Alpha 5.0 - Apna College Java

11. Creating a Java File. →

VS Code → Code Editor (IDE)
place where we write code.

★ Intellij, VS Code, Eclipse are popular Code Editor and Free of Cost

★ Online Code Editors are also available (OnlineGDB)

Java File. → .java → for Java File
[.java] Extension of Java Program

12. Broler Plate Code of Java →

```
class ClassName {
public static void main(String[] args) {
        ;
}
}
```

★ First in Java File, we define a class and write all Code inside that class
★ Class is a user defined datatype that contain data members and member functions. • Blueprint of Object

Class [Data member] + [Member Functions]

All Code of Java is written in a class

```
public class ClassName{
    public static void main(String[] args){
    //This is main method.
    }
}
```

★ Java public class name and File name should be same. ★

main() method executed by Java
It is entry point of Java Program.
All command starts running from main() method.

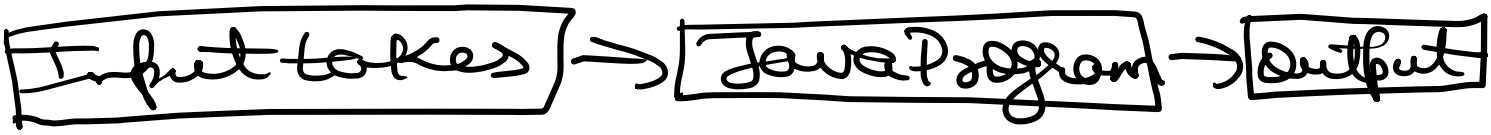
Boiler Plate Code is like a template that is mandatory to be written

13 Output in Java →
Output on Screen →
System.out.println("Hello"); (Print with Next line)
method of System.out

System.out.print("Hello"); (method with no next line)

; → Terminator to end a statement
statement terminator

Java Program →



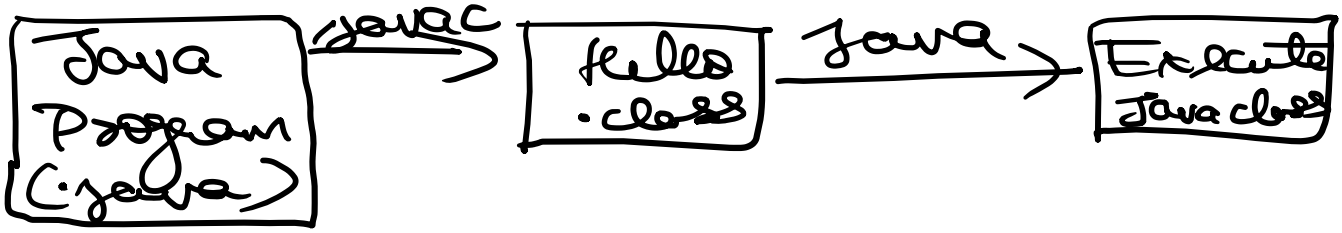
Input Processing by Java Prog. Output

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

To Compile →

javac Hello.java (Java Compiler)

java Hello



System.out.println("——");

(For new Line)

System.out.print("——");

(Only Print no Next Line)

; Terminator for Statements (ends a statement)

System.out.println("——————");
Next line print

\n → Next line Escape sequence character

Output in Java →

1. System.out.println("—");
2. System.out.print("——");
3. \n → Escape sequence for Next line

★ sout("Things in Quotes will be printed as it is");

14 Printing Pattern in Java

```
* * * *
* * *
* *
*
```

```
public class Pattern {
public static void main(String args[])
{
    System.out.println("* * * *");
                       ("* * *");
                       ("* *");
                       ("*");
}
}
```

15 Variables in Java →

Variable is Named Memory Location

It's like a container that contain values stored in computer memory It's value can be changed.

It's like Name assigned to a memory location where we can stored data.

a=10, b=5 ← 2*(a+b)

Constant/Literal Variables

2 Units in Java →

① Literals or Constants →

2, 4, 6, 8 — Numeric literals

'a', 'b', 'c', '@', '*' — character literals

② Variables — Named Memory Location (can be changed)

Tokens in Java →

K → Keyword (Reserved words)
I — Identifier (Name of class or variable)
L — Literal / Constants
P — Punctuators / Separators
O — Operators (Special Symbol)

Tokens in Java is KILPO

Variable Declaration and assignment →

int a = 5, b = 10; // Memory Reserved Block

a and b are variables of int datatype

★ a and b declared and initialized with value

★ In Double Quotes " ——— " whatever is written printed as it is ★

Datatypes in Java →

Primitive Datatype (Coded by Developer / Pre made DT)	Non Primitive Datatype (User Created) (Referenced Types)
→ byte	String
→ short	Array
→ char	class
→ boolean	Object
→ int	Interface
→ long	
→ float	
→ double	

Java Typed language (Strict lang and very Safe language)

byte range from (−128 to 127) (8 Bits)

char takes single character (1 Byte)

boolean true, false (1 Bit)

float float price = 10.65;
int val = 25 (Whole Number)
int used to store whole values

long is used to store values that are bigger then int without decimal

double is used to store values greater than range of float.

(short 2Bytes whole No. values)

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Why Different Datatypes →
① Due to their different ranges and sizes and memory allocated according to their ranges and sizes.

Datatype tell →
① Type of Data (Character, String, Number, Real No. etc)
② Range and Size of Data

Sum of a & b →

```java
public class SumofAB {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        System.out.println("Addition of Two Numbers = "+(a+b));
    }
}
```

Comments in Java →

★ ① Comments in Java for User understanding
★ ② For making code unexecutable

// Single Line Comment
/* ———— */ multiline Comment

Input in Java :→

Import Package →

```
import java.util.Scanner;

Scanner input = new Scanner
(inputobject of Scanner class) (System.in);
int num1 = input.nextInt();
```

Output
Variables / Datatypes

Input in Java →

① Import Scanner class →

```
import java.util.Scanner;
```

② Create object of Scanner class inside main() method →

```
Scanner input = new Scanner
                (System.in);
```

③ Use Scanner Object →

```
int a = input.nextInt()
```

Types of next →

1. next();
2. nextLine();
3. nextInt();
4. nextByte();
5. nextFloat();
6. nextDouble();
7. nextBoolean();
8. nextShort()
9. nextLong()

WholeNumber →
byte (1Byte), short (2Byte)
int (4Byte), long (8Byte)

Decimal Numbers →
Float (4Bytes), Double (8Bytes)
Boolean (true, false) (1Bit)
Char (2Bytes)

Whole No →
nextByte(); | nextShort();
nextInt(); | nextLong();

Decimal No →
nextFloat(), nextDouble();

Bool →
nextBoolean();

String →
next(); | nextLine();
Discards characters after white spaces | Entertain string after white spaces

Input Sum of Two No using Scanner Class

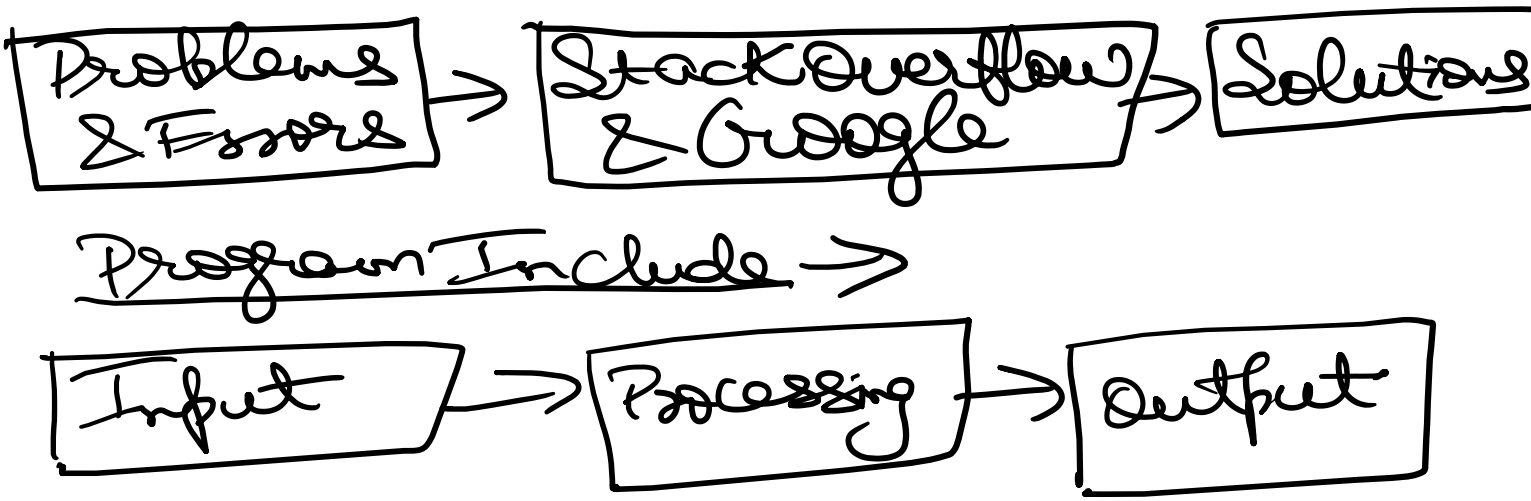
```java
import java.util.Scanner;

public class SumofABInput {
    public static void main(String[] args) {
        int num1, num2, sum;
        System.out.println("Enter the Value of Two Numbers = ");
        Scanner input = new Scanner(System.in);

        // input
        System.out.print("Enter the Value of Number 1 = ");
        num1 = input.nextInt();
        System.out.print("Enter the Value of Number 2 = ");
        num2 = input.nextInt();

        // sum
        sum = num1 + num2;
        System.out.println("Sum of Two Numbers = "+sum);

        // input close
        input.close();
    }
}
```



Product of a and b →

a * b

Scanner →

Input a & b

int product = a * b

Print product

```java
import java.util.Scanner;

class Product {
    public static void main(String[] args) {
        int num1, num2, product;
        System.out.println("Enter the Value of Two Numbers = ");
        Scanner input = new Scanner(System.in);

        // input
        System.out.print("Enter the Value of Number 1 = ");
        num1 = input.nextInt();
        System.out.print("Enter the Value of Number 2 = ");
        num2 = input.nextInt();

        // sum
        product = num1 * num2;
        System.out.println("Product of Two Numbers = "+product);

        // input close
        input.close();
    }
}
```

Area of Circle →

Input Radius

Area = Radius^2 * PI

Print Area

Java Program for Area of Circle →

```java
import java.util.*;

class AreaofCircle {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter Radius of Circle = ");
        double radius = input.nextFloat();
        double Area = Math.PI * Math.pow(radius, 2);
        System.out.println("Pi is = "+Math.PI);
        System.out.println("Area of Circle is = "+Area);
        input.close();
    }
}
```

** 3.14 f → float Value otherwise Java consider it double value

Type Conversion in Java →

One type → other type

Conversion Happens →

① type Compatibility

② destination type > source type

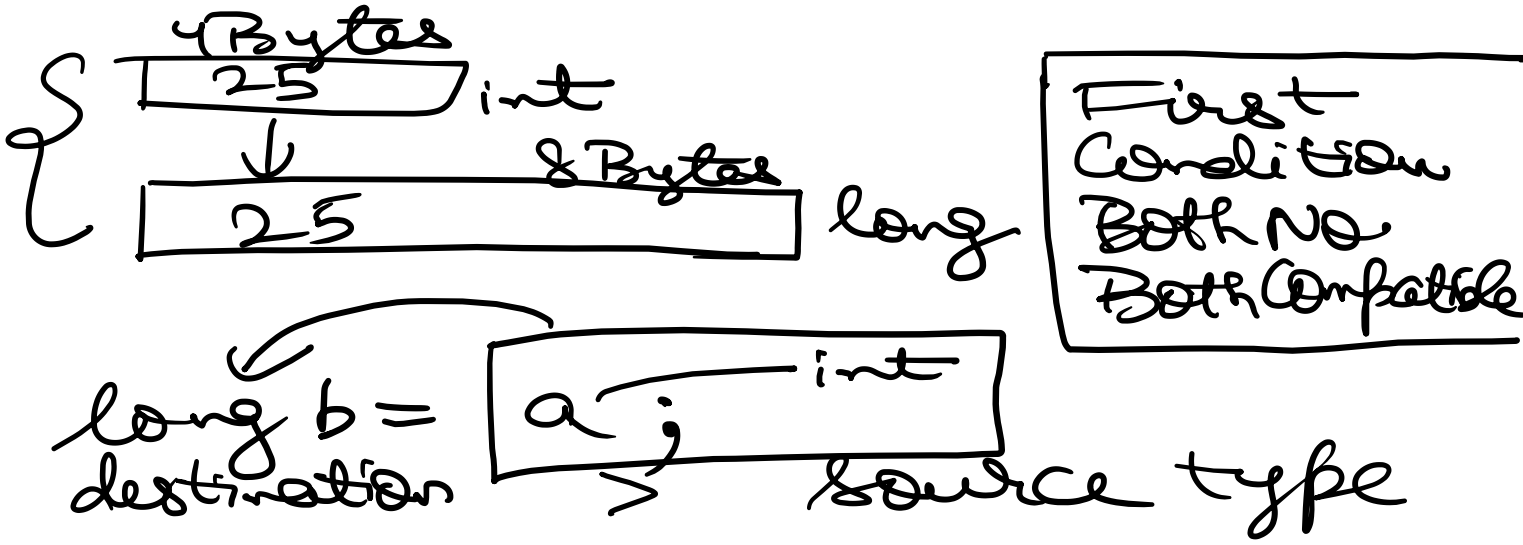
* byte > short > int > float > long > double

```
int a = 25
long b = a;
sout (b);
```

1st Condition fulfilled →

① Due to int and long are Compatible types

2nd also possible →
destination type long (8 Bytes) >
greater than int (4 Bytes).



long → int type Conversion
is Lossy type Conversion

Data can be loosed

★ Don't done automatically by Java ★

Type Conversion →
Types →
Implicit Type Conversion →
It is done by Java automatically

★ byte → short → int → float →
long → double ★

byte data can be stored by any further type

long data can only be stored by double only

If we go in backward type then it lead to loss of data (lossy type conversion)

★ If we try to store long data to int it will lead loss of data ★

★ Type Conversion in Java also said as widening Conversion and Implicit conversion.

★ → In it value of small datatype is stored in big and compatible datatype.

Example of Type Conversion or Implicit Type Conversion or Widening

float num = input.nextInt();
Scanner class object
Sout (num) //print with decimal

allowed in Java int → float
Implicit Type Conv or Type Conversion

Type Casting → Explicit Type Conversion
not allowed by Java but
done by User

```
float a = 25.12;
int b = (int) a;
```

It leads to loss of data

```
float a = 25.12;
int b = a;
```

Error → Possibly lossy conversion from float to int

Type Conversion → Code Java

```
import java.util.Scanner;

class TypeConversion {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the Value of Number = ");
        float n1 = input.nextInt();
        // allowed in java, int -> float implicit type conversion
        System.out.println("Number is = "+n1);
        input.close();
    }
}
```

Type Casting is automatically done by Java
It is said as →
① Widening Conversion
② Implicit Type Conversion

Type Casting Code →

```java
import java.util.Scanner;

class TypeCasting {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the Value of Double Number = ");
        double d1 = input.nextDouble();
        int i1 = (int) d1;
        System.out.println("Double to int Type Casting is = "+i1);
        input.close();
    }
}
```

Type Casting → done by Programmer explicitly lead to loose of data.

Also called as →
→ Narrow type Conversion
→ Explicit Conversion

Leads to loose of data.

★

```
char ch = 'a';
int no = ch;
sout(no); // Prints 97
```

★

Value of ASCII value of 'a' is 97

Type Promotion in Expression →

(1) Java automatically promotes each byte, short or char operand to int when evaluating an expression.

(2) If one operand is long, float or double the whole expression is promoted to long, float or double respectively.

☆ byte, short, char — Promote → int

Type Promotion only in case of Expressions & calculations

☆ In an Expression in Java, Largest Possible Container is Choosed ☆

Type Promotion in Exp Example →

Wrong

```
byte b = 5;
b = b * 2;
```

Error Lossy Conversion from int to byte

Right →

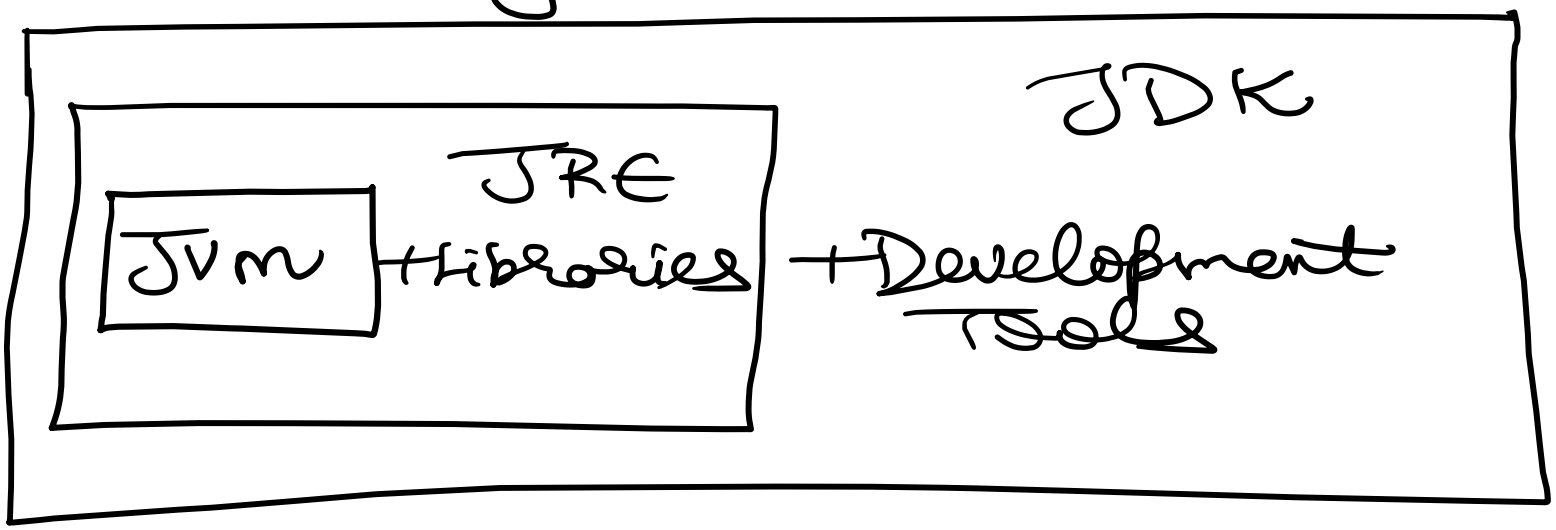
```
byte b = 5;
b = (byte)(b * 2);
```

Type Promotion Java Code →

```java
class TypePromotion {
    public static void main(String[] args) {
        char a = 'a';
        char b = 'b';
        System.out.println("a+b is "+(a+b));
        System.out.println("b-a is "+(b-a));
        // char c = a - b Error converting from char to int and storing in char
        short s = 5;
        byte b1 = 25;
        char A = 'A';
        // byte bt = s + b1 + A; Error because possible lossy conversions from int to byte
        byte bt = (byte)(s + b1 + A); //using type casting everything possible
        System.out.println(bt); // 5 + 25 + A(65) = 95

        // 2nd principle ->
        int int1 = 40;
        float flo1 = 128.193028f;
        long long1 = 9123;
        double double1 = 12908.123890;
        // int resulttest = int1 + flo1 + long1 + double1; error lossy conversion from
double to int
        double result = int1 + flo1 + long1 + double1;
        System.out.println(result);
    }
}
```

How Java Code is Executed or Running →



JDK → Java Development Kit

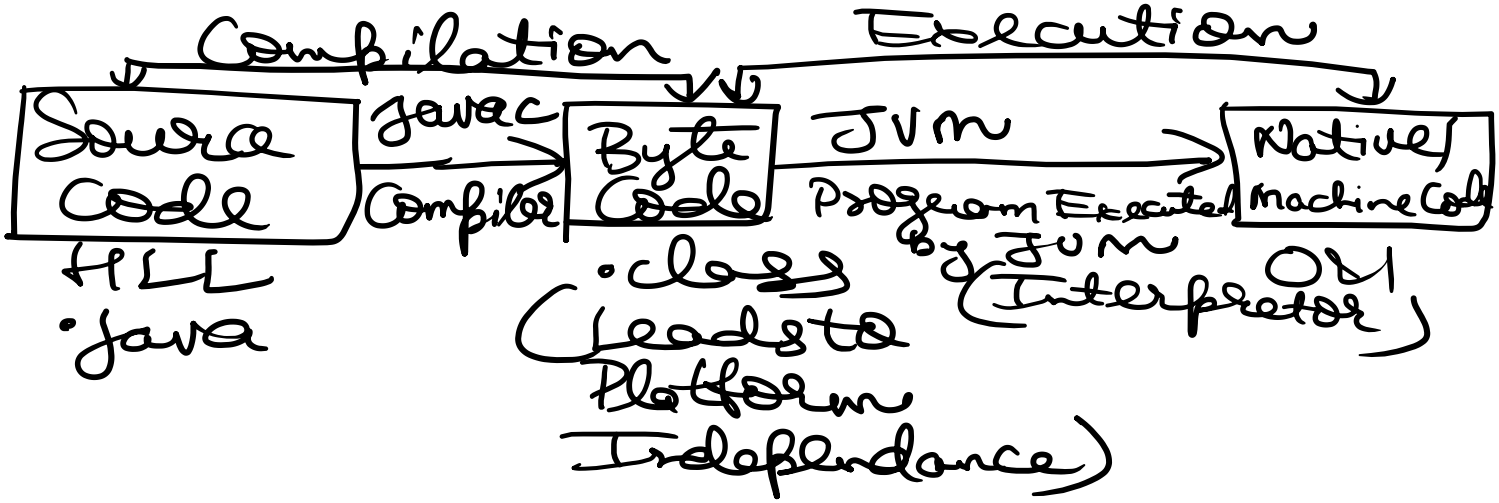
JRE → Java Runtime Environment

JVM → Java virtual machine (Code Executor)

JDK → JRE + Development Tools

JRE → JVM + Libraries

JVM → Code Executor



.class leads to Portability and Platform Independance