Java Fundamentals:

Data type: It Data types are containers for storing data values.In Java, there are different types of variables.

String - stores text, such as "Hello". String values are surrounded by double quotes.

int - stores integers (whole numbers), without decimals, such as 123 or -123.

float - stores floating point numbers, with decimals, such as 19.99 or -19.99.

char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes.

boolean - stores values with two states: true or false.

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	whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	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

For Example:

```
public class date {

   public static void main(String[] args)
 {
    byte myNum = 100;
    int Num = 9;
   float myFloatNum = 8.99f;
   char myLetter = 'A';
   boolean myBool = false;
```

```
    String myText = "Hello World";
    System.out.println("this is byte value "+myNum);
    System.out.println("this is integer value "+Num);
    System.out.println("this is float value " +myFloatNum);
    System.out.println("this is character value " +myLetter);
    System.out.println("this is boolean value " +myBool);
    System.out.println("this is String value " +myText);
}
}
```

Operators: Operators are used to perform operations on variables and values.Java divides the operators into the following groups:

Arithmetic operators : Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
_	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable	++ x
--	Decrement	Decreases the value of a variable	-- x

For Example:

```
public class arth {
    public static void main(String[] args) {
      int x = 20;
      int y = 30;
      int sum=0;
      int sub=0;
      int mul=0;
      int div,mod,increement = 0,decrement = 0;
      sum=x+y;
```

```
      sub=y-x;
      mul=x*y;
      div=x/y;
      mod=x%y;
      System.out.println(" the value of  x =" +x);
      System.out.println("the value of   y=" +y);
   System.out.println("Sum the value of x+y x + y=" +sum);
  System.out.println("Subtract the value of x - y=" +sub);
  System.out.println("Multiply in the two digits in x * y=" +mul);
  System.out.println("Divide the two digits is x / y=" +div);
  System.out.println("Modular between two digits in x % y=" +mod);
  System.out.println("In y is  value y is "+y);
  int incr = ++y;
  System.out.println("In  y  is  increase  then  print  value  y++  is
"+incr);
  System.out.println("In x is print the value is x="+x);
   decrement = --x;
  System.out.println("In  x  is  decrease  then  print  value  x--  is
="+decrement);
    }
  }
```

Assignment operators:-Assignment operators are used to assign values to variables.In the example below, we use the assignment operator (=) to assign the value 10 to a variable called x.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3

For Example:

```
public class ment
{
 public static void main(String[] args)
 {
      int x = 5;

      System.out.println("The value of x="+x);
      x += 3;
      System.out.println("The increase value of x is :"+x);
      x -= 4;
    System.out.println("The increase value of x is :"+x);
    x *= 3;
    System.out.println("The multiply value of x is :"+x);
    x /= 4;
    System.out.println("The divide value of x is :"+x);
    x %= 2;
    System.out.println("The modluse value of x is :"+x);
    x &= 11;
    System.out.println("The & sign use in program :"+x);
    }
  }
```

Comparison operators: Comparison operators are used to compare two values:

Operator	Example	Same As
==	Equal to	x ==y
!=	Not equal	x != y
>	Greater then	x> y
<	Less than	x < 3
>=	Greater than or equal to	x >= 3
<=	Less than or equal to	x <= 3

For Example:

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

```
{
   int x = 5;
 int y = 3;
System.out.println(x == y);
// returns false because 5 is not equal to 3
          System.out.println(x != y);
              //return true because 5 is not equal to 3.
          System.out.println(x > y);
// returns true because 5 is greater than 3
          System.out.println(x < y);
// returns false because 5 is less than 3

          System.out.println(x >= y);
    //returns true because 5 is greater, or equal, to 3
          System.out.println(x <= y);
//returns false because 5 is neither less than or equal to 3
        }
      }
```

Logical Operator: A logical operator is a symbol or word used to connect two or more expressions such that the value of the compound expression produced depends only on that of the original expressions and on the meaning of the operator. Common logical operators include AND, OR, and NOT.

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

For Example:

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

```
        int a=5;
        int b=10;
        System.out.println("Line 0 is valu of a is  "+a +" and b is
"+b);
        if((a>b)&&(a<b))
        {
            System.out.println("Line-1 condition is true");
        }
        else
        {
            System.out.println("Line-2 condition is  not true");
        }
        if((a==b)||(a<b))
        {
            System.out.println("Line-3 condition is true");
        }
        else
        {
            System.out.println("Line-4 condition is  not true");
        }

        if(a==b)
        {
            System.out.println("Line-5 condition is true");
        }
        else
        {
            System.out.println("Line-6 condition is  not true");
        }
        if(!(a==b))
        {
            System.out.println("Line-7 condition is true");
        }
        else
        {
            System.out.println("Line-8 condition is  not true");
```

```
        }
    }
}
```

Bitwise Operator: Bitwise Operators: The Bitwise Operator in C is a type of operator that operates on bit arrays, bit strings, and tweaking binary values with individual bits at the bit level. For handling electronics and IoT-related operations, programmers use bitwise operators. It can operate faster at a bit level.

For Example:

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

{
    int a=12;
    int b=25;
    int c=a&b;
    System.out.println("value of a-b in binary is " + c);
    c=a|b;
    System.out.println("value of a or b in binary in or operator is
" + c);
    c=a^b;
    System.out.println("value of and not operator in binary is " +
c);

}
}
```

Increment /Decrement operator: Java also provides increment and decrement operators: ++ and -- respectively. ++ increases the value of the operand by 1, while -- decrease it by 1. For example: Here, the value of num gets increased to 6 from its initial value of 5.

There are two types of Preefix and postfix operator.

Preefix operator: This is used in operator like as ++a (increment operator) and –a (decrement operator).

Postfix operator: This is used by the operator is like as a++ (increment operator) and a— (decrement operator).

For Example:

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

    // declare variables
    int a = 12, b = 18;
    int result1, result2;

    // original value
    System.out.println("Value of a: " + a);

    // increment operator
    result1 = ++a;
    System.out.println("After increment: " + result1);

    System.out.println("Value of b: " + b);

    // decrement operator
    result2 = --b;
    System.out.println("After decrement: " + result2);
System.out.println("--------------");
    // increment operator
    result1 = a++;
    System.out.println("After increment: " + result1);
    System.out.println("Again increment : "+ a);
    // decrement operator
    result2 = b--;
    System.out.println("After decrement: " + result2);
    System.out.println("Again decrement is :" +b);
  }
}
```

instanceof Operator: The instanceof operator checks whether an object is an instanceof a particular class.

For Example:

```
class inst {

        public static void main(String[] args) {

          String ptr="Hello world";
          boolean result;

          // checks if str is an instance of
          // the String class
          result = ptr instanceof String;
          System.out.println("Is str an object of String? "
+result);
        }
      }
```

Ternary Operator: The ternary operator (conditional operator) is shorthand for the if-then-else statement.

Syntax:

variable = Expression ? expression1: expression2.

For Example:

```
public class tany {

        public static void main(String[] args) {

          int februaryDays = 29;
          String result;

          // ternary operator
          result = (februaryDays == 28) ? " Leap year" : " Not a
leap year";
          System.out.println(result);
        }
      }
```

Misc Operators: misc operator is miscellaneous operator, which is conditional operator which has 3 operands, The first operand is always evaluated first. If nonzero, the second operand is evaluated, and that is the value of the result. Otherwise, the third operand is evaluated, and that is the value of the result.

For Example:

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

        System.out.println("Line - 1 Integer is "+ Integer.BYTES);
        System.out.println("char Line - 2" + Character.BYTES);

System.out.println(" Line - 3  long " + Long.BYTES);
System.out.println(" Line - 4 short " + Short.BYTES);
System.out.println("Line - 5  double " + Double.BYTES);
System.out.println(" Line - 6  float " + Float.BYTES);
System.out.println("Line - 7  byte is  " + Byte.BYTES);


    }
}
```

Control statements: A control statement in java is a statement that determines whether the other statements will be executed or not. It controls the flow of a program. An 'if' statement in java determines the sequence of execution between a set of two statements.

Simple if statement: The if statement determines whether a code should be executed based on the specified condition.

Syntax:

if (condition) {

Statement 1; //executed if condition is true

}

Statement 2; //executed irrespective of the condition.

If..else statement: In this statement, if the condition specified is true, the if block is executed. Otherwise, the else block is executed

For Example:

```
class ife
{
public static void main(String[] args)
{
int a = 15;
if (a > 20)
{
System.out.println("a is greater than 20");
}
else
{
System.out.println("a is less than 20");
}
}
}
```

Nested if statement: An if present inside an if block is known as a nested if block. It is similar to an if..else statement, except they are defined inside another if..else statement .

Syntax:

```
if (condition1)
{
Statement 1; //executed if first condition is true
if (condition2)
 {
Statement 2; //executed if second condition is true
}
else
{
Statement 3; //executed if second condition is false
}
}
```

For Example:

```
public class nest
{
public static void main(String[] args)
{
int s = 18;
if (s > 10)
{
if (s%2==0)
System.out.println("s is an even number and greater than 10!");
else
System.out.println("s is a odd number and greater than 10!");
}
else
{
System.out.println("s is less than 10");
}
}
}
```

Continue Statement: The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

For Example:

```
public class conn {
public static void main(String[] args) {
for (inti = 0; i< 8; i++) {
if (i == 5) {
continue;
      }
System.out.println("Value of i="+i);
    }
  }
}
```

Break Statement: The break statement in Java terminates the loop immediately, and the control of the program moves to the next statement following the loop.

For Example:

```
class Test
{
public static void main(String[] args)
 {
        // for loop
for (inti = 1; i<= 10; ++i) {

            // if the value of i is 5 the loop terminates
if (i == 5)
{
break;
            }
System.out.println("Value of i="+i);
        }
    }
}
```

Switch statement: A switch statement in java is used to execute a single statement from multiple conditions. The switch statement can be used with short, byte, int, long, types, etc.

Certain points must be noted while using the switch statement:

- One or N number of case values can be specified for a switch expression.
- Case values that are duplicate are not permissible. A compile-time error is generated by the compiler if unique values are not used.
- The case value must be literal or constant. Variables are not permissible.
- Usage of break statement is made to terminate the statement sequence. It is optional to use this statement. If this statement is not specified, the next case is executed.

For Example:

```
public class swith {
    public static void main(String[] args)
{
int days = 4;
String day;
```

```
// switch statement with int data type
switch (days) {
case 1:
day = "Monday";
break;
case 2:
day = "Tuesday";
break;
case 3:
day = "Wednesday";
break;
case 4:
day = "Thursday";
break;
case 5:
day = "Friday";
break;
case 6:
day= "Saturday";
break;
default:
day = "Sunday";
break;
}
System.out.println("Today is "+day);
}
}
```

Looping: Statements that execute a block of code repeatedly until a specified condition is met are known as looping statements. Java provides the user with three types of loops:

While: Known as the most common loop, the while loop evaluates a certain condition. If the condition is true, the code is executed. This process is continued until the specified condition turns out to be false.

The condition to be specified in the while loop must be a Boolean expression. An error will be generated if the type used is int or a string.

Syntax:

while (condition)

{

statementOne;

}

For Example:

```
public class whileTest
{
public static void main(String args[])
{
int i = 5;
while (i <= 15)
{
System.out.println(i);
i = i+2;
}
}
}
```

Do-while: The do-while loop is similar to the while loop, the only difference being that the condition in the do-while loop is evaluated after the execution of the loop body. This guarantees that the loop is executed at least once.

Syntax:

```
Do
{
//code to be executed
}
while(condition);
```

For Example:

```
public class Main
{
public static void main(String args[])
{
int i = 20;
do
```

```
{
System.out.println(i);
i = i+1;
} while (i <= 20);
}
}
```

Enhanced for loop: The enhanced for loop was introduced. This is mainly used to traverse collection of elements including arrays.

Syntax

for(declaration : expression)
{
  // Statements
}

Declaration: The newly declared block variable, is of a type compatible with the elements of the array you are accessing. The variable will be available within the for block and its value would be the same as the current array element.

Expression: This evaluates to the array you need to loop through. The expression can be an array variable or method call that returns an array.

For Example:

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

           for(int x : numbers )
            {
              System.out.print( x );
              System.out.print(",");
           }
           System.out.print("\n");
           String [] names = {"James", "Larry", "Tom", "Lacy"};
```

```
            for( String name : names )
            {
               System.out.print( name );
               System.out.print(",");
            }
         }
      }
```

Array: Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with square brackets:

For Example:

```
public class Main {
  public static void main(String[] args) {
    String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
    System.out.println(cars[1]);
  }
}
```

Declare an Array: An array as we declare it, meaning we assign values as when we create the array.

For Example:

```
class ArrayDemo {
    public static void main(String[] args) {
        // declares an array of integers
        int[] anArray;

        // allocates memory for 10 integers
        anArray = new int[10];

        // initialize first element
        anArray[0] = 100;
        // initialize second element
        anArray[1] = 200;
        // and so forth
        anArray[2] = 300;
```

```
        anArray[3] = 400;
        anArray[4] = 500;
        anArray[5] = 600;
        anArray[6] = 700;
        anArray[7] = 800;
        anArray[8] = 900;
        anArray[9] = 1000;

        System.out.println("Element at index 0: "
                           + anArray[0]);
        System.out.println("Element at index 1: "
                           + anArray[1]);
        System.out.println("Element at index 2: "
                           + anArray[2]);
        System.out.println("Element at index 3: "
                           + anArray[3]);
        System.out.println("Element at index 4: "
                           + anArray[4]);
        System.out.println("Element at index 5: "
                           + anArray[5]);
        System.out.println("Element at index 6: "
                           + anArray[6]);
        System.out.println("Element at index 7: "
                           + anArray[7]);
        System.out.println("Element at index 8: "
                           + anArray[8]);
        System.out.println("Element at index 9: "
                           + anArray[9]);
    }
}
```

Access the Elements of an Array (Initialize Array): we have not provided the size of the array. In this case, the Java compiler automatically specifies the size by counting the number of elements in the array.

For Example:

```
class Demo {
```

```
    public static void main(String[] args) {
int anArray [] = { 200 , 300 , 400 , 500 , 600 , 700 , 800 , 900 ,
1000 };
System.out.println("Element at index 9: "
                            + anArray[1]);
    }
}
```

Change an Array Element: To change the value of a specific element, refer to the index number:

Syntax:

cars[0] = "Opel";

For Example:

```
class change {
  public static void main(String[] args) {
    String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
    cars[0] = "Opel";
    System.out.println(cars[0]);
    System.out.println(cars[1]);
    System.out.println(cars[0]);
    System.out.println(cars[2]);
    System.out.println(cars[3]);
  }
}
```

Array Length: To find out how many elements an array has, use the length property:

For Example:

{

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars.length);
}
```

+

Loop Through an Array: You can loop through the array elements with the for loop, and use the length property to specify how many times the loop should run.

For Example:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

```
for (int i = 0; i < cars.length; i++) {
  System.out.println(cars[i]);
}
```

Loop through an Array with For-Each: There is also a "for-each" loop, which is used exclusively to loop through elements in arrays**:**

Syntax:

```
for (type variable : arrayname) {
  ...
}
```

For Example:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

Multidimensional Arrays: A multidimensional array is an array of arrays.

To create a two-dimensional array, add each array within its own set of curly braces.

For Example: (Print the length of an array in each rows)

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

        // create a 2d array
        int[][] a = {
            {1, 25, 3},
            {4, 5, 6, 9},
            {7},
        };

        // calculate the length of each row
        System.out.println("Length of row 1: " + a[0].length);
        System.out.println("Length of row 2: " + a[1].length);
        System.out.println("Length of row 3: " + a[2].length);
    }
}
```

Two Dimensional Array: Print all data in matrix form using 2D array.

For Example:

```
public class TwoDArray {
   public static void main(String[] args) {
      final int[][] matrix = {
         { 1, 2, 3 },
         { 4, 5, 6 },
         { 7, 8, 9 }
      };
      for (int i = 0; i < matrix.length; i++)
{ //this equals to the row in our matrix.
         for (int j = 0; j < matrix[i].length; j++)
{ //this equals to the column in each row.
            System.out.print(matrix[i][j] + " ");
         }
         System.out.println();
//change line on console as row comes to end in the matrix.
      }
   }
}
```

Two-D Array in Addition: In this program two matrix in perform Addition.

For Example:

```
class MultidimensionalArray1 {
    public static void main(String[] args) {

        int[][] a = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
               int[][] b={
               {2,4,3},
               {4,2,4},
               {5,7,7}        };
               int c[][]=new int [3][3];
```

```
                for(int i=0;i<3;i++)
                {
                 for(int j=0;j<3;j++)
                 {
                 c[i][j]=a[i][j]+b[i][j];
                    System.out.print(c[i][j]+" ");
                 }
                System.out.println();
            }
            }
    }
```

Variable Arguments: Varargs is a short name for variable arguments. In Java, an argument of a method can accept arbitrary number of values. This argument that can accept variable number of values is called varargs.

For Example : (in this program using non variable argument)

```
class var1
{
    public int sumNum(int a,int b)
    {
return a+b;
}
public int sumNum(int a,int b, int c)
{
    return a+b+c;
}
public static void main(String[] args)
{
    Var1 p1=new var1 ();
    System.out.println(p1.sumNum(3,7));
    System.out.println(p1.sumNum(2,4,6));
}
}
```

For Example: (In this program using variable argument)

```
class VarargExample {
```

```
    public int sumNumber(int ... args){
        System.out.println("argument length: " + args.length);
        int sum = 0;

        for(int x: args){
            sum += x;
        }
        return sum;
    }

    public static void main( String[] args ) {
        VarargExample ex = new VarargExample();

        int sum2 = ex.sumNumber(2, 4);
        System.out.println("sum2 = " + sum2);

        int sum3 = ex.sumNumber(1, 3, 5);
        System.out.println("sum3 = " + sum3);

        int sum4 = ex.sumNumber(1, 3, 5, 7);
        System.out.println("sum4 = " + sum4);
    }
}
```

ENUM: AJava Enumeration(Enums) defines a class type through which we can define a list of constants. By default, these constants are public, static and final.

- An enumeration type is created by using the enum keyword.
- Enumeration constants can only be accessed by using the dot operator with its enumeration type.
- An enumeration can be defined inside or outside a class.
- Enumeration in Java can have constructors, methods and instance variables.

For Example:

```
enum Cities{
Frankfurt, NewYork, Sydney, Tokyo           //enumeration constants
```

```
}

class A
{
public static void main(String... ar)
{
      //Creating a variable of Enum type, Cities and initializing it
to the enum constant, Sydney
      Cities c= Cities.NewYork;

      //Printing the value of Enum variable
      System.out.println(c);
}
}
```

Note: You can't use new keyword to create variables of an enum type

Program 2:

```
enum Cities{
Frankfurt, NewYork, Sydney, Tokyo           //enumeration constants
}
class A
{
public static void main(String... ar)
{
      //Creating a variable of Enum type by using the new keyword,
which is not allowed, hence it will lead to a compile error.
      Cities c= new Cities.Sydney;

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

Static import statement: By using static import statements in Java, we can access data members or member functions of a class directly without the use of a fully-qualified name.

Below is the implementation in which we are accessing the method under the class without any use of static import statement and by using fully-qualified name.

Syntax 1:

```
import package1[.package2].(*);
```

For Example: **(Program: 1)**

```
class geeks {
public static void main(String[] args)
    {
System.out.println(Math.sqrt(4));
System.out.println(Math.pow(2, 2));
System.out.println(Math.abs(6.3));
    }
}
```

Program: 2 (First program import to the second program)

```
import static java.lang.Math.*;
class Test2 {
public static void main(String[] args)
    {
System.out.println(sqrt(4));
System.out.println(pow(2, 2));
System.out.println(abs(6.3));
    }
}
```

Program: 3 (Final program)

```
import static java.lang.Math.*;
import static java.lang.System.*;
classGeekks {
public static void main(String[] args)
    {
        // We are calling static member of System class
        // directly without System class name
out.println(sqrt(4));
out.println(pow(2, 2));
out.println(abs(6.3));
```

```
    }
}
```

C-style formatted I/O: Formatted I/O functions are used to take various inputs from the user and display multiple outputs to the user. These types of I/O functions can help to display the output to the user in different formats using the format specifiers. These I/O supports all data types like int, float, char, and many more.

Sr. No.	Format Specifier	Type	Description
1	%d	int/signed int	used for I/O signed integer value
2	%c	char	Used for I/O character value
3	%f	float	Used for I/O decimal floating-point value
4	%s	string	Used for I/O string/group of characters
5	%ld	long int	Used for I/O long signed integer value
6	%u	unsigned int	Used for I/O unsigned integer value.
7	%i	unsigned int	used for the I/O integer value
8	%If	double	Used for I/O fractional or floating data
9	%n	prints	prints nothing

printf(): printf() function is used in a C program to display any value like float, integer, character, string, etc on the console screen. It is a pre-defined function that is already declared in the stdio.h(header file) .

For Example :

```
public class print {
public static void main(String[] args)
    {
int a=21;
float b=2.4f;
System.out.printf("%d",a);
System.out.printf("\n");
System.out.printf("%f",b);
    }
}
```

For Example:

```
public class scann {
public static void main(String[] args)
    {
int a=100;
System.out.printf("Printing simlpe integer a:"+"%d \n",a);
System.out.printf("Formatted     with"+"     "+"Precision     PI=%.2f
\n",Math.PI);
float n=5.2f;
System.out.printf("Formatted to"+" "+"specific width:n=%.4f \n",n);
    }
}
```

scanf(): scanf() function is used in the C program for reading or taking any value from the keyboard by the user, these values can be of any data type like integer, float, character, string, and many more. This function is declared in stdio.h(header file), that's why it is also a pre-defined function. In scanf() function we use &(address-of operator) which is used to store the variable value on the memory location of that variable.

For Example:

```
importjava.util.*;
class demo
{
public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

System.out.println("Enter the firts number=");
int a = sc.nextInt();
System.out.println("Enter the second number=");
int b = sc.nextInt();
System.out.println("Enter the third number=");
int c = sc.nextInt();
int d=a+b+c;
System.out.println("Total="+d);
System.out.println("Hello world");
    }
```

```
}
```

println(): println() method in Java is also used to display a text on the console. This text is passed as the parameter to this method in the form of String. This method prints the text on the console and the cursor remains at the start of the next line at the console.

For Example:

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

        // The cursor will after Hello
        // will at the start
        // of the next line
System.out.println("Hello1");

        // This will be printed at the
        // start of the next line
System.out.println("Hello World");
    }
}
```

Auto Boxing: Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this is called unboxing.

For Example:

```
public class main {
public static void main(String args[]) {
   Integer i = 60, io; // autobox an int
Int i = io; // auto-unbox
System.out.println(i + " " + io);
   io2 = io+(io/3);
System.out.println(i + " " + io2);
 }
}
```