**Basics of Java**

**Java why?**

**JRE, JVM, JDK**

Step1:

Install Eclipse.

Location:

<https://www.eclipse.org/downloads/packages/release/2025-06/r/eclipse-ide-java-developers>?

Click Download -> Select One time \_> eclipse will download

Extract the folder -> You can find eclipse ready

**(Optional)Run the installer** (eclipse-inst-jre-win64.exe) and select **Eclipse IDE for Java Developers** when prompted.

**(Optional)Install a 64-bit JDK**, if you haven't already, to ensure compatibility.

**Launch Eclipse**, choose or create a workspace, and you're ready to start coding!

**Class 3:**

**Basic Java Program Structure**

Here’s a simple program that prints "Hello World":

class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello World");

}

}

**Explanation:**

* class HelloWorld → Defines a class named HelloWorld.
* public static void main(String[] args) → Starting point of the program.
* System.out.println() → Prints output to the screen.

### Program 1: Print "Hello World"

class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello World");

}

}

### Program 2: Print your name

class PrintName {

public static void main(String[] args) {

System.out.println("My name is Suresh");

}

}

### Program 3: Print multiple lines

class PrintLines {

public static void main(String[] args) {

System.out.println("Java is powerful.");

System.out.println("It is platform independent.");

System.out.println("Let's start learning!");

}

}

**Java Data Types**

Java is **strongly typed**, meaning every variable must have a type.

**Primitive Data Types**

| **Data Type** | **Size** | **Example** |
| --- | --- | --- |
| byte | 1 byte | 127 |
| short | 2 bytes | 32767 |
| int | 4 bytes | 2147483647 |
| long | 8 bytes | 9223372036854775807 |
| float | 4 bytes | 3.14f |
| double | 8 bytes | 3.14159 |
| char | 2 bytes | 'A' |
| boolean | 1 bit | true/false |

**Non-Primitive Data Types**

* Strings
* Arrays
* Classes
* Interfaces

**Variables**

Variables store data.

### ****Types of Variables:****

1. **Local Variables** – Inside methods.
2. **Instance Variables** – Inside a class, but outside methods.
3. **Static Variables** – Shared among all objects.

**Example:**

int age = 25;

String name = "Suresh";

System.out.println("My name is " + name + " and I am " + age + " years old.");

### Program 1: Display different data types

class DataTypesDemo {

public static void main(String[] args) {

int age = 25;

float height = 5.9f;

char grade = 'A';

boolean isJavaFun = true;

String name = "Suresh";

System.out.println("Name: " + name);

System.out.println("Age: " + age);

System.out.println("Height: " + height);

System.out.println("Grade: " + grade);

System.out.println("Java Fun? " + isJavaFun);

}

}

### Program 2: Swap two numbers

class SwapNumbers {

public static void main(String[] args) {

int a = 10, b = 20;

System.out.println("Before swapping: a = " + a + ", b = " + b);

int temp = a;

a = b;

b = temp;

System.out.println("After swapping: a = " + a + ", b = " + b);

}

}

**Operators**

| **Type** | **Example** |
| --- | --- |
| Arithmetic | + - \* / % |
| Relational | > < >= <= == != |
| Logical | `&& |
| Assignment | = += -= \*= |
| Increment/Decrement | ++ -- |

**Example:**

int a = 5, b = 3;

System.out.println(a + b); // Output: 8

### Program 1: Arithmetic operations

class ArithmeticDemo {

public static void main(String[] args) {

int a = 15, b = 4;

System.out.println("Addition: " + (a + b));

System.out.println("Subtraction: " + (a - b));

System.out.println("Multiplication: " + (a \* b));

System.out.println("Division: " + (a / b));

System.out.println("Modulus: " + (a % b));

}

}

### Program 2: Relational operators

class RelationalDemo {

public static void main(String[] args) {

int a = 10, b = 20;

System.out.println(a == b); // false

System.out.println(a != b); // true

System.out.println(a > b); // false

System.out.println(a < b); // true

}

}

### Program 3: Logical operators

class LogicalDemo {

public static void main(String[] args) {

boolean x = true, y = false;

System.out.println("x && y: " + (x && y));

System.out.println("x || y: " + (x || y));

System.out.println("!x: " + (!x));

}

}

**Control Structures in Java**

Control structures in Java are used to control the **flow of execution** of a program. They allow you to make decisions, repeat tasks, and control how and when certain parts of the code run.

There are **three main types** of control structures in Java:

| **Control Structure Type** | **Examples** |
| --- | --- |
| Conditional Statements | if, if-else, if-else-if, switch |
| Looping Statements | for, while, do-while |
| Jump Statements | break, continue, return |

**1. Conditional (Decision-Making) Statements**

These are used to **make decisions** in the program based on conditions.

**a) if Statement**

Executes a block of code **only if** a given condition is true.

**Syntax:**

if (condition) {

// Code executes if condition is true

}

**Example:**

public class IfExample {

public static void main(String[] args) {

int age = 20;

if (age >= 18) {

System.out.println("You are eligible to vote.");

}

}

}

**b) if-else Statement**

Executes one block if the condition is true, otherwise executes another block.

**Syntax:**

if (condition) {

// Code if condition is true

// Code if condition is true

// Code if condition is true

}

else {

// Code if condition is false

}

**Example:**

public class IfElseExample {

public static void main(String[] args) {

int number = 5;

if (number % 2 == 0) {

System.out.println("Even number");

} else {

System.out.println("Odd number");

}

}

}

**c) if-else-if Ladder (Nested if)**

Used when there are **multiple conditions** to check.

**Syntax:**

if (condition1) {

// Code if condition1 is true

} else if (condition2) {

// Code if condition2 is true

} else {

// Code if none of the above is true

}

**Example:**

public class IfElseIfExample {

public static void main(String[] args) {

int marks = 85;

if (marks >= 90) {

System.out.println("Grade A+");

} else if (marks >= 75) {

System.out.println("Grade A");

} else {

System.out.println("Grade B");

}

int marks2 = 100;

if (marks2 >= 90) {

System.out.println("Grade A+");

} else if (marks2 >= 75) {

System.out.println("Grade A");

} else {

System.out.println("Grade B");

}

int marks3 = 70;

if (marks3 >= 90) {

System.out.println("Grade A+");

} else if (marks3 >= 75) {

System.out.println("Grade A");

} else {

System.out.println("Grade B");

}

}

}

**d) switch Statement**

Used to **replace multiple if-else statements** when comparing a single variable with multiple values.

**Syntax:**

switch (variable) {

case value1:

// Code block

break;

case value2:

// Code block

break;

default:

// Code if none match

}

**Example:**

public class SwitchExample {

public static void main(String[] args) {

int day = 3;

switch (day) {

case 1: System.out.println("Monday"); break;

case 2: System.out.println("Tuesday"); break;

case 3: System.out.println("Wednesday"); break;

default: System.out.println("Invalid day");

}

}

}

**2. Looping (Iteration) Statements**

Used to **repeat a block of code** multiple times.

**a) for Loop**

When the **number of iterations is known**.

**Syntax:**

for (initialization; condition; increment/decrement) {

// Code to be executed

}

**Example:**

public class ForLoopExample {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

System.out.println("Count: " + i);

}

}

}

Nested loop

Multiplication

public class ForLoopExample {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int i = 1; i <= 5; i++) {

System.out.print(i\*j + “\t”);

}

System.out.print(“”);

}

}

Pyramid:

public class ForLoopExample {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int i = 1; i <= i; i++) {

System.out.print(j + “\t”);

}

System.out.print(“”);

}

}

**b) while Loop**

Used when the **number of iterations is not known** in advance.

**Syntax:**

while (condition) {

// Code to be executed

}

**Example:**

public class WhileLoopExample {

public static void main(String[] args) {

int i = 1;

while (i <= 5) {

System.out.println("Count: " + i);

i++;

}

}

}

**c) do-while Loop**

Executes the code block **at least once**, and then repeats while the condition is true.

**Syntax:**

do {

// Code to be executed

} while (condition);

**Example:**

public class DoWhileExample {

public static void main(String[] args) {

int i = 1;

do {

System.out.println("Count: " + i);

i++;

} while (i <= 5);

}

}

**3. Jump Statements**

Used to **alter the normal flow of execution**.

**a) break**

* Exits the loop or switch immediately.

**Example:**

public class BreakExample {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

if (i == 3) {

break; // Exit loop when i is 3

}

System.out.println(i);

}

}

}

**b) continue**

* **Skips the current iteration** and moves to the next one.

**Example:**

public class ContinueExample {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

if (i == 3) {

continue; // Skip when i is 3

}

System.out.println(i);

}

}

}

**c) return**

* Exits from a method and optionally returns a value.

**Example:**

public class ReturnExample {

static int sum(int a, int b) {

return a + b;

}

public static void main(String[] args) {

System.out.println("Sum is: " + sum(5, 10));

}

}

**Arrays**

**What is Array?**

Array is the collection of similar data types

**1D:**

Example:

Int a[] = new int[5];

A[] -> reference of an array

Int[6] -> object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |

A 0 1 2 3 4

|  |
| --- |
|  |

Here we have to assign value,

A[0] = 1;

Every array has length(length is a property, not a method), to print the length,

S.o.p (a.length);

(Or)

Directly to mention the values,

Example:

Int a[] = {1,2,3,4,5,6}

|  |
| --- |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |

A

How to access?

s.o.p(a[0])

s.o.p(a[1])

**Example:**

public class arrayExample {

public static void main(String[] args) {

int a[] = {2,4,6,8,10}

for (int i = 0; i < a.length ; i++) {

System.out.println(a[i]);

}

}

}

public class arrayExample {

public static void main(String[] args) {

int a[] = {2,4,6,8,10}

for (int i = a.length -1; i >=0; i--) {

System.out.println(a[i]);

}

}

}

X -> x takes the value directly, a[i] means it traverse thr index’s.

Only forward direction

public class arrayExample {

public static void main(String[] args) {

int a[] = {2,4,6,8,10}

for (int x : a) {

System.out.println(x);

}

}

}

public class arrayPractise {

public static void main(String[] args) {

int a[] = new int[10];

int b[] = {1,2,3,4,5}

b[2] = 15;

int c[];

c= new int[10];

int[] d = {2,4,6,8,10}

int []e= {3,6,9,}

for (int i = 0; i <= a.length ; i++)

{

System.out.println(a[i]);🡪 default value will be o if not initialized

System.out.println(b[i]);

}

}

}

Program:

1. Find the sum of all the numbers
2. Searching element
3. Maximum element
4. Second largest element
5. copy array A to B

//program 1

public class arrayExample {

public static void main(String[] args) {

Int a[] = {3,9,7,8,12,6,15,5,4,10}

Int sum = 0;

For ( int i=0; i<=a.length; i++}

{

Sum = sum + a[i];

}

s.o.p(“sum is ”+ sum);

}

}

//program2

Int key = 12;

For ( int i=0; i<=a.length; i++}

{

If(key == a[i]}

{

s.o.p(“found the element in index number ”+ i);

System.exit(0);

}

}

s.o.p(“not found in array”);

//program3

Int max = a[0];

For ( int i=0; i<a.length; i++)

{

If (a[i] > max)

{

Max = a[i];

}

}

s.o.p( “max number is” +max)

program4:

Int a[] = {3,9,7,8,12,6,15,5,4,10}

Rough:

Max1 = 3,9,12,15 max2 = 3,3,7,8,9,12

Int max1, max2 ;

Max1=max2 = a[0];

For ( int i=0; i<a.length; i++)

{

If (a[i] > max1)

{

Max2 = max1;

Max1 = a[i];

}

Else if (a[i] > max2)

{

Max2 = a[i];

}

}

s.op.( “second max element is ” +max2)

}

Program5:

public static void main(String[] args)

{

int A[]={8,6,10,9,2,15,7,13,14,11};

int B[]=new int[10];

for(int i=0;i<A.length;i++)

{

B[i]=A[i];

}

for(int x:B)

{

System.out.print(x+",");

}

}

}

**2D**

Collection of arrays or array of arrays

Int a[][] = new int[3][4]

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |

|  |
| --- |
| 0 |
| 1 |
| 2 |
|  |

a

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |

Int a[][]= {{1,2,3,4},{2,4,6,8},{3,5,7,9}}

Int a[][] = new int[3][4]

A[] – reference to an array of reference

(2nd)[] – reference to an array

|  |
| --- |
| 0 |
| 1 |
| 2 |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |

a

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 4 | 6 | 8 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 5 | 7 | 9 |

Can be also declared as,

Int a[][];= new int[5][5];

Int b[][] = {{1,2,3,4},{2,4,6,8},{3,5,7,9}}

Int c[][] ;

C= new int[5][5];

Int[] d[] = new int[5][5];

Int[] E,F[];

E 🡪 1D array

F 🡪 2Darray

**Program:**

**to print the values in 2D array**

|  |
| --- |
| No. of rows |

For (int i=0; i<a.length;i++) no of columns

{ for (int j= 0; j<a[0].length ;j++)

{

s.o.p(a[i][j]);

}

s.o.p(“\n”);

**to access each loop**

For (int x[] :a) 🡪 x[] reference to array

{ for (int y : x) 🡪 Y takes the element

{

s.o.p(y) }

s.o.p(“\n”);

**Program**:

public class arrayExample {

public static void main(String[] args) {

Int b[][] = {{1,2,3},{2,4,6},{1,3,5}};

For ( int i=0; i<b.length; i++}

{

For ( int j=0; j<b[0].length; j++}

{

s.o.print(b[i][j]+””);

} s.o.pln(“”);

} } }

(OR)

For(int x[]:B)

{

For(int y:x)

{

s.o.print(y);

}

s.o.pln(“”):

} } }

(OR)

**What will happen if I directly print B?**

**s.o.pn(B);**

**Guess the result ?**

**how to access the jagged array 🡪 means of different size**

0 1

|  |
| --- |
| 0 |
| 1 |
| 2 |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 |  |  |

a

0 0 1 2 3

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 4 | 6 | 8 |

0 1 2

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 5 | 7 | 9 |

Int a[][];

A = new int [3][];

A[0] = new int[2];

A[1]= new int[4];

A[2] = new int[3];

Example:

public class arrayExample {

public static void main(String[] args) {

Int A[][] = new int[3][];

A[0] = new int[5];

A[1]= new int[3];

A[2] = new int[8];

For ( int i=0; i<A.length; i++}

{

For ( int j=0; j<A[i].length; j++}

{

s.o.print(A[i][j]+””);

} s.o.pln(“”);

} } }

(OR)

public class arrayExample {

public static void main(String[] args) {

Int A[][] = new int[3][];

A[0] = new int[5];

A[1]= new int[3];

A[2] = new int[8];

For ( int x[] : A)

{

For ( int Y : x)

{

s.o.print(y +””);

} s.o.pln(“”);

} } }

Program : Addition of 2 arrays and store it in 3rd array.

public class arrayExample {

public static void main(String[] args) {

Int a[][] = {{3,5,9},{7,6,2},{4,3,5}}

Int b[][]= {{1,5,2},{6,8,4},{3,9,7}}

Int c[][]= new int[3][3]

For ( int i = 0; i<a.length; i++)

{

For ( int j=0; j<a[0].length; j++)

{

C[i][j] = a[i][j]+b[i][j];}

}

For ( x[] :c)

{

For ( y : x)

{

s.o.pln(y+””)

} } }}

Methods: **Methods** are **blocks of code that perform a specific task** and are executed when they are **called or invoked**

**In c, C++ - called as functions but in java we call as Method**

Data

Method 1 Method 2 Method 3 Method n

returnType methodName (Parameter list)

{

// body

//

//

}

returnType -> signature of the method / header of the method (In C or C++ 🡪 prototype)

Example: print the max number

class Test

{

static int max( int x, int y) 🡪 defining method / formal parameter

{

If( x > y)

Return x;

Else

Return y;

}

Public static void main (String args[])

{

Int a=10,b=15,c;

C = max(a,b); 🡪 calling method / actual parameter

s.o.pln(c);

}

}

Main method defining method

a b x y

10 15 10 15

X++ 🡪 value will change to 11 only in defining method

Program:

Static method:

public class MethodPractice {

static int max(int x,int y) //static method

{

if(x>y)

return x;

else

return y;

}

public static void main(String[] args) {

int a=10,b=15;

System.out.println(max(a,b)); 🡪 static method

}

}

Non static method:

public class MethodPractice {

int max(int x, inty) // non static method

{ if(x>y)

return x;

else

return y;

}

public static void main(String[] args) {

int a=10,b=15;

//non static method – create object for class then call the method

MethodPractice mp=new MethodPractice();

System.out.println(mp.max(a,b));

}

}

Difference between actual and formal parameter values:

public class MethodInc {

static void inc(int x)

{

X++;

s.o.pl(“formal parmeter value” +x)

}

public static void main(String[] args) {

int a=10;

inc(a)

s.o.pl(“actual parmeter value” +a)

}

}

**Passing object as Parameter:**

public class Test {

static void update(int a[])

{ a[1] = 25;

}

public static void main(String[] args) {

int a[]={2,4,6,8};

update(a);

System.out.println(a[0])

}

}

Array will be modified to {25,4,6,8} since the reference is passed to change the object, the object itself changing ( different from basic data types like int)

Program for object passing as parameter:

public class MethodPractice1 {

static void change(int A[],int index,int value)

{

A[index]=value;

}

static void change2(int x,int value)

{

x=value;

}

public static void main(String[] args) {

int A[]={2,4,6,8,10};

change(A,2,20);

for(int x:A)

{

System.out.println(x);

}

int x=10;

change2(x,20);

System.out.println("Value of "+x); 🡪 value of x remains as 10 because its not an array/ basic primitive data type.

}

}

**Parameter passing in Java**

Content of actual parameter is passed to formal parameter

Example:

1. A passed to X, B passed to Y -🡪 Z passed to C after addition

X Y Z

Int add (int x, inty) 10 15 25

{int z;

Z=x+y;

Return z; }

P.S.V.M(String args[]) A B C

{ int a=10,b=15,c; 10 15 25

C = add(int a,b)

s.o.pln(C);

} }

1. Name & N both methods will point to same **object** because string is object not primitive data type like int

Void welcome(String n) n

{ S.o.pln(“ Welcome to this “+ n);

}

P.S.V.M(String args[]) name

{ String name = “Victor”; Victor

welcome(name);

} }

Program:

Find prime number:

public class primeNumber

{

static boolean isPrime(int n)

{

for(int i=2;i<n/2;i++)

{

if(n%i==0)

return false;

}

return true;

}

public static void main(String[] args)

{

System.out.println(isPrime(19));

//check 91

} }