operator
      System.out.println("result = "+result);
      result = ++x; // Unary ++ operator
      System.out.println("result = "+result);
      result = --x; // Unary -- operator
      System.out.println("result = "+result);
   }
}
```

Output:
result = -10
result = 10
result = -9
result = -10

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class Triangle{
   public static void main(String[] args)
   {
      int x = 25;
      int y;
      y = x++; // post increment
      System.out.println("y = "+y);
      y = x;
      System.out.println("y = "+y);
   }
}
```

Output:
y = 25
y = 26

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class Triangle{
   public static void main(String[] args)
   {
      int x = 25;
      int y;
```

```
        y = ++x; // pre increment
        System.out.println("y = "+y);
        y = x;
        System.out.println("y = "+y);
    }
}
```

Output;
y = 26
y = 26

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class Triangle{
    public static void main(String[] args)
    {
        int x = 25;
        int y;
        y = --x; // pre increment
        System.out.println("y = "+y);
        y = x--;
        System.out.println("y = "+y);
        y = x;
        System.out.println("y = "+y);
        y = ++x;
        System.out.println("y = "+y);
```

```
        y = x++;
        System.out.println("y = "+y);
    }
}
```

Output:
y = 24
y = 24
y = 23
y = 24
y = 24

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
    public static void main(String[] args)
    {
        Scanner inputt = new Scanner(System.in);
        int num1, num2, large;
        System.out.print("Enter 2 numbers : ");
        num1 = inputt.nextInt();
        num2 = inputt.nextInt();
        large = (num1>num2) ? num1 : num2;
        System.out.println("Large number = "+large);
    }
}
```

Output:
Enter 2 numbers : 5 6
Large number = 6

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int a = 32;
      int b = 12;
      int c;
      c = a & b;
      System.out.println("a & b = "+c);
      c = a | b;
      System.out.println("a | b = "+c);
      c = a ^ b;
      System.out.println("a ^ b = "+c);
   }
}
```

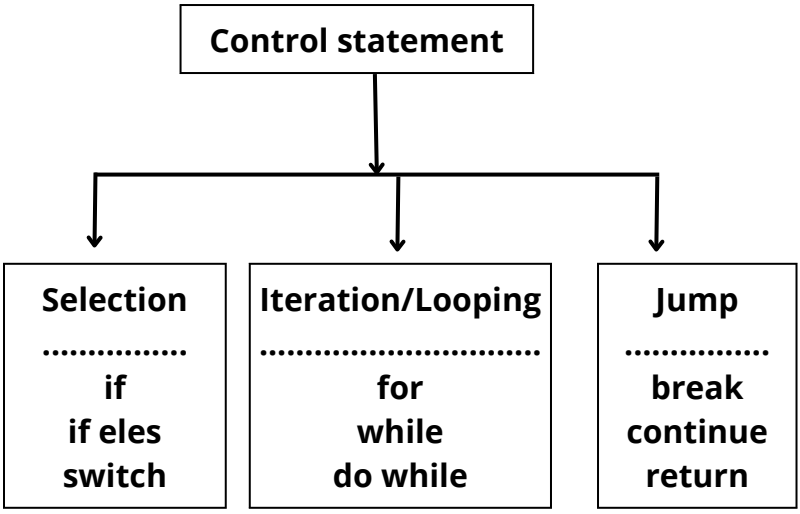
Output:
a & b = 0
a | b = 44
a ^ b = 44

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int a = 32;
      int c;
      c = a >> 3;
      System.out.println("a >> 3 = "+c);
      c = a << 3;
      System.out.println("a << 3 = "+c);
      c = a >> 2;
      System.out.println("a >> 2 = "+c);
   }
}
```

Output:
a >> 3 = 4
a << 3 = 256
a >> 2 = 8

6. Control Statement



What is statement ?

Any meaningful expression is called a statement.

Example:
x = 10; -> It's a meaningful expression.

The statement of this expression can be :
System.out.println("Positive");
System.out.println("Negative");
System.out.println("Zero");
But only one statement can be true for this expression. In this case, we need a control statement to determine it.

- **if statement :**

The Java if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

```
if(condition)
{
  // Statements to execute if
  // condition is true
}
```

- **Make a program that will show positive number.**

```
package basicjava;
import java.util.Scanner;

public class PositiveDemo{
   public static void main(String[] args)
   {
      Scanner Input = new Scanner(System.in);
      int num;
      System.out.print("Enter any integer : ");
      num = Input.nextInt();
      if(num>0){
         System.out.printf("It's a positive number.",
args);
      }
   }
}
```

Output:
Enter any integer : 5
It's a positive number.

- **if-else statement :**

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false. Here comes the else statement. We can use the else statement with the if statement to execute a block of code when the condition is false.

Syntax:

```
if (condition)
{
   // Executes this block if
   // condition is true
}
else
{
   // Executes this block if
   // condition is false
}
```

- **Make a program that will show positive and negative number.**

```
package basicjava;
import java.util.Scanner;

public class PositiveDemo{
   public static void main(String[] args)
   {
      Scanner Input = new Scanner(System.in);
```

```
        int num;
        System.out.print("Enter any integer : ");
        num = Input.nextInt();
        if(num>0){
            System.out.printf("It's a positive number.",
args);
        }
        else{
            System.out.println("It's a negative number.");
        }
    }
}
```

Output:
Enter any integer : -2
It's a negative number.

- **if-else-if ladder syntax:**

Java if-else-if ladder is used to decide among multiple options. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

Syntax:

```
if (condition){
    statement 1;
  }
else if (condition){
    statement 2;
  }
.
.
else{
    statement;
  }
```

- **Make a program that will show whether an integer number is positive, negative or zero number.**

```
package basicjava;
import java.util.Scanner;

public class PositiveDemo{
   public static void main(String[] args)
   {
      Scanner Input = new Scanner(System.in);
      int num;
      System.out.print("Enter any integer : ");
      num = Input.nextInt();
      if(num>0){
         System.out.printf("It's a positive number.",
args);
      }
      else if(num<0){
         System.out.println("It's a negative number.");
      }
      else
         System.out.println("It's zero");

   }
}
```

Output:
Enter any integer : 0
It's zero

- **Make a program that will show either a positive integer number is even or odd.**

```
package basicjava;
import java.util.Scanner;

public class PositiveDemo{
   public static void main(String[] args)
   {
      Scanner Input = new Scanner(System.in);
      int num;
      System.out.print("Enter any positive integer :
");
      num = Input.nextInt();
      if(num%2==0){
         System.out.printf("Even.", args);
      }
      else
         System.out.println("Odd");

   }
}
```

Output:
Enter any positive integer : 3
Odd

- **Make a program that will be able to check Vowel/Consonant.**

```
package basicjava;
import java.util.Scanner;

public class PositiveDemo{
   public static void main(String[] args)
   {
      Scanner Input = new Scanner(System.in);
      char  letter;
      System.out.print("Enter any letter : ");
      letter = Input.next().charAt(0);
      if(letter == 'a' || letter == 'e' || letter == 'i' ||
letter == 'o' || letter == 'u' || letter == 'A' || letter
== 'E' || letter == 'I' || letter == 'O' || letter == 'U'){
         System.out.println("Vowel");
      }
      else
         System.out.println("Consonant");

   }
}
```

Output:
Enter any letter : a
Vowel

Here,
charAt(0) will take the first character of string because here we mentioned the index at 0 position of the string. That's why it will take only the first letter of the string.

- **Make a program that will be able to check capital letter and small letter.**

```
package basicjava;
import java.util.Scanner;

public class PositiveDemo{
   public static void main(String[] args)
   {
      Scanner Input = new Scanner(System.in);
      char  letter;
      System.out.print("Enter any letter : ");
      letter = Input.next().charAt(0);
      if(letter>='a' && letter<='z'){
         System.out.println("Small letter");
      }
      else if(letter>='A' && letter<='Z'){
         System.out.println("Capital letter");
      }
      else{
         System.out.print("Not a letter");
      }
   }
}
```

Output:

```
Enter any letter : A
Capital letter
```

- **Nested if-else condition:**

Nested if-else statements in Java allow you to have an if statement inside another if statement. This is useful when you need to check multiple conditions in a specific order.

Syntax:

```
if (condition1) {

    if (condition2) {

    } else {

    }
} else {

}
```

- **Make a program that will be able to check capital letter and small letter.**

```
public class Test {
   public static void main(String[] args) {
      int i = 10;

      if (i == 10) {
         if (i < 15)
            System.out.println("i is smaller than 15");
         if (i < 12)
            System.out.println("i is smaller than 12 too");
         else
            System.out.println("i is greater than 15");
      }
   }
}
```

Output:
i is smaller than 15
i is smaller than 12 too

- **Make a program that will be able to check capital letter and small letter.**

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter your age: ");
    int age = scanner.nextInt();

    if (age >= 18) {
      System.out.println("You are an adult.");

      System.out.print("Do you have a driver's license? 
(yes/no): ");
      String license = scanner.next();

      if (license.equalsIgnoreCase("yes")) {
        System.out.println("You can drive!");
      } else if (license.equalsIgnoreCase("no")) {
        System.out.println("You cannot drive without a 
license.");
      } else {
        System.out.println("Invalid input for the license. 
Please enter 'yes' or 'no'.");
      }
```

```
 } else {
 System.out.println("You are a minor.");
 }

 scanner.close();
 }
}
```

Output:

```
Enter your age: 23
You are an adult.
Do you have a driver's license? (yes/no): no
You cannot drive without a license.
```

- **switch statement:**

The switch statement is a multi-way branch statement. In simple words, the Java switch statement executes one statement from multiple conditions. It is like an if-else-if ladder statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression. Basically, the expression can be a byte, short, char, and int primitive data types. It basically tests the equality of variables against multiple values.

1. switch statement provides a better alternative than a large series of if-else if statements.
2. switch statement executes one statement from multiple conditions.
3. Keywords used in switch statement -> switch, case, break, default.

Note: Java switch expression must be of byte, short, int, long(with its Wrapper type), enums and string. Beginning with JDK7, it also works with enumerated types (Enums in java), the String class, and Wrapper classes.

Syntax :

```
switch (expression){
      case value1:
                    //codes to be executed
                    break;
     case value1:
                    //codes to be executed
                    break;
     case value1:
                   //codes to be executed
                   break;
      default:
                  //codes to be executed
```

- **Make a digit spelling program using switch statement.**

```
package basicjava;
import java.util.Scanner;

public class PositiveDemo{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int digit;
      System.out.print("Enter any digit : ");
      digit = inputt.nextInt();
      switch(digit){
         case 0:
            System.out.println("Zero");
```

```
    break;
case 1:
    System.out.println("One");
    break;
case 2:
    System.out.println("Two");
    break;
case 3:
    System.out.println("Three");
    break;
case 4:
    System.out.println("Four");
    break;
case 5:
    System.out.println("Five");
    break;
case 6:
    System.out.println("Six");
    break;
case 7:
    System.out.println("Seven");
    break;
case 8:
    System.out.println("Eight");
    break;
case 9:
    System.out.println("Nine");
    break;
```

```
            default:
                System.out.println("Not a valied character");
        }
    }
}
```

Output:

```
Enter any digit : 7
Seven
```

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String fruit = scan.next();

      switch(fruit){
       case "Mango":
       System.out.println("king of fruits");
       break;
       case "Apple":
       System.out.println("A sweet red fruit");
       break;
       case "Orange":
       System.out.println("Round fruit");
       break;
       case "Grapes":
       System.out.println("Small fruit");
       break;
       default:
       System.out.println("Print a valid fruit");
      }
   }
}
```

Output:

Orange
Round fruit

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter a day (e.g., Monday, Tuesday,
etc.): ");
      String day = scanner.nextLine();

      switch (day) {
         case "Monday":
            System.out.println("It's the first day of the
week.");
            break;
         case "Tuesday":
         case "Wednesday":
         case "Thursday":
            System.out.println("It's a weekday.");
            break;
         case "Friday":
            System.out.println("It's the end of the workweek.
TGIF!");
            break;
         case "Saturday":
         case "Sunday":
            System.out.println("It's the weekend! Time to
relax.");
```

```
            break;
        default:
            System.out.println("Invalid input or an
unrecognized day.");
            break;
    }

    scanner.close();
  }
}
```

Output:

Enter a day (e.g., Monday, Tuesday, etc.): Wednesday
It's a weekday.

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);

    int day = scan.nextInt();

    switch (day) {
      case 1:
      case 2:
      case 3:
      case 4:
      case 5:
        System.out.println("Weekday");
        break;
      case 6:
      case 7:
        System.out.println("Weekend");
        break;
    }

  }
}
```

Output:
6
Weekend

- **Enhanced switch:**

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);

      int day = scan.nextInt();

      switch (day) {
         case 1 -> System.out.println("Satarday");
         case 2 -> System.out.println("Sunday");
         case 3 -> System.out.println("Monday");
         case 4 -> System.out.println("Tuesday");
         case 5 -> System.out.println("Wednesday");
         case 6 -> System.out.println("Thrusday");
         case 7 -> System.out.println("Friday");
         default -> System.out.println("Invalid day");
      }

   }
}
```

Output:

6
Thrusday

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);

      int day = scan.nextInt();

      switch (day) {
         case 1 -> {
            System.out.println("Satarday");
         }
         case 2 -> {
            System.out.println("Sunday");
         }
         case 3 -> {
            System.out.println("Monday");
         }
         case 4 -> {
            System.out.println("Tuesday");
         }
         case 5 -> {
            System.out.println("Wednesday");
         }
         case 6 -> {
            System.out.println("Thrusday");
         }
```

```
            case 7 -> {
                System.out.println("Friday");
            }
            default -> {
                System.out.println("Invalid day");
            }
        }

    }
}
```

Output:

4

Tuesday

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      int dayOfWeek = 3;

      switch (dayOfWeek) {
         case 1, 2, 3, 4, 5 -> System.out.println("Weekday");
         case 6, 7 -> System.out.println("Weekday");
      }

   }
}
```

Output:
Weekday

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    String fruit = scan.next();

    switch (fruit) {
      case "Mango" -> {
        System.out.println("king of fruits");
      }
      case "Apple" -> {
        System.out.println("A sweet red fruit");
      }
      case "Orange" -> {
        System.out.println("Round fruit");
      }
      case "Grapes" -> {
        System.out.println("Small fruit");
      }
      default -> {
        System.out.println("Ha Ha Ha");
      }
    }
  }
}
```

Output:

Apple
A sweet red fruit

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      int dayOfWeek = 3;
      String dayName = switch (dayOfWeek) {
         case 1 -> "Sunday";
         case 2 -> "Monday";
         case 3 -> "Tuesday";
         case 4 -> "Friday";
         // More cases can be added as needed
         default -> "Invalid day";
      };

      System.out.println(dayName);
   }
}
```

Output:
Tuesday

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String fruit = scan.next();

      String message = switch (fruit) {
         case "Mango" -> "king of fruits";
         case "Apple" -> "A sweet red fruit";
         case "Orange" -> "Round fruit";
         case "Grapes" -> "Small fruit";
         default -> "Print a valid fruit";
      };

      System.out.println(message);
   }
}
```

Output:
Apple
A sweet red fruit

- **Nested switch:**

In Java, nested switch statements are a way to use a switch statement inside another switch statement. This allows you to handle more complex scenarios where you have multiple levels of decision-making. The syntax for a nested switch is the same as a regular switch statement, but it is placed within the case block of another switch statement.

Syntax:

```
switch (outerExpression) {
  case outerValue1:
    // Outer case block code

    switch (innerExpression) {
      case innerValue1:
        // Inner case block code
        break;
      case innerValue2:
        // Inner case block code
        break;
      // Add more cases for the inner switch as needed
      default:
        // Default case for the inner switch
        break;
    }

    break; // End of outer case block

  case outerValue2:
    // Outer case block code
    break;
  // Add more cases for the outer switch as needed
  default:
    // Default case for the outer switch
    break;
}
```

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);

      int empID = scan.nextInt();
      String department = scan.next();

      switch (empID) {
         case 1:
            System.out.println("Hamim Talukder");
            break;
         case 2:
            System.out.println("Jim Talukder");
            break;
         case 3:
            System.out.println("Emp Number 3
");
            switch (department) {
               case "IT":
                  System.out.println("IT Department");
                  break;
               case "Management":
                  System.out.println("Management
Department");
                  break;
```

```
                default:
                    System.out.println("No department
entered");
            }
            break;
        default:
            System.out.println("Enter correct EmpID");

    }

  }
}
```

Output:

3
IT
Hridi Chowdhury
IT Department

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);

      int empID = scan.nextInt();
      String department = scan.next();

      switch (empID) {
         case 1 -> System.out.println("Hamim Talukder");
         case 2 -> System.out.println("Jim Talukder");
         case 3 -> {
            System.out.println("Emp Number 3");
            switch (department) {
               case "IT" -> System.out.println("IT
Department");
               case "Management" ->
System.out.println("Management Department");
               default -> System.out.println("No department
entered");
            };
         }
         default -> System.out.println("Enter correct
EmpID");
      }

   }
}
```

Output:

3
IT
Emp Number 3
IT Department

- **Loop statement:**

A loop statement allows us to execute a statement or group of statements multiple times.

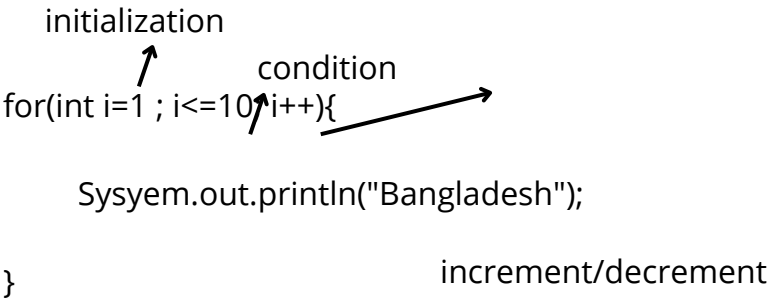
Here are the different types of loops in Java :

- For Loop
- While Loop
- Do-While Loop
- Enhanced For Loop (For-each Loop)
- Nested loops

- **for loop statement:**

The for loop is used when you know the number of iterations required. It consists of three parts: initialization, condition, and iteration expression.

Syntax:



- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      for(int i=1; i<=5; i++){
         System.out.println("Hamim");
      }
   }
}
```

Output:

Hamim
Hamim
Hamim
Hamim
Hamim

- **while loop statement:**

The while loop is used when the number of iterations is not known in advance, but you have a condition that decides whether to continue or exit the loop.

Syntax:

```
initialization;
while(condition){
        statement(s)
        increment/decrement
}
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int i=1;
      while(i<=5){
         System.out.println("Hamim");
         i++;
      }
   }
}
```

Output:

Hamim
Hamim
Hamim
Hamim
Hamim

- **do while loop statement:**

The do-while loop is similar to the while loop, but it guarantees that the loop body executes at least once before checking the condition.

Syntax:

```
initialization;
do{
        statement(s)
        increment/decrement
}
while(condition);
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int i=1;
      do{
         System.out.println("Hamim");
         i++;
      }
      while(i<=5);
   }
}
```

Output:
Hamim
Hamim
Hamim
Hamim
Hamim

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int i=1;
      do{
         System.out.println("Jim");
         i++;
      }while(i<=5);
   }
}
```

Output:
Jim
Jim
Jim
Jim
Jim

- **Enhanced For Loop (For-each Loop):**

In Java, the "for-each" loop is also known as the "enhanced for loop." It provides a simpler and more concise way to iterate over elements in an array or a collection (e.g., ArrayList, LinkedList) without the need for explicit indexing or using the traditional "for" loop.

The enhanced for loop simplifies iterating over arrays and collections. It doesn't require explicit indexing.

Syntax:

```
int[] numbers = {1, 2, 3, 4, 5};
for (int num : numbers) {
   System.out.print(num + " ");
}
```

Syntax:

```
for (datatype element : collection) {
   // code to be executed for each element
}
```

Here's what each part of the for-each loop means:

datatype: This is the data type of the elements in the collection. It should match the type of elements stored in the collection.

element: This is a variable that represents the current element being iterated over. You can use this variable within the loop to access the current element's value.

collection: This is the array or collection over which the loop iterates. It could be an array, a List, a Set, or any other class that implements the Iterable interface.

- **Iterating over an array:**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      int[] numbers = { 1, 2, 3, 4, 5 };
      for (int num : numbers) {
         System.out.print(num + " ");
      }
   }
}
```

Output:
1 2 3 4 5

- **Iterating over an ArrayList:**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

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

       ArrayList<String> names = new ArrayList<>();
       names.add("Alice");
       names.add("Bob");
       names.add("Charlie");

       for (String name : names) {
          System.out.println(name);
       }
   }
}
```

Output:
Alice
Bob
Charlie

- **Iterating over a String:**

- **What is the output of the following Java program fragment:**

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

      String message = "Hello";
      for (char ch : message.toCharArray()) {
         System.out.print(ch + " ");
      }

   }
}
```

Output:
H e l l o

- **Nested loops :**

Nested loops refer to the concept of using one loop inside another loop. This means one loop is placed inside the body of another loop. By using nested loops, you can perform repetitive tasks in multiple dimensions, such as working with 2D arrays, matrices, or creating patterns.
In Java, you can nest any type of loop inside any other loop, including for, while, do-while, and even enhanced for-each loops.

Syntax of for loop :

```
for (initialization; condition; iteration) {
   // Outer loop code

   for (initialization; condition; iteration) {
      // Inner loop code
   }
}
```

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      int tableSize = 10;

      // Nested for loop to create the multiplication table
      for (int i = 1; i <= tableSize; i++) {
         for (int j = 1; j <= tableSize; j++) {
            System.out.printf("%4d", i * j);
         }
         System.out.println();
      }
   }
}
```

Output:

```
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

break statement:

break statement is used to break loop, switch statement.
It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int i, j;
      for(i=1 ;i<=2; i++){
         for(j=1 ;j<=100; j++){
            if(j==5){
               break;
            }
            System.out.println("i and j = "+i+j);
         }
      }
   }
}
```

Output:
i and j = 11
i and j = 12
i and j = 13
i and j = 14
i and j = 21
i and j = 22
i and j = 23
i and j = 24

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int i, j;
      for(i=1 ;i<=3; i++){
         for(j=1 ;j<=100; j++){
            if(j==5){
               break;
            }
            System.out.println("i and j = "+i+j);
         }
         if(i==2){
            break;
         }
      }
   }
}
```

Output:
i and j = 11
i and j = 12
i and j = 13
i and j = 14
i and j = 21
i and j = 22
i and j = 23
i and j = 24

continue statement:
The continue statement skips the current iteration of a loop (for, while, and do....while loop).

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int i, j;
      for(i=1 ;i<=3; i++){
         if(i==2){
            continue;
         }
```

```
        for(j=1 ;j<=6; j++){
            if(j==5){
                continue;
            }
            System.out.println("i and j = "+i+j);
        }
    }
  }
}
```

Output:

```
i and j = 11
i and j = 12
i and j = 13
i and j = 14
i and j = 16
i and j = 31
i and j = 32
i and j = 33
i and j = 34
i and j = 36
```

- **Make a program that will print Even number from m to n.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, start, end;
      System.out.print("Enter start and ending
number : ");
      start=input.nextInt();
      end=input.nextInt();
      for(i=start ;i<=end; i++){
        if(i%2==0){
           System.out.println(i);
        }
      }
   }
}
```

Output:

```
Enter start and ending number : 1 10
2
4
6
8
10
```

- **Make a program that will print the sum of even numbers from m to n.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, start, end, sum=0;
      System.out.print("Enter initial number : ");
      start=input.nextInt();
      System.out.print("Enter final number : ");
      end=input.nextInt();
      for(i=start ;i<=end; i++){
        if(i%2==0){
          sum=sum+i;
          System.out.print("  "+i);
        }
      }
      System.out.println();
      System.out.println("The sum is : "+sum);
   }
}
```

Output:

```
Enter initial number : 1
Enter final number : 10
  2  4  6  8  10
The sum is : 30
```

- **Make a program that will print the sum of 1 + 2 + 3 + ... + n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i++){
         System.out.print(i+" ");
        sum = sum + i;
      }
      System.out.println();
      System.out.println("The sum is : "+sum);
   }
}
```

Output:
Enter the last term number : 10
1 2 3 4 5 6 7 8 9 10
The sum is : 55

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i++){
        sum = sum + i;
      }
      System.out.println("The sum is : "+sum);
   }
}
```

Output:

```
Enter the last term number : 10
The sum is : 55
```

- **Make a program that will print the sum of 1 + 3 + 5 + ... + n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i=i+2){
        sum = sum + i;
      }
      System.out.println("The sum is : "+sum);
   }
}
```

Output:
Enter the last term number : 5
The sum is : 9

- **Make a program that will print the sum of 1 + 3 + 5 + ... + n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i=i+2){
        sum = sum + i;
      }
      System.out.println("The sum is : "+sum);
   }
}
```

Output:
Enter the last term number : 5
The sum is : 9

- **Make a program that will print the sum of 1.5 + 2.5 + 3.5 + ... + n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      double term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextDouble();
      for(double i=1.5 ;i<=term; i++){
         System.out.print(i+"  ");
        sum = sum + i;
      }
      System.out.println();
      System.out.println("The sum is : "+sum);
   }
}
```

Output:

```
Enter the last term number : 10
1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5
The sum is : 49.5
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      float i, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextFloat();
      for(i=1.5f ;i<=term; i++){
        sum = sum + i;
      }
      System.out.println("The sum is : "+sum);
   }
}
```

Output:

```
Enter the last term number : 2.5
The sum is : 4.0
```

- **Make a program that will print the sum of 1^3 + 2^3+ 3^3 + ... + n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, k, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i++){
         k = i;
         for (j = 1; j <= 2; j++) {
            k = k*i;
         }
         sum = sum + k;
      }
      System.out.println("The sum is : "+sum);
   }
}
```

Output:
Enter the last term number : 3
The sum is : 36

- **Make a program that will print the sum of 1^2 + 2^2+ 3^2 + ... + n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i++){
         System.out.print(i+"x"+i+" ");
        sum = sum + i*i;
      }
      System.out.println();
      System.out.println("The sum is : "+sum);
   }
}
```

Output:
Enter the last term number : 3
1x1 2x2 3x3
The sum is : 14

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, k, term, sum=0;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i++){
         k = i;
         for (j = 1; j <= 1; j++) {
            k = k*i;
         }
         sum = sum + k;
      }
      System.out.println("The sum is : "+sum);
   }
}
```

Output:

```
Enter the last term number : 3
The sum is : 14
```

- **Make a program that will print the sum of 1 x 2 x 3 x ..... x n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, term, sum=1;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i++){
         System.out.print(i);
         if(i!=term){
            System.out.print(" x ");
         }
        sum = sum*i;
      }
      System.out.println();
      System.out.println("The sum is : "+sum);
   }
}
```

Output:
Enter the last term number : 4
1 x 2 x 3 x 4
The sum is : 24

- **Make a program that will print the sum of 1 x 3 x 5 x ..... x n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, term, sum=1;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i=i+2){
         System.out.print(i);
         sum = sum*i;
         if(i+1==term){
            break;
         }
         else if(i!=term){
            System.out.print(" x ");
         }
      }
      System.out.println();
      System.out.println("The sum is : "+sum);
   }
}
```

Output:

```
Enter the last term number : 5
1 x 3 x 5
The sum is : 15
```

- **Make a program that will print the sum of 1.5 x 2.5 x 3.5 x ..... x n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      double term, multi=1;
      System.out.print("Enter the last term number : ");
      term = input.nextDouble();
      for(double i=1.5 ;i<=term; i++){
         System.out.print(i);
         multi = multi * i;
         if(i!=term){
            System.out.print(" x ");
         }
      }
      System.out.println();
      System.out.println("The sum is : "+multi);
   }
}
```

Output:
Enter the last term number : 3.5
1.5 x 2.5 x 3.5
The sum is : 13.125

- **Make a program that will print the sum of 1^2 x 2^2 x 3^2 x ..... x n series.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, term, multi=1;
      System.out.print("Enter the last term number : ");
      term = input.nextInt();
      for(i=1 ;i<=term; i++){
         System.out.print(i+"^"+"2");
        multi = multi * i * i;
        if(i!=term){
            System.out.print(" x ");
        }
      }
      System.out.println();
      System.out.println("The sum is : "+multi);
   }
}
```

Output:
Enter the last term number : 3
1^2 x 2^2 x 3^2
The sum is : 36

- **Make a program that will find the factorial of a number.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, num, fact=1;
      System.out.print("Enter any positive number : ");
      num = input.nextInt();
      for(i=num ;i>0; i--){
         System.out.print(i);
         fact = fact*i;
         if(i!=1){
           System.out.print(" x ");
         }
      }
      System.out.println();
      System.out.println("Factorial of "+num+"! is "+fact);
   }
}
```

Output:
Enter any positive number : 5
5 x 4 x 3 x 2 x 1
Factorial of 5! is 120

- **Make a program that will print multiplication table.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, num;
      System.out.print("Enter any number : ");
      num = input.nextInt();
      for(i=1 ;i<=10; i++){
         System.out.println(num+" x "+i+" = "+num*i);
      }
   }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, num;
      System.out.print("Enter any number : ");
      num = input.nextInt();
      for(i=1 ;i<=10; i++){
         System.out.print(num+" x "+i+" = "+num*i+"\n");
      }
   }
}
```

Output:

```
Enter any number : 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

- **Make a program that will print multiplication tables from m to n.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      int i, j, m, n;
      System.out.print("Enter Initial number : ");
      m = input.nextInt();
      System.out.print("Enter Final number : ");
      n = input.nextInt();
      for(i = m ;i <= n; i++){
         for (j = 1; j <= 10; j++) {
            System.out.print(i+" x "+j+" = "+i*j+"\n");
         }
         System.out.println();
      }
   }
}
```

- **Make a program that will be able to find prime number.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int i, num, count = 0;
      System.out.print("Enter any positive number : ");
      num = scan.nextInt();
      if(num==0 || num==1){
         System.out.println("Not prime");
      }
      else{
        for (i = 2; i <= num/2 ; i++) {
         if(num%i==0){
            count++;
            break;
         }
      }
      if(count == 0){
         System.out.println("Prime number");
      }
      else
         System.out.println("Not prime");
      }
   }
}
```

Output:
Enter any positive number : 5
Not prime

- **Make a program that will be able to find prime number from m to n.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int i, j, m, n, count;
      System.out.print("Enter initial number : ");
      m = scan.nextInt();
      System.out.print("Enter ending number : ");
      n = scan.nextInt();
      for (i = m; i <= n; i++) {
         count = 0;
         if(i<2){
         continue;
         }
         else{
            for (j = 2; j <= i/2 ; j++) {
               if(i%j==0){
                  count++;
                  break;
               }
            }
```

```
            if(count == 0){
               System.out.println("Prime number = "+i);
            }
         }
      }

   }
}
```

Output:
Enter initial number : 1
Enter ending number : 10
Prime number = 1
Prime number = 2
Prime number = 3
Prime number = 5
Prime number = 7

- **Make a program that will be able to print Fibonacci numbers.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      // 0 1 1 2 3 5 8 .....
      Scanner scan = new Scanner(System.in);
      int i, num, count = 0;
      System.out.print("How many numbers : ");
      num = scan.nextInt();
      int first = 0, second = 1, fibo;
      System.out.print(first+"  "+second);
      for (i = 3; i <= num; i++) {
         fibo = first + second;
         System.out.print("  "+fibo);
         first = second;
         second = fibo;
      }
   }
}
```

Output:
How many numbers : 6
0 1 1 2 3 5

- **Make a program that will be able to print sum of digits.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int i, num, sum = 0, rem;
      System.out.print("Enter any positive integer
number : ");
      num = scan.nextInt();
      for (i = num;  i>0; i = i/10) {
         rem = i%10;
         sum = sum+rem;
      }
      System.out.println("Sum is : "+sum);
   }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int i, num, sum = 0, rem;
```

```
        System.out.print("Enter any positive integer
number : ");
        num = scan.nextInt();
        i = num;
        while(i!=0) {
            rem = i%10;
            sum = sum+rem;
            i = i/10;
        }
        System.out.println("Sum is : "+sum);
    }
}
```

Output:
Enter any positive integer number : 123
Sum is : 6

- **Make a program that will be able to print reverse integer number**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int i, num, sum = 0, rem;
      System.out.print("Enter any positive integer
number : ");
      num = scan.nextInt();
      for (i = num;  i!=0; i = i/10) {
         rem = i%10;
         sum = sum*10+rem;
      }
      System.out.println("Revers number is : "+sum);
   }
}
```

Output:
Enter any positive integer number : 79576
Revers number is : 67597

- **Make a program that will be able to show palindrom number.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int i, num, sum = 0, rem;
      System.out.print("Enter any positive integer
number : ");
      num = scan.nextInt();
      for (i = num;  i!=0; i = i/10) {
         rem = i%10;
         sum = sum*10+rem;
      }
      if(num == sum){
         System.out.println(num+" is a palindrom
number");
      }
      else{
         System.out.println(num+" is not a palindrom
number");
      }
   }
}
```

Output:
Enter any positive integer number : 121
121 is a palindrom number

- **Make a program that will show whether a given number is armstrong or not.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int i, num, sum = 0, rem;
      System.out.print("Enter any positive integer
number : ");
      num = scan.nextInt();
      for (i = num;  i!=0; i = i/10) {
         rem = i%10;
         sum = sum+rem*rem*rem;
      }
      if(num == sum){
         System.out.println(num+" is a armstrong
number");
      }
      else{
         System.out.println(num+" is not a armstrong
number");
      }
   }
}
```

Output:
Enter any positive integer number : 153
153 is a armstrong number

7. Functions/Methods

In Java, functions or methods are blocks of code that perform specific tasks and can be defined within classes. They encapsulate a set of actions and allow you to call and reuse code multiple times. Methods in Java can have input parameters (arguments) and return values.

The basic syntax for defining a method in Java:

```
accessModi  returnType  methodName( parameter list )
{
   // body of the function/method - code that performs
      the desired task

   // Optionally, a return statement can be used to
     return a value
}
```

Let's break down the components of the method definition:

accessModifier: Specifies the visibility and accessibility of the method. It can be public, private, protected, or package-private (no access modifier).

returnType: Specifies the data type of the value that the method returns. If the method does not return anything, the void keyword is used.

methodName: The name of the method, which can be any valid identifier.

Parameters/arguments: A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

Method body: The set of statements enclosed in curly braces {} that define the actions performed by the method.

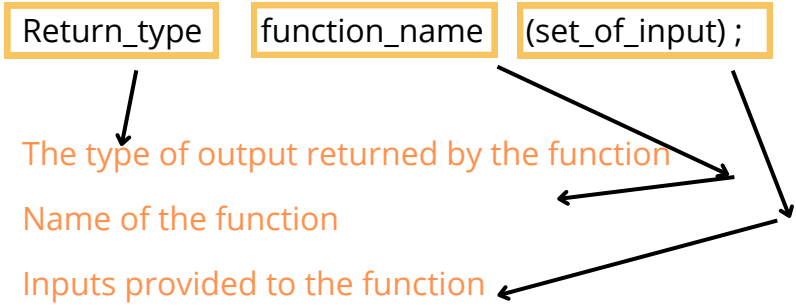
return statement (optional): If the method has a non-void return type, you can use the return statement to send back a value of the specified return type.

Basic of Function:

Definition:

Function is basically a set of statements that takes inputs, perform some computation and products output.

Syntax:



Why Functions ?

There are two important reasons of why we are using function:

1. Reusability:
Once the function is defined, it can be reused over and over again.
2. abstraction:
If you are just using the function in your program then you don't have to worry about how it works inside!

example: println() function

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      int result = addNumbers(5, 10);
      System.out.println("The result is: " + result);
   }

   public static int addNumbers(int a, int b) {
      int sum = a + b;
      return sum;
   }
}
```

Output:
The result is: 15

Here,
In this example, we have a method called addNumbers, which takes two int parameters and returns their sum. The main method calls addNumbers(5, 10) and prints the result, which is 15.

In the above code, the addNumbers method is static, so it can be called directly from the main method without creating an instance of the Example class.
If you want to use instance-specific data or access non-static members of the class inside the method, you would need to make it a non-static (instance) method and call it through an instance of the class.

In the example provided, the method addNumbers is defined as a static method (public static int addNumbers(int a, int b)). There are a few reasons why it is declared as static:

- Access from the main method: In Java, the main method (entry point of the program) is also declared as static. When the program starts, it does not create an instance of the class to call the main method. Instead, the main method is called directly on the class itself. To call a method from the static main method, the called method must also be static.

- No instance required: Static methods belong to the class and not to any specific instance (object) of the class. They can be accessed without creating an instance of the class. In contrast, non-static (instance) methods require an object to be created before they can be called.

- Utility methods: Static methods are often used for utility functions or helper methods that do not depend on any instance-specific data. They perform generic tasks that are independent of the state of the object.

- Simplicity and Efficiency: Static methods can be called directly using the class name, without the need to create an object. This makes the code concise and can lead to more efficient memory usage since there is no need to allocate memory for objects.

Java supports method overloading, which means you can define multiple methods with the same name but different parameter lists. The method that gets called depends on the number or types of arguments passed to it.

Remember that methods declared with void as the return type do not return any value. They are used to perform actions without returning data. Methods with non-void return types must use the return statement to return a value of the specified type.

- **What is the output of the following Java program fragment:**

```
public class Test {

   static int areaOfRect(int length, int breadth) {
      int area;
      area = length * breadth;
      return area;
   }

   public static void main(String[] args) {
      int l = 10, b = 5;
      int area = areaOfRect(l, b);
      System.out.println(area);
   }

}
```

Output:
50

- **What is the output of the following Java program fragment:**

```
public class Test {

   public static int areaOfRect(int length, int breadth) {
      int area;
      area = length * breadth;
      return area;
   }

   public static void main(String[] args) {
      int l = 10, b = 5;
      int area = areaOfRect(l, b);
      System.out.println(area);

      l = 50;
      b = 20;
      area = areaOfRect(l, b);
      System.out.printf("%d\n", area);
   }

}
```

Output:
50
1000

- **What is the output of the following Java program fragment:**

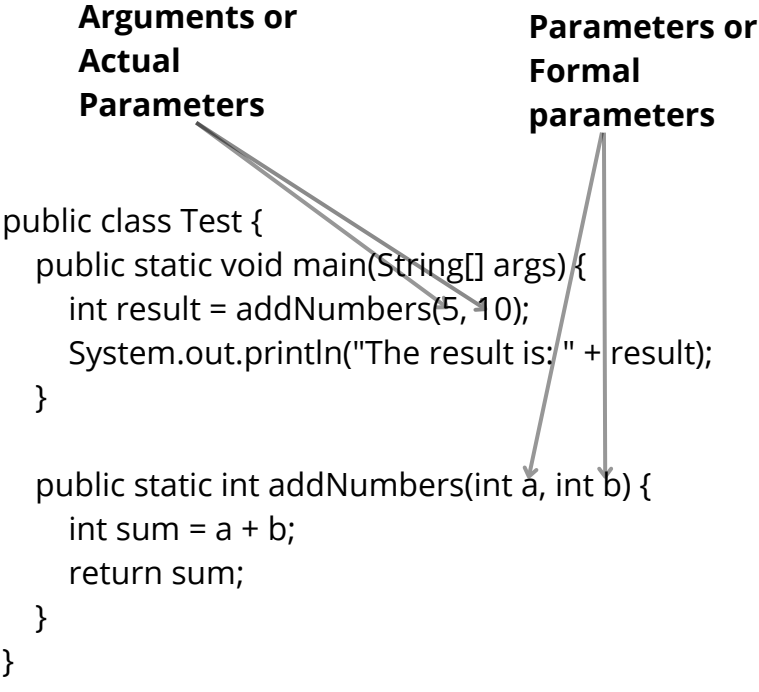
```
public class Test {

   public static char fun()
   {
     return 'a';
   }

   public static void main(String[] args) {
      char c = fun();
      System.out.println("Character is : "+c);
   }

}
```

Output:
Character is : a

Difference between an Argument and Parameter:

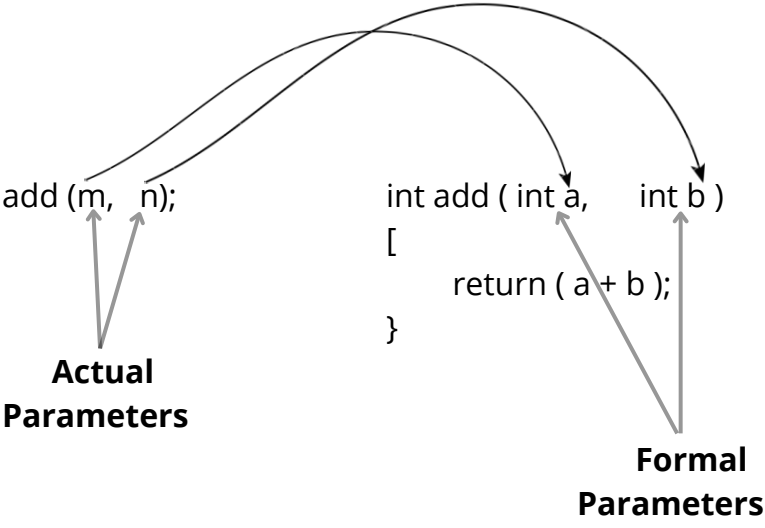
Parameter: is a variable in the declaration of the function.
Argument: is the actual value of the parameter that gets passed to the function.

Note: Parameter is also called as Formal Parameter and Argument is also called as Actual parameter.



Recall:

Actual Parameters: The parameters passed to a function.
Formal Parameters: the parameters received by a function.



Type of Function call

While calling a function, there are two ways that arguments can be passed to a function:

Call Type	Description
Call by value	This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.
Call by reference	This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

Remember that,
Java language only supports pass by or call by value not call by reference because it doesn't have pointer concept.

Call by Value:

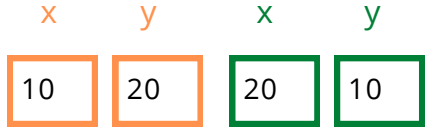
Here values of actual parameters will be copied to formal parameters and these two different parameters store values in different locations.

- In this case, changes made to the parameter inside the function have no effect on the argument.
- By default, Java/C++ programming language uses call by value method to pass arguments. In general, this means that code within a function cannot alter the arguments used to call the function. Consider the function swap() definition as follows.

```
int x = 10, y = 20;
fun(x, y);
printf("x = %d, y = %d,x,y);
```

```
int fun(int x, int y)
{
   x=20;
   y=10;
}
```

Output:
x=10, y=20



- **What will be the output of the program given bellow.**

```
public class Test {
   public static void main(String[] args) {
      String name = "Hamim";

      System.out.println(name);

      change(name);

      System.out.println(name);
   }

   static void change(String naam) {
      naam = "Hridi";
   }
}
```

Output:
Hamim
Hamim

- **What will be the output of the program given bellow.**

```
public class Test {
   public static void main(String[] args) {
      int a = 10, b = 20;
      swap(a,b);
      System.out.println(a +" "+b);
   }

   static void swap(int x, int y) {
      int temp = x;
      x = y;
      y = temp;
   }
}
```

Output:
10 20

Here,
In the swap() function all variables are assigned by the new object which is a copy of the a and b variables. That's why it can't swap these two variables a and b.

- **What will be the output of the program given bellow.**

```
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      int[] arr = {1, 3, 2, 45, 6};
      change(arr);
      System.out.println(Arrays.toString(arr));
   }

   static void change(int[] nums) {
      nums[0] = 99;
   }
}
```

Output:
[99, 3, 2, 45, 6]

Here,
In Array when we pass the array in the change() function nums[ ] array actually point the heap memory address of these array index and if we change any of value then it will be changed for both. That's why the arr[ ] got changed.
But it is actually call by value.

- **Make a program that will add two numbers using function.**

```
import java.util.Scanner;

public class Test {

  public static int addNumbers(int a, int b) {
    int sum = a + b;
    return sum;
  }

  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    int num1, num2;
    System.out.print("Enter two integer numbers : ");
    num1 = scan.nextInt();
    num2 = scan.nextInt();
    int result = addNumbers(num1, num2);
    System.out.println("The result is: " + result);
  }

}
```

or,

```
import java.util.Scanner;

public class Test {

  public static int addNumbers(int a, int b) {
    int sum = a + b;
    return sum;
  }

  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    int num1, num2;
    System.out.print("Enter two integer numbers : ");
    num1 = scan.nextInt();
    num2 = scan.nextInt();
    System.out.println("The result is: " +
addNumbers(num1, num2));
    scan.close();
  }

}
```

Output:

```
Enter two integer numbers : 12 2
The result is: 14
```

- **What will be the output of the program given bellow.**

```
public class Test {

   public static int test_function(int x) {
      int y = x;
      x = 2 * y;
      return (x * y);
   }

   public static void main(String[] args) {
      int x = 10, y = 20, z = 30;
      z = test_function(x);
      System.out.printf("%d %d %d\n", x, y, z);
   }

}
```

Output:
10 20 200

Here,
The output will be 10 20 200 where we expect that the output will be 20 10 200. Why it is happening ? The reason is every functions have different variables. It is called local variable. We have print the value of x, y of main function but do not print the value of x, y of test_function. The existence of local variable of a function do not stay in other function.
If we want the existence of variable in all function then we have to declare global variable.

- **Make a program that will show sum of 1,2 and 5,6 using function and also print 5 without function.**

```
public class Test {
   public static int sum(int a, int b) {
      return a + b;
   }

   public static void main(String[] args) {
      int result = sum(1, 2);
      System.out.println("The sum is: " + result);

      result = sum(5, 6);
      System.out.println("The sum is: " + result);

      System.out.println("The sum is: " + 5);
   }
}
```

or,

```
public class Main {
   public static int sum(int a, int b) {
      return a + b;
   }

   public static void main(String[] args) {
      System.out.println("The sum is : " + sum(1, 2));
      System.out.println("The sum is : " + sum(5, 6));
      System.out.println("The sum is : " + 5);
   }
}
```

Output:
The sum is: 3
The sum is: 11
The sum is: 5

- **Make a program that will show sum of 1,2,3 and 5,6,7 using function.**

```
public class Main {
   public static int sum(int a, int b, int c) {
      return a + b + c;
   }

   public static void main(String[] args) {
      System.out.println("The sum is : " + sum(1, 2, 3));
      System.out.println("The sum is : " + sum(5, 6, 7));
   }
}
```

```
public class Main {

   public static void sum(int a, int b, int c) {
      System.out.println("The sum is : " + (a + b + c));
   }

   public static void main(String[] args) {
      sum(1, 2, 3);
      sum(5, 6, 7);
   }
}
```

Output:
The sum is : 6
The sum is : 18

- **Make a program that will show sum of 1,2,3 using function.**

```
public class Main {

   public static void sum(int a, int b, int c) {
      System.out.println("The sum is : " + (a + b + c));
   }

   public static void main(String[] args) {
      sum(1, 2, 3);
   }
}
```

or,

```
public class Test {
   public static int sum(int a, int b, int c) {
      System.out.println("The sum is : " + (a + b + c));
      return 0;
   }

   public static void main(String[] args) {
      sum(1, 2, 3);
   }
}
```

Output:
The sum is : 6

- **Make a program that will show sum of 1,2,3 and 5,6,7 and also subtraction of 5,4 using function.**

```
public class Main {

   public static void sum(int a, int b, int c) {
      System.out.println("The sum is : " + (a + b + c));
   }

   public static void sub(int a, int b) {
      System.out.println("The subtraction is : " + (a - b));
   }

   public static void main(String[] args) {
      sum(1, 2, 3);
      sum(5, 6, 7);
      sub(5, 4);
   }
}
```

Output:

```
The sum is : 6
The sum is : 18
The subtraction is : 1
```

- **Make a program that will show square of any integer number using function.**

```
import java.util.Scanner;

public class Main {

   public static int square(int a) {
      return a * a;
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter any integer number : ");
      int num = scanner.nextInt();

      System.out.println("Square is : " + square(num));

      scanner.close();
   }
}
```

Output:

```
Enter any integer number : 5
Square is : 25
```

- **Make a program that will show area of a triangle using function.**

```
import java.util.Scanner;

public class Main {
   public static double areaOfTriangle(double base,
double height) {
      return 0.5 * base * height;
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      double base, height, result;
      System.out.print("Enter base of the triangle: ");
      base = scanner.nextDouble();
      System.out.print("Enter height of the triangle: ");
      height = scanner.nextDouble();
      result = areaOfTriangle(base, height);
      System.out.println("Area of triangle is: " + result);
      scanner.close();
   }
}
```

Output:

```
Enter base of the triangle: 5
Enter height of the triangle: 8
Area of triangle is: 20.0
```

- **Make a program that will calculate power(base, exponent) using function.**

```
import java.util.Scanner;

public class Main {
   public static double power(double a, int b) {
      double result = 1;
      for (int i = 1; i <= b; i++) {
         result *= a;
      }
      return result;
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      double base, ans;
      int exponent;
      System.out.print("Enter base in double and exponent
in integer: ");
      base = scanner.nextDouble();
      exponent = scanner.nextInt();
      ans = power(base, exponent);
      System.out.println("Ans: " + ans);
      scanner.close();
   }
}
```

Output:

```
Enter base in double and exponent in integer: 2.5 3
Ans: 15.625
```

or,

```
public class Main {

   public static void power(double base, double exp) {
      double result = 1;
      for (double i = 1; i <= exp; i++) {
         result *= base;
      }
      System.out.println(result);
   }

   public static void main(String[] args) {
      power(2, 3);
   }
}
```

Output:

8.0

- **Make a program that will show Passing Array to function.**

```
public class Main {

   public static void display(int[] a) {
      for (int i = 0; i < a.length; i++) {
         System.out.print(a[i] + " ");
      }
   }

   public static void main(String[] args) {
      int[] num = { 10, 20, 30, 40, 50 };
      display(num);
   }
}
```

Output:
10 20 30 40 50

- **Make a program that will show maximum value from an array using function.**

```
public class Main {

   public static int maximum(int[] a) {
      int max = a[0];

      for (int i = 1; i < a.length; i++) {
         if (max < a[i]) {
            max = a[i];
         }
      }

      return max;
   }

   public static void main(String[] args) {
      int[] num = { 10, 20, 30, 40, 50 };
      int maximumValue = maximum(num);
      System.out.println("Maximum value is " +
maximumValue);
   }
}
```

Output:
Maximum value is 50

- **Make a program that will show maximum value from an array using function.**

```
import java.util.Scanner;

public class Main {
   public static float maximum(float[] a, int n) {
      float max = a[0];
      for (int i = 1; i < n; i++) {
         if (max < a[i]) {
            max = a[i];
         }
      }
      return max;
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int n;
      System.out.print("How many numbers do you want
to check: ");
      n = scanner.nextInt();
      float[] num = new float[n];
      for (int i = 0; i < n; i++) {
         num[i] = scanner.nextFloat();
      }

      System.out.println("Maximum number = " +
maximum(num, n));

      scanner.close();
   }
}
```

Output:
How many numbers do you want to check: 5
3 2 5 9 0
Maximum number = 9.0

- **Make a program that will show Passing String to function.**

```
public class Test {

   public static void display(char[] str2) {
      int i = 0;
      while (i < str2.length) {
         System.out.println(str2[i]);
         i++;
      }
   }

   public static void main(String[] args) {
      char[] str1 = "Hamim".toCharArray();
      display(str1);
   }
}
```

Output:
H
a
m
i
m

- **Make a function that takes an array and returns the array after adding 1 with all the elements of the array in Java.**

```
public class Test {

    public static int[] addOneToArray(int[] inputArray) {
        int[] resultArray = new int[inputArray.length];

        for (int i = 0; i < inputArray.length; i++) {
            resultArray[i] = inputArray[i] + 1;
        }

        return resultArray;
    }

    public static void main(String[] args) {
        int[] originalArray = {1, 2, 3, 4, 5};

        int[] newArray = addOneToArray(originalArray);

        System.out.print("Original Array: ");
        for (int num : originalArray) {
            System.out.print(num + " ");
        }
        System.out.println();

        System.out.print("New Array after adding 1: ");
        for (int num : newArray) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output:

Original Array: 1 2 3 4 5

New Array after adding 1: 2 3 4 5 6

or,

```
import java.util.Arrays;

public class Test {

   public static int[] addOneToArray(int[] inputArray) {
      int[] resultArray = new int[inputArray.length];

      for (int i = 0; i < inputArray.length; i++) {
         resultArray[i] = inputArray[i] + 1;
      }

      return resultArray;
   }

   public static void main(String[] args) {
      int[] originalArray = {1, 2, 3, 4, 5};

      System.out.println("New Array after adding 1:
"+Arrays.toString(originalArray));

      System.out.print("New Array after adding 1:
"+Arrays.toString(addOneToArray(originalArray)));
   }
}
```

Output:

New Array after adding 1: [1, 2, 3, 4, 5]
New Array after adding 1: [2, 3, 4, 5, 6]

- **Make a function that will take an array and that will search a target element from the array. After finding the elements that are equal to the target element from the array it will return the indexes of them by an array.**

```
import java.util.ArrayList;
import java.util.List;

public class Test {

    public static int[] searchIndexes(int[] inputArray, int
target) {
        List<Integer> indexList = new ArrayList<>();

        for (int i = 0; i < inputArray.length; i++) {
            if (inputArray[i] == target) {
                indexList.add(i);
            }
        }

        // Convert the List<Integer> to int[]
        int[] indexes = new int[indexList.size()];
        for (int i = 0; i < indexes.length; i++) {
            indexes[i] = indexList.get(i);
        }

        return indexes;
    }
```

```
    public static void main(String[] args) {
        int[] array = {4, 2, 6, 3, 2, 7, 2, 8};
        int target = 2;

        int[] indexes = searchIndexes(array, target);

        System.out.println("Target Element: " + target);
        System.out.print("Indexes of target element: ");
        for (int index : indexes) {
            System.out.print(index + " ");
        }
        System.out.println();
    }
}
```

Output:

Target Element: 2
Indexes of target element: 1 4 6

or,

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Test {

    public static int[] searchIndexes(int[] inputArray, int
target) {
        List<Integer> indexList = new ArrayList<>();

        for (int i = 0; i < inputArray.length; i++) {
            if (inputArray[i] == target) {
                indexList.add(i);
            }
        }

        // Convert the List<Integer> to int[]
        int[] indexes = new int[indexList.size()];
        for (int i = 0; i < indexes.length; i++) {
            indexes[i] = indexList.get(i);
        }

        return indexes;
    }

    public static void main(String[] args) {
        int[] array = {4, 2, 6, 3, 2, 7, 2, 8};
        int target = 2;
```

```
        System.out.println("Target Element: " + target);
        System.out.println("Indexes of target element:
"+Arrays.toString(searchIndexes(array, target)));

    }
}
```

Output:

```
Target Element: 2
Indexes of target element: [1, 4, 6]
```

8. Math class

Math Class methods helps to perform the numeric operations like square, square root, cube, cube root, exponential and trigonometric operations

Declaration :

public final class Math
  extends Object

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int x = 20;
      int y = 10;
      int result;
      System.out.println(Math.max(x,y));
   }
}
```

Output:
20

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int x = 20;
      int y = 10;
      int result;
      System.out.println(Math.min(x,y));
   }
}
```

Output:
10

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner inputt = new Scanner(System.in);
      int x = 20;
      int y = 10;
```

```
      int max, min;
       max = Math.max(x,y);
       min = Math.min(x,y);
       System.out.println("Maximum = "+max);
       System.out.println("Minimum = "+min);
   }
}
```

Output:
Maximum = 20
Minimum = 10

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
       Scanner inputt = new Scanner(System.in);
       int x = 20;
       int y = -10;
       int max, min;
       max = Math.max(x,y);
       min = Math.min(x,y);
       System.out.println("Maximum = "+max);
       System.out.println("Minimum = "+min);
       int absolute = Math.abs(y);
```

```
        System.out.println("absolute = "+absolute);
        int a = 2, b = 3;
        double power = Math.pow(a, b);
        System.out.println("a to the power b   =
"+power);
    }
}
```

Output:
Maximum = 20
Minimum = -10
absolute = 10
a to the power b   = 8.0

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
    public static void main(String[] args)
    {
        Scanner inputt = new Scanner(System.in);
        float x = 8.8f;
        float y = 8.5f;
        float z = 8.4f;
        int round1 = Math.round(x);
        int round2 = Math.round(y);
        int round3 = Math.round(z);
```

```
        System.out.printf("Round of %.2f = %d\n",x,round1);
        System.out.printf("Round of %.2f = %d\n",y,round2);
        System.out.printf("Round of %.2f = %d\n",z,round3);
    }
}
```

Output:
Round of 8.800000 = 9
Round of 8.500000 = 9
Round of 8.400000 = 8

- **What is the output of the following Java program fragment:**

```
package SecondPackage;
import java.util.Scanner;

public class Halum {
        public static void main(String[] args)
        {
            Scanner inputt = new Scanner(System.in);
            System.out.println(Math.PI);
            double pi = Math.PI;
            System.out.printf("pi = %.4f",pi);
        }
}
```

Output:
3.141592653589793
pi = 3.1416

- **What is the output of the following Java program fragment:**

```
public class Test{
   public static void main(String[] args)
   {
      System.out.println(Math.abs(-9.5));
      System.out.println(Math.sqrt(25.0));
      System.out.println(Math.pow(2,3));
      System.out.println(Math.PI);
      System.out.println(Math.log(2.0));
      System.out.println(Math.exp(1.0));
      System.out.println(Math.max(-2.3,12.8));
      System.out.println(Math.min(-2.3,12.8));
      System.out.println(Math.ceil(-5.6));
      System.out.println(Math.floor(-5.6));
   }
}
```

Output:
9.5
5.0
8.0
3.141592653589793
0.6931471805599453
2.718281828459045
12.8
-2.3
-5.0
-6.0

9. static keyword

The static keyword in Java is mainly used for memory management.

It makes the program more efficient by saving memory. The static keyword in Java is used to share the same variable or method of a given class. The users can apply static keywords with variables, methods, blocks, and nested classes. The static keyword belongs to the class than an instance of the class. The static keyword is used for a constant variable or a method that is the same for every instance of a class.

The static keyword is a non-access modifier in Java that is applicable for the following:

1. Blocks
2. Variables
3. Methods
4. Classes

Note: To create a static member(block, variable, method, nested class), precede its declaration with the keyword static.

When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object.

- **static variable :**

static variable is not related with object, it is related with class. That's why we don't need to declare an object to use static variable.

- **What is the output of the following Java program fragment:**

```
class Student {
  String name;
  int id;
  static String universityName = "DIU";
  Student(String n, int i)
  {
   name = n;
   id = i;
  }
  void displayInfo(){
   System.out.println("Name : "+name);
   System.out.println("Id : "+id);
   System.out.println("University Name :
"+universityName);
  }
}

public class Test {
   public static void main(String[] args) {
      Student s1 = new Student("Hamim",101);
      Student s2 = new Student("Mim",102);

      s1.displayInfo();
      s2.displayInfo();
   }
}
```

Output:
Name : Hamim
Id : 101
University Name : DIU
Name : Mim
Id : 102
University Name : DIU

- **What is the output of the following Java program fragment:**

```
class Student {

  static String universityName = "DIU";

}

public class Test {
   public static void main(String[] args) {
      System.out.println("Name :
"+Student.universityName);
   }
}
```

Output:
Name : DIU

- **What is the output of the following Java program fragment:**

```
class Student {

  static int count = 0;
  Student(){
   count++;
  }

  void totalStudent(){
   System.out.println("Total student = "+count);
  }
}

public class Test {
   public static void main(String[] args) {
      Student s1 = new Student();
      s1.totalStudent();

      Student s2 = new Student();
      s2.totalStudent();

      Student s3 = new Student();
      s3.totalStudent();
   }
}
```

Output:

Total student = 1
Total student = 2
Total student = 3

- **static method :**

- **What is the output of the following Java program fragment:**

```
class staticMethod {

  void display1(){
   System.out.println("I am non static method");
  }

  static void display2(){
   System.out.println("I am static method");
  }
}

public class Test {
   public static void main(String[] args) {
      staticMethod ob1 = new staticMethod();
      ob1.display1();
      staticMethod.display2();
   }
}
```

Output:
I am non static method
I am static method

Restrictions of static method :

1. static method can't use non-static members.
2. "this" and "super" keywords can't be used here.

- **static block :**

- **What is the output of the following Java program fragment:**

```
class staticBlock {

  static int id;
  static String name;

  static{  // static block
   id = 101;
   name = "Hamim";
  }

  static void display(){
   System.out.println("id : "+id);
   System.out.println("Name : "+name);
  }
}
```

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

        staticBlock.display();
    }
}
```

Output:
id : 101
Name : Hamim

- **What is the output of the following Java program fragment:**

```
class staticBlock {

   static int id;
   static String name;

   static{  // static block
    id = 101;
    name = "Hamim";
   }

   static void display(){
    System.out.println("id : "+id);
    System.out.println("Name : "+name);
   }

   public static void main(String[] args) {
```

```
    staticBlock.display();
  }
}
```

Output:
id : 101
Name : Hamim

- **What is the output of the following Java program fragment:**

```
class staticBlock {

   static int id;
   static String name;

   static{  // static block
    System.out.println("I am in static block");
   }

   public static void main(String[] args) {
       System.out.println("I am in main method");
   }
 }
```

Output:
I am in static block
I am in main method

10. Wrapper class

A wrapper class is a class that encapsulates a primitive data type and provides methods and utility functions for working with that data type as an object. Wrapper classes are part of the Java standard library and are used to convert primitive data types into objects, which can be helpful in various situations, such as when you need to work with collections that require objects rather than primitives.

Wrapper classes are used to convert primitive data type into object and object into primitive data type.

Here are the wrapper classes for the primitive data types in Java

1. Integer: Wraps int.
2. Long: Wraps long.
3. Short: Wraps short.
4. Byte: Wraps byte.
5. Character: Wraps char.
6. Float: Wraps float.
7. Double: Wraps double.
8. Boolean: Wraps boolean.

primitive	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

- Autoboxing = converting primitive to object
- Unboxing = converting object to primitive

- **Make a java program that will swap two integer numbers :**

```
public class Test {
  public static void main(String[] args) {
    int a = 10;
    int b = 20;

    System.out.println("Before swap: a = "+a+" b =
"+b);

    swap(a, b);
    //After swap a and b
    System.out.println("After swap: a = "+a+" b =
"+b);
  }

  // call-by-value
  static void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
  }
}
```

Output:
Before swap: a = 10 b = 20
After swap: a = 10 b = 20

Here,
We can see that the values of the a and b variables aren’t swapped. The reason is that variables a and b are primitive that's means they will be saved in the stack memory. Java’s primitive variable only allows call-by-value not call-by-reference because Java doesn’t have pointer to point any memory address like C/C++.

Using Wrapper class:

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to object
        int x = 30;
        Integer y = Integer.valueOf(x);
        System.out.println("y = "+y);
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to object
        int x = 30;
        Integer y = x;//Integer.valueOf(x) ->
autoboxing
        System.out.println("y = "+y);
    }
}
```

Output:

```
y = 30
```

Here,
We have converted primitive data type into object (Autoboxing).

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //object to primitiveb data type
        Double d = new Double(10.25);
        System.out.println("d = "+d);
        double e = d.doubleValue();
        System.out.println("e = "+e);
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //object to primitiveb data type
        Double d = new Double(10.25);
        System.out.println("d = "+d);
        double e = d; //d.doubleValue()
        System.out.println("e = "+e);
    }
}
```

Output:
d = 10.25
e = 10.25

Here,
We have converted object into primitive data type (Unboxing).

Conversion between String and Primitive Data type

- **Converting primitive to String data type :**

- **Write a program that will convert a primitive data type into String data type..**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to String data type
        int i = 100;
        String s = Integer.toString(i);
        System.out.println("s = "+s);
    }
}
```

Output:
s = 100

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to String data type
        double i = 100;
```

```
        String s = Double.toString(i);
        System.out.println("s = "+s);
    }
}
```

Output:
s = 100.0

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to String data type
        boolean i = true;
        String s = Boolean.toString(i);
        System.out.println("s = "+s);
    }
}
```

Output:
s = true

- **Converting String to Primitive data type :**

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //String to primitive data type
        String s = "32";
        int i = Integer.parseInt(s);
        System.out.println("i  = "+i);
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //String primitive data type
        String s = "32";
        int i = Integer.valueOf(s);
        System.out.println("i  = "+i);
    }
}
```

Output:
i = 32

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //String to primitive data type
        String s = "32";
        double i = Double.parseDouble(s);
        System.out.println("i  = "+i);
    }
}
```

Output:
i = 32.0

- **Write a program that will convert decimal To binary, octal, and hexadecimal.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int decimal = 15;
        String binary =
Integer.toBinaryString(decimal);
        System.out.println("Binary form of 15 =
"+binary);
        String octal = Integer.toOctalString(decimal);
        System.out.println("Octal form of 15 =
"+octal);
        String hexa = Integer.toHexString(decimal);
        System.out.println("Hexadecimal form of 15
= "+hexa);
    }
}
```

Output:
Binary form of 15 = 1111
Octal form of 15 = 17
Hexadecimal form of 15 = f

- **Write a program that will convert binary, octal, and hexadecimal to decimal.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String binary = "1010";
      Integer decimal = Integer.parseInt(binary, 2);
      System.out.println("Decimal = "+decimal);
      String octal = "675";
      decimal = Integer.parseInt(octal, 8);
      System.out.println("Octal = "+decimal);
      String hexa = "A";
      decimal = Integer.parseInt(hexa, 16);
      System.out.println("Hexadecimal =
"+decimal);
   }
}
```

Output:
Decimal = 10
Octal = 445
Hexadecimal = 10

Here,
The first parameter of parseInt( ) method take the binary number as a String and second parameter is the base number <parseInt(binary, 2)>.

- **Write a program that will show date.**

```
package basicjava;
import java.util.Date;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Date date = new Date();
        System.out.println("Date is : "+date);
    }
}
```

Output:

```
ate is : Sun Jan 29 04:05:08 BDT 2023
```

or,

```
package basicjava;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Date date = new Date();
        DateFormat dateFormat = new
SimpleDateFormat("dd/MM/YYYY");
        String currentDate = dateFormat.format(date);
        System.out.println("Current date : "+currentDate);
    }
}
```

Output:
Current date : 29/01/2023

Here,
Date is a class and we declare date as an object/variavle of the Date class.
We also declare dataFormat variable of the DateFormat class for formatting the date.

- **Write a program that will show time.**

```
package basicjava;
import java.time.LocalTime;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        LocalTime time = LocalTime.now();
        System.out.println("Time is : "+time);
    }
}
```

Output:
Time is : 04:29:13.131902400

or,

```
package basicjava;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        LocalTime time = LocalTime.now();
        DateTimeFormatter fomatter =
DateTimeFormatter.ofPattern("hh:mm:ss");
        String formattedTime = time.format(fomatter);
        System.out.println("Time is : "+formattedTime);
    }
}
```

Output:

```
Time is : 04:34:11
```

- **Write a program that will create random number 0 to 9.**

```
package basicjava;
import java.util.Random;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Random rand = new Random();
        int randomNumber = rand.nextInt(10); // 0 to 9
        System.out.println("Random number =
"+randomNumber);
    }
}
```

Output:
Random number = 5

Here,
We have used Random class to create random number.

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
```

```
int randomNumber = (int) (Math.random()*10); // 0 to 9
        System.out.println("Random number = "+randomNumber);
    }
}
```

Output:
Random number = 7

Here,
We have used random method of Math class to create random number.

- **Write a program that will create random number from 1 to 10.**

```
package basicjava;
import java.util.Random;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      Random rand = new Random();
      int randomNumber = rand.nextInt(10) + 1; // 1 to
10
      System.out.println("Random number =
"+randomNumber);
   }
}
```

Output:
Random number = 1

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int randomNumber = (int) (Math.random()*10) +
1; // 1 to 10
      System.out.println("Random number =
"+randomNumber);
   }
}
```

Output:
Random number = 3

11. file of Java

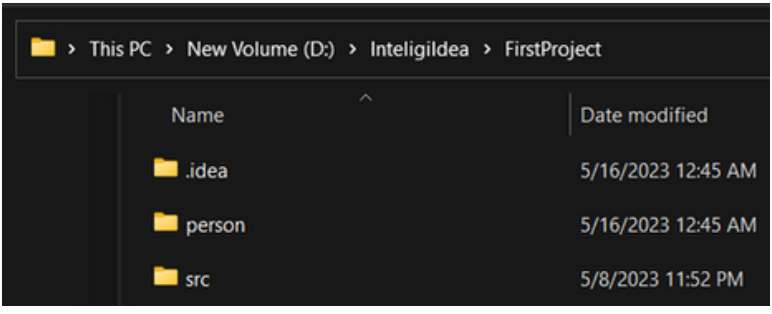
In Java, with the help of File Class, we can work with files. This File Class is inside the java.io package. The File class can be used by creating an object of the class and then specifying the name of the file.

- **Creating file :**

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
    public static void main(String[] args) {
        File dir = new File ("person");
        dir.mkdir(); // directory will be created
    }
}
```

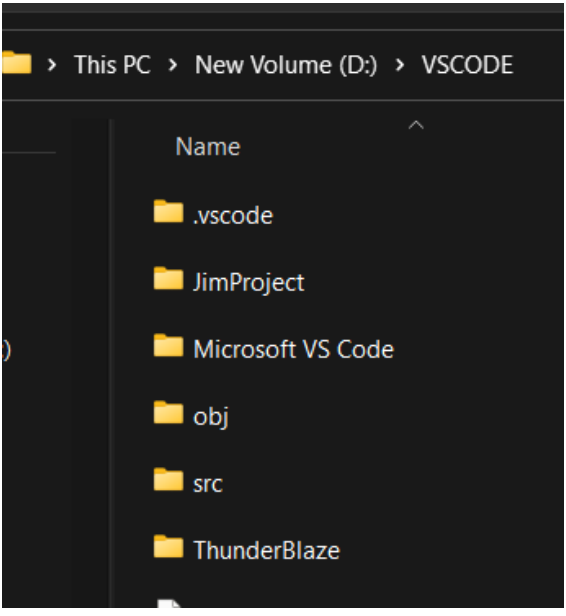
Output:



- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
   public static void main(String[] args) {
      File dir = new File
("D:\\VSCODE\\ThunderBlaze");
      dir.mkdir(); // directory will be created
   }
}
```

Output:



- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
   public static void main(String[] args) {
      File dir = new File
("D:\\VSCODE\\ThunderBlaze");
      dir.mkdir(); // directory will be created

      String dirLocation = dir.getAbsolutePath();
      System.out.println(dirLocation);
      System.out.println(dir.getName());
   }
}
```

Output:
D:\VSCODE\ThunderBlaze
ThunderBlaze

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
   public static void main(String[] args) {
      File dir = new File
("D:\\VSCODE\\ThunderBlaze");
      dir.mkdir(); // directory will be created

      String dirLocation = dir.getAbsolutePath();
      System.out.println(dirLocation);
      System.out.println(dir.getName());



      System.out.println();
      if(dir.delete()){
         System.out.println(dir.getName()+" folder
has been deleted");
      }
   }
}
```

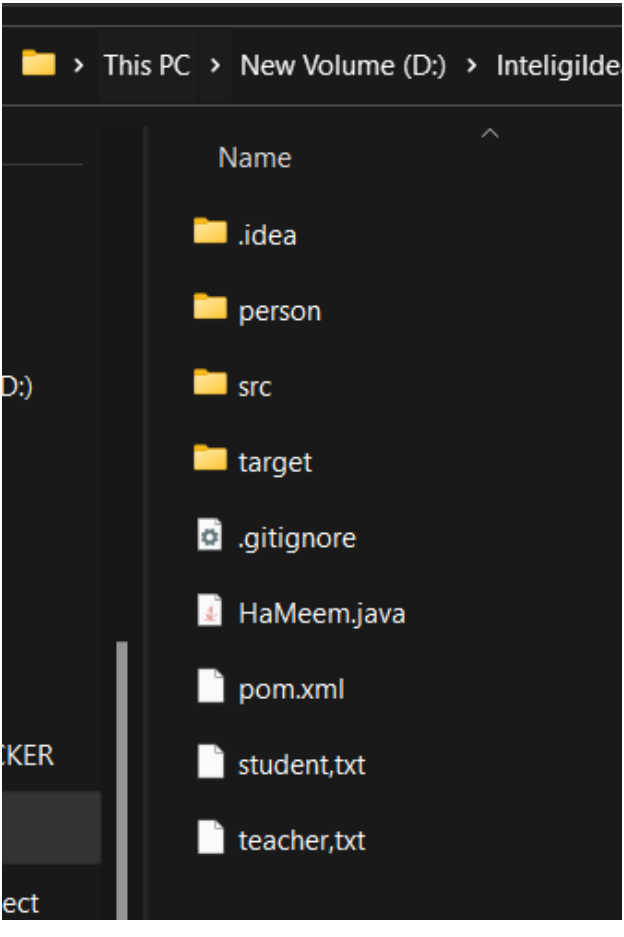
Output:

```
D:\VSCODE\ThunderBlaze
ThunderBlaze

ThunderBlaze folder has been deleted
```

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
   public static void main(String[] args) {
      File dir = new File ("person");
      dir.mkdir(); // directory will be created

      File file1 = new File("student,txt");
      File file2 = new File("teacher,txt");

      try{
         file1.createNewFile();
         file2.createNewFile();
         System.out.println("Files are created");

      }catch (Exception a){
         System.out.println(a);
      }
   }
}
```

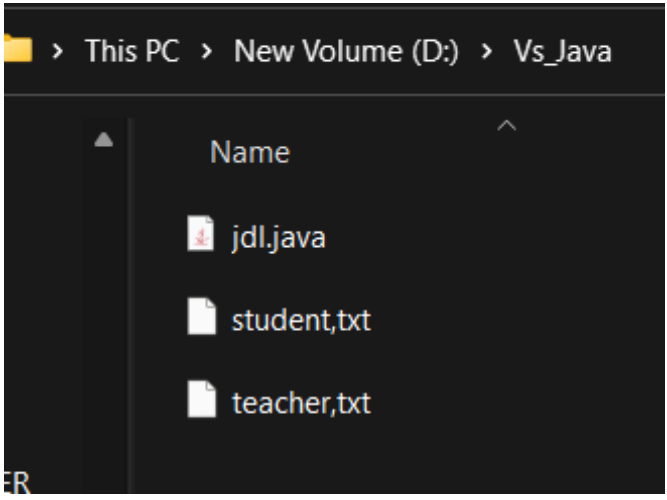
Output:
Files are created



- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
   public static void main(String[] args) {
      File dir = new File ("person");
      dir.mkdir(); // directory will be created

      File file1 = new File("D:\\Vs_Java\\student,txt");
      File file2 = new File("D:\\Vs_Java\\teacher,txt");

      try{
         file1.createNewFile();
         file2.createNewFile();
         System.out.println("Files are created");

      }catch (Exception a){
         System.out.println(a);
      }
   }
}
```

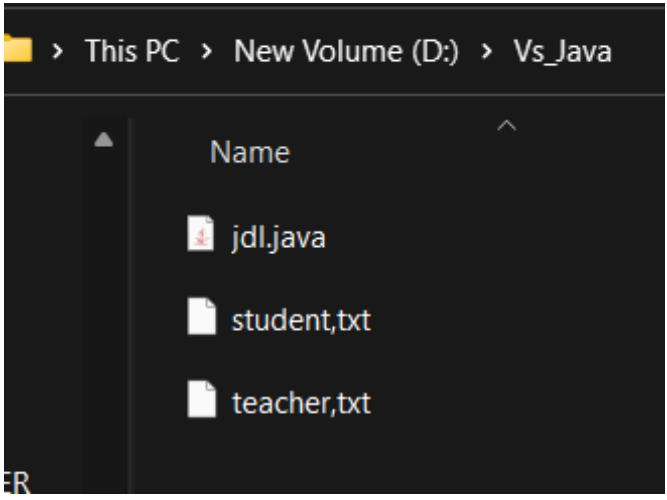
Output:
Files are created



- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
   public static void main(String[] args) {
      File dir = new File ("person");
      dir.mkdir(); // directory will be created

      String path =  dir.getAbsolutePath();
      File file1 = new File(path+"student,txt");
      File file2 = new File(path+"teacher,txt");

      try{
         file1.createNewFile();
         file2.createNewFile();
         System.out.println("Files are created");

      }catch (Exception a){
         System.out.println(a);
      }
   }
}
```

Output:
Files are created



- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;

public class FileDemo {
   public static void main(String[] args) {
      File dir = new File ("person");
      dir.mkdir(); // directory will be created

      String path =  dir.getAbsolutePath();
      File file1 = new File(path+"student,txt");
      File file2 = new File(path+"teacher,txt");

      try{
         file1.createNewFile();
         file2.createNewFile();
         System.out.println("Files are created");

      }catch (Exception a){
         System.out.println(a);
      }

      System.out.println();

      file2.delete();
```

```
        if(file1.exists()){
            System.out.println("File1 exists");
        }

        if(file2.exists()){
            System.out.println("File2 exists");
        }
        else
            System.out.println("File2 not exists");
    }
}
```

Output:

Files are created

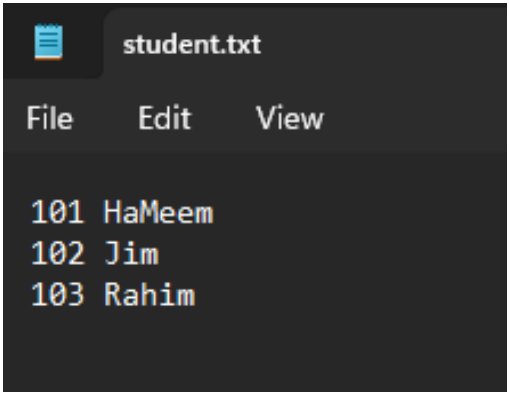
File1 exists
File2 not exists

- **write into a File :**

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Formatter;

public class FileDemo {
   public static void main(String[] args) {
      try{
         Formatter formatter = new
Formatter("D:\\Vs_Java\\student.txt");
         formatter.format("%s
%s\r\n","101","HaMeem");
         formatter.format("%s %s\r\n","102","Jim");
         formatter.format("%s
%s\r\n","103","Rahim");

         System.out.println("File create and write
successfully done.");
         formatter.close();
      }catch (FileNotFoundException a){
         System.out.println(a);
      }

   }
}
```

Output:
File create and write successfully done.



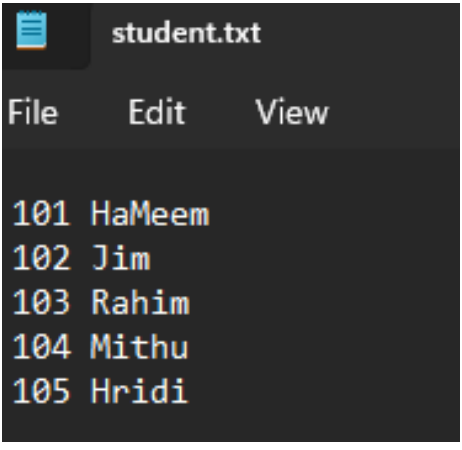
- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Formatter;
import java.util.Scanner;

public class FileDemo {
   public static void main(String[] args) {
      String id, name;
      try{
         Formatter formatter = new
Formatter("D:\\Vs_Java\\student.txt");

         Scanner scan = new Scanner(System.in);
         System.out.print("How many students : ");
         int num = scan.nextInt();
         for (int i=1 ; i<=num;i++){
            System.out.print("Enter student id and
name : ");
            id = scan.next();
            name = scan.next();
            formatter.format("%s %s\r\n",id,name);

         }
```

```
            System.out.println("File create and write
successfully done.");
            formatter.close();
        }catch (FileNotFoundException a){
            System.out.println(a);
        }


    }
}
```

Output:

```
How many students : 5
Enter student id and name : 101 HaMeem
Enter student id and name : 102 Jim
Enter student id and name : 103 Rahim
Enter student id and name : 104 Mithu
Enter student id and name : 105 Hridi
File create and write successfully done.
```



- **read a File :**

- **What is the output of the following Java program fragment:**

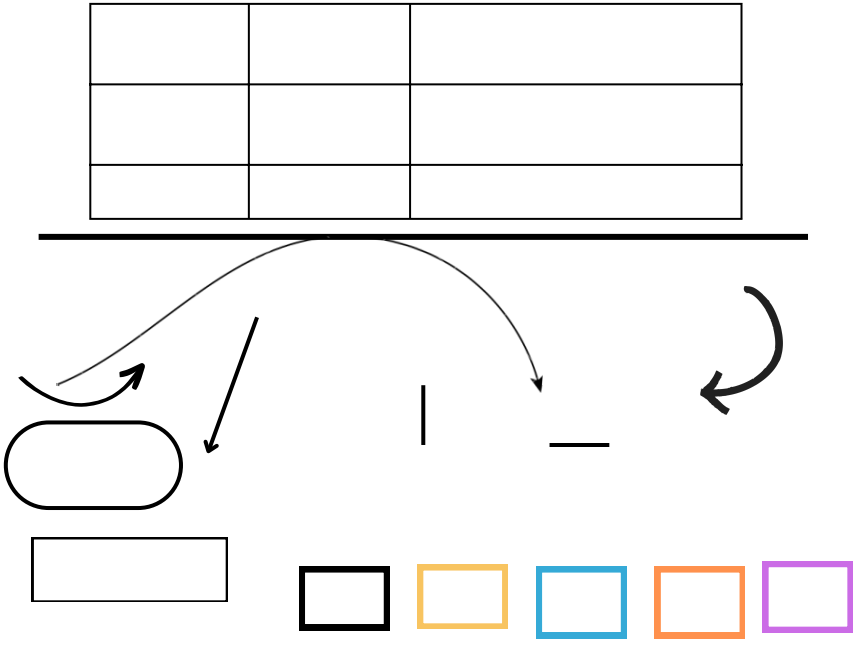
```
package FirstPackage;

import java.io.File;
import java.util.Scanner;

public class FileDemo {
   public static void main(String[] args) {
      try{
         File file = new File("D:\\Vs_Java\\student.txt");
         Scanner scan = new Scanner(file);
         while (scan.hasNext()){
            String id = scan.next();
            String name = scan.next();

            System.out.println(id+" "+name);
         }

         System.out.println("File read successfully done.");
         scan.close();

      }catch (Exception a){
         System.out.println(a);
      }


   }
}
```

Output:
101 HaMeem
102 Jim
103 Rahim
104 Mithu
105 Hridi

Here,
hasNext() method read the line rowoise and take the words one by one.

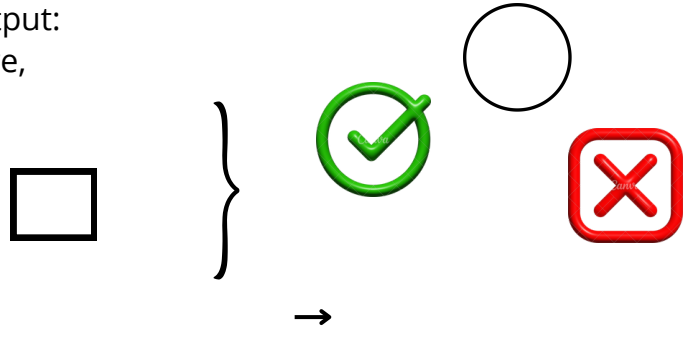
Operator Precedence and Associativity:

- **Make a program that will show String palindrome.**



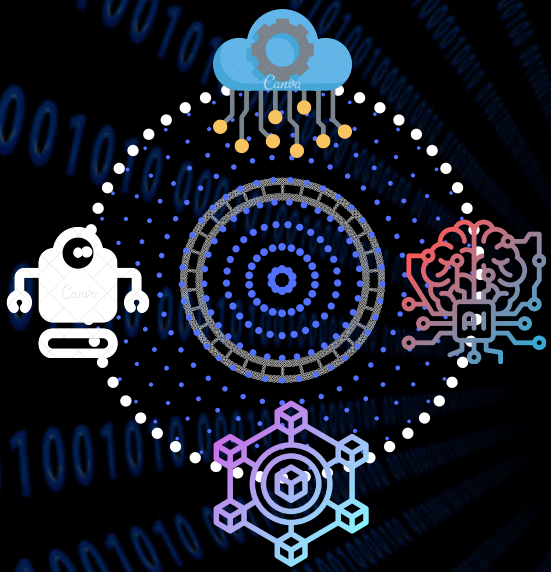
Output:
Here,

Jim



- **What is the output of the following Java program fragment:**

Operator



JAVA LANGUAGE (FIRST PART)

T.I.M. HAMIM

ABC PROKASHONI