**INTRODUCTION TO JAVA**

**What is \*Pseudo Code\* ?**

* Pseudocode is a description of an algorithm or program that uses natural language instead of a programming language.
* It's a blueprint for writing code, and it's often used as a rough draft.
* Basically converting programming language into English language that is understand by the user.

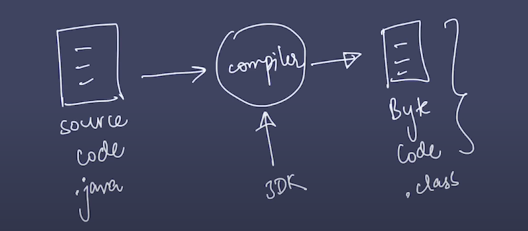
**How my java Code is Running?**

There are two steps from which my code is running

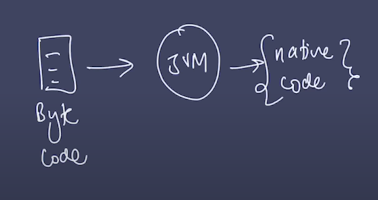
1. **Compilation**
2. **Execution**



**What happens in the compilation stage?**

****

**What happens in the Execution stage?**

****

**Together**

****

**What does JVM stands for what it is?**

* JVM stands for Java Virtual Machine it is a Java interpreter.
* It is responsible for loading, verifying, and executing the bytecode created in Java.

**What does JRE stand for and what it is?**

* The Java Runtime Environment (JRE) is software that allows Java programs to run on a computer.
* It's a collection of tools that includes the Java Virtual Machine (JVM), Java class libraries, and the Java class loader.
* The JRE act as a translator between Java programs and the operating system, ensuring compatibility.

**What does JDK stands for and what it does?**

* JDK (Java Development Kit) stands for Java Development Kit.
* It is a software development kit used to develop Java applications.

**Difference between JDK, JRE, and JVM**

* JDK: Java Development Kit is a software development environment used for developing and Executing Java applications and applets.
* JRE: JRE stands for Java Runtime Environment and it provides an environment to run only the Java program onto the system. (not to deploy)
* JVM: JVM stands for Java Virtual Machine and is responsible for executing the Java program. (also called as interpreter)

**BASIC FUNDAMENTALS IN JAVA**

* **OUTPUT**
* **VARIABLES**
* **DATA TYPES**
* **INPUT**

**Input/output**

**Syntax :-** System.out.print(“ “);

1. **print :- system.out.print(“….”);** the line continues
2. **println :- system.out.println(“…”);** the 2nd output goes to nextline
3. **“\n”:-** goes to next line

**VARIABLES**

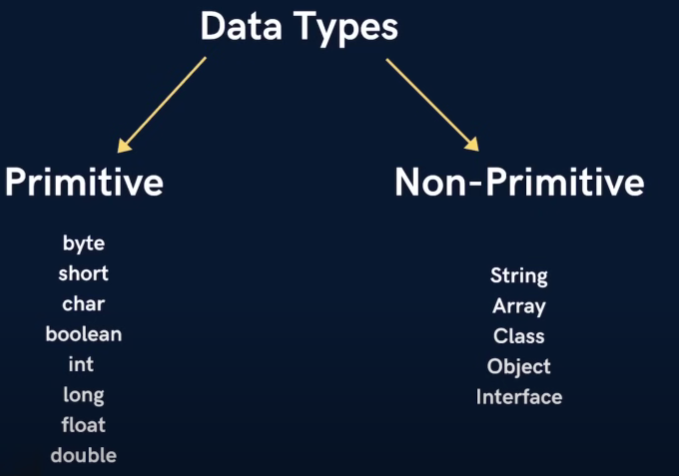
A variable is a named location in computer memory that stores data that can change.

**How variables are stored in the computer memory?**

1. Variables are stored in memory cells built from transistors and other components on an integrated circuit.
2. Variables are assigned a memory address when they are created.
3. The memory address is the location of the variable in the computer's memory.
4. The value of a variable is stored in the memory address.

**DATATYPES**

Since JAVA is a **Typed Language** we have to mention what is the type of data we have written to get him the proper space in the memory

****

* JAVA does not follow the BODMAS rule.
* In Java calculation happens from left to right.

INPUT

Eg code

package Input\_03; 🡪 Declares that `Question.java` belongs to the `Input\_03` package, which organizes classes and prevents naming conflicts; the file must be inside the `Input\_03` folder for correct compilation.

import java.util.\*; 🡪 Imports the `java.util` package, which includes the `Scanner` class used for taking user input.

public class Question { 🡪 Declares a public class Question, which must match the filename Question.java.

public static void main(String[] args) {

 This is the entry point of the Java program.

 public → Allows access from anywhere.

 static → Belongs to the class, doesn't need an object to be called.

 void → No return value.

 String[] args → Command-line arguments (not used here).

Scanner sc = new Scanner(System.in);

 Scanner sc → Creates a Scanner object named sc.

 new Scanner(System.in) → Reads input from the keyboard.

System.out.println("Enter two numbers : ");

int a = sc.nextInt();

int b = sc.nextInt();

 sc.nextInt(); → Reads an integer input from the user and stores it in a.

 sc.nextInt(); → Reads another integer input and stores it in b.

int sum = a + b;

System.out.println("Sum : " + sum);

sc.close(); 🡪 Closes the Scanner to free up system resources.

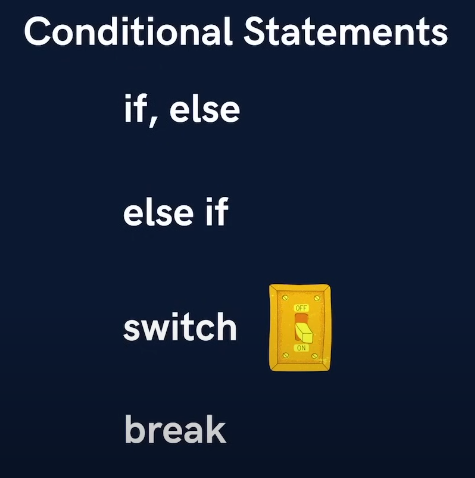
}

}

**CONDITIONAL STATEMENTS**

Conditional statements control the flow of a program based on certain conditions (true or false).

**The main Conditional Statements are**



1. **If, Else**



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

if (age >= 18) {

System.out.println("You can vote.");

} else {

System.out.println("You cannot vote.");

}

1. **else-if**

* Checks multiple conditions sequentially.
* if (marks >= 90) {

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

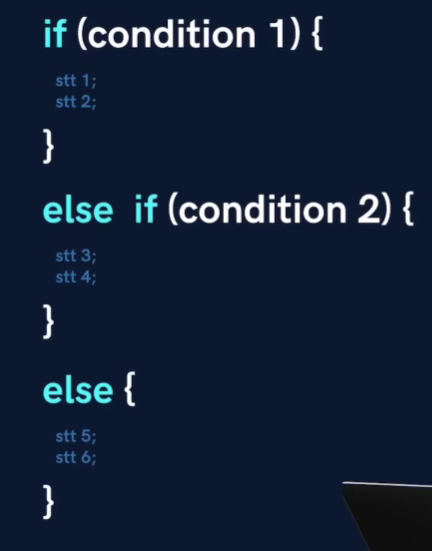
} else if (marks >= 75) {

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

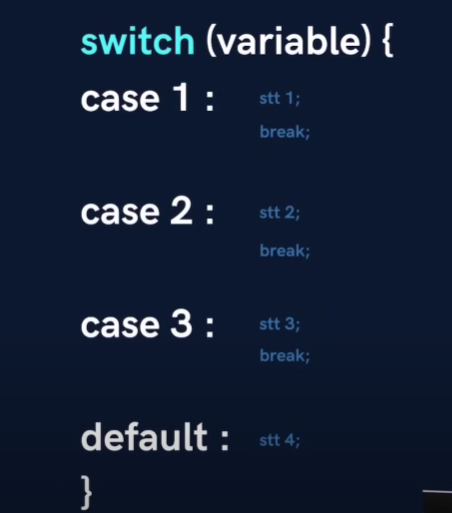
} else {

System.out.println("Grade: C");

}



1. **switch Statement**



* Used when multiple conditions depend on a single variable.
* 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");

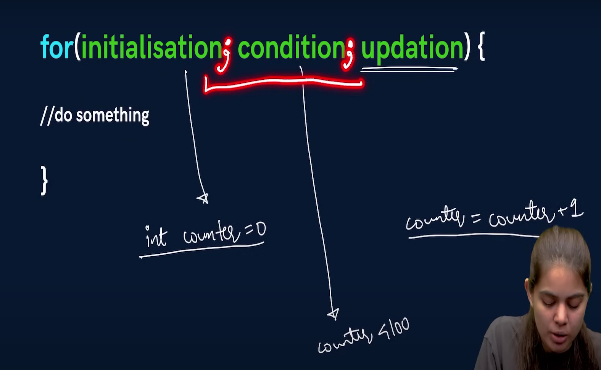
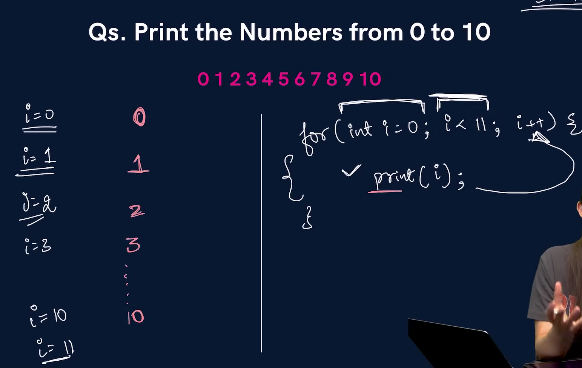
}

**LOOPS**

* A loop is used for executing a block of statements repeatedly until a particular  condition is satisfied.
* A loop consists of an initialization statement, a test condition and an increment statement.)

**Types of Loops** :

1. **For loop**

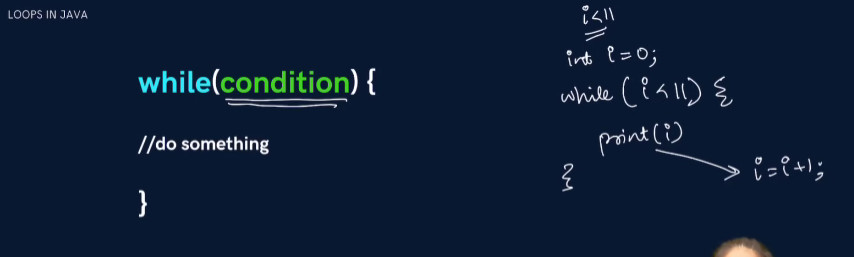
** **

* Runs a block of code a specific number of times.
* for (int i = 1; i <= 5; i++) {

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

}

1. **While loop**

****

* Runs a block of code **while** a condition is true.

int i = 1;

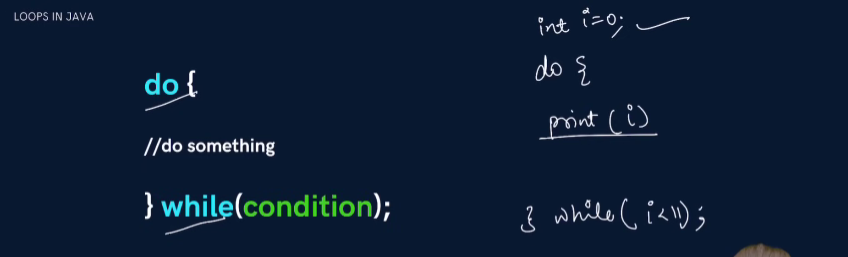
while (i <= 5) {

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

i++;

}

1. **Do while loop**

****

* Runs the code **at least once**, then repeats **while** the condition is true.
* int i = 1;

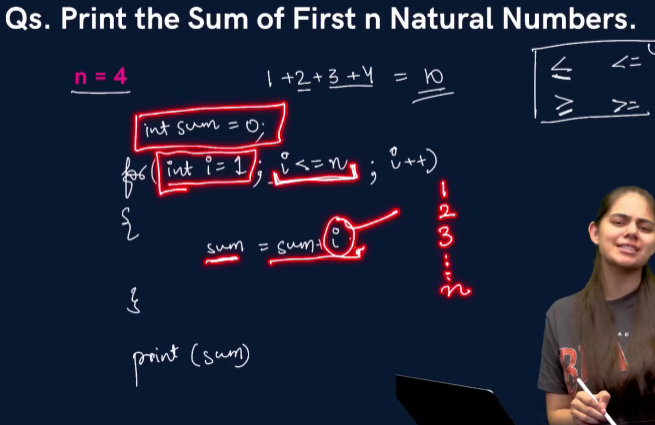
do {

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

i++;

} while (i <= 5);

**Egs on Loops**



package Loops;

import java.util.\*;

public class eg {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the no :");

        int n = sc.nextInt();

        int sum = 0;

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

            sum = sum + i;

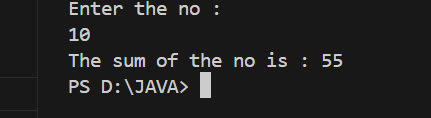
        }

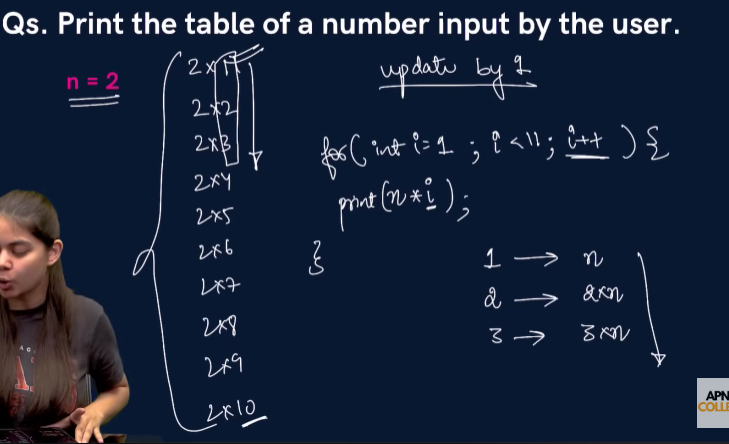
        System.out.println("The sum of the no is : " + sum);

        sc.close();

    }

}





package Loops;

import java.util.Scanner;

public class eg2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the no :");

        int n = sc.nextInt();

        for (int i = 1; i < 11; i++) {

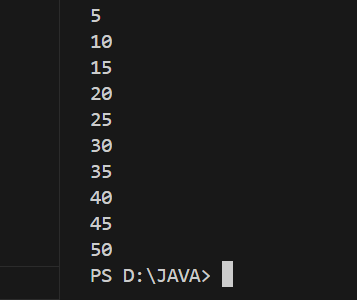
            System.out.println(i \* n);

        }

        sc.close();

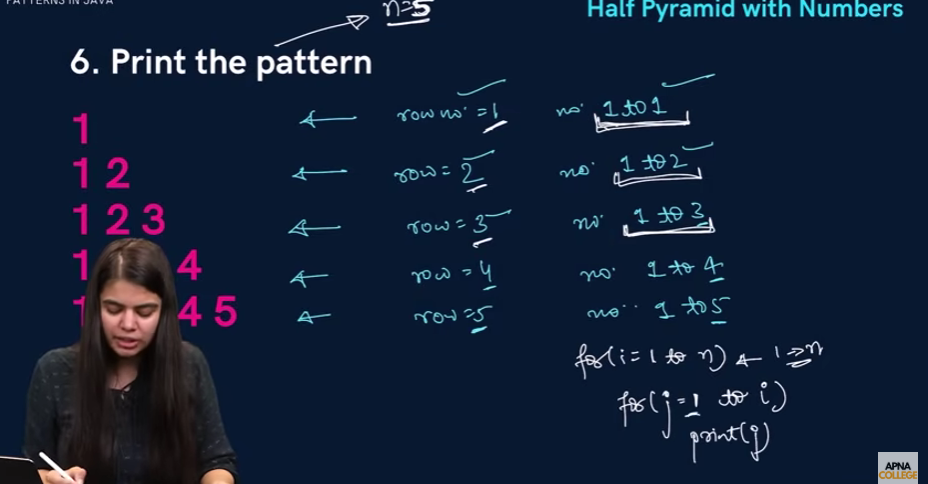
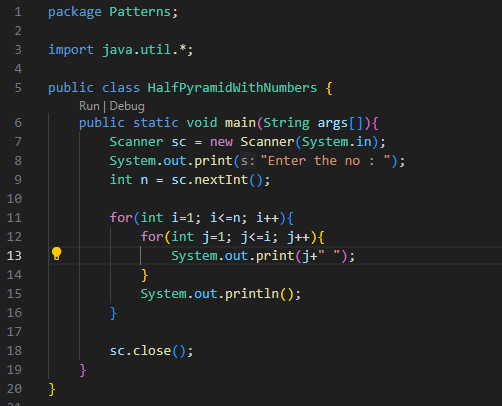
    }

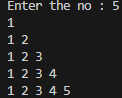
}



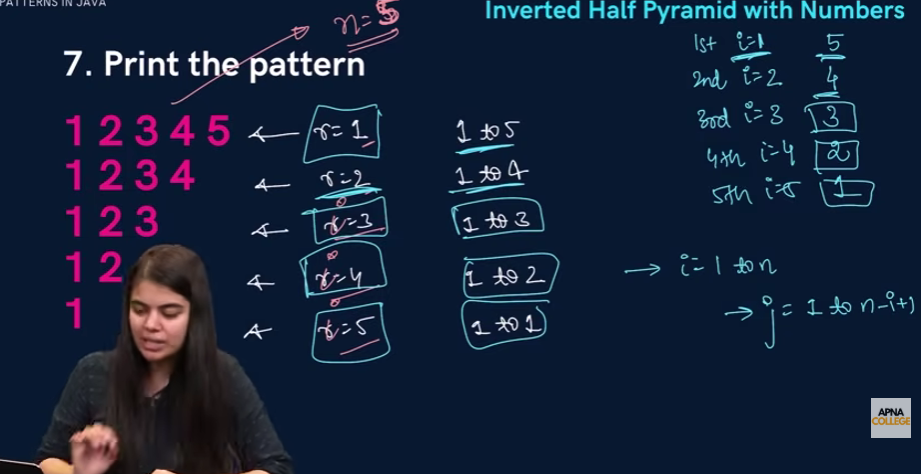
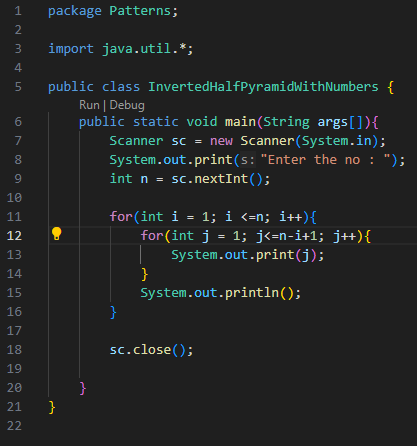
**PATTERNS**

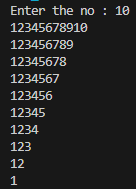
**Half Pyramind with Numbers**

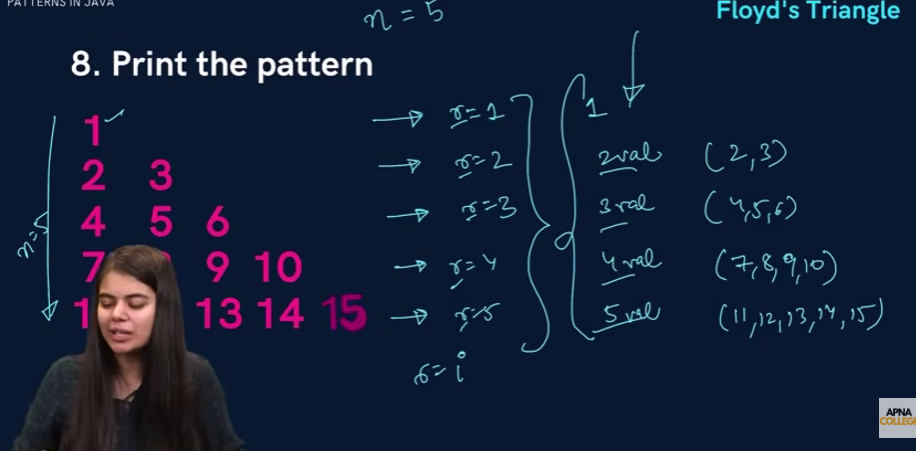
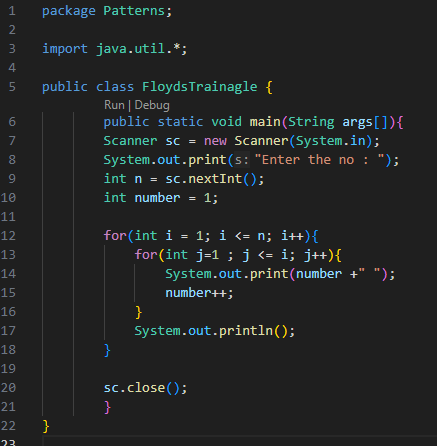


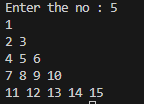
**Inverted Half Pyramind with Numbers**

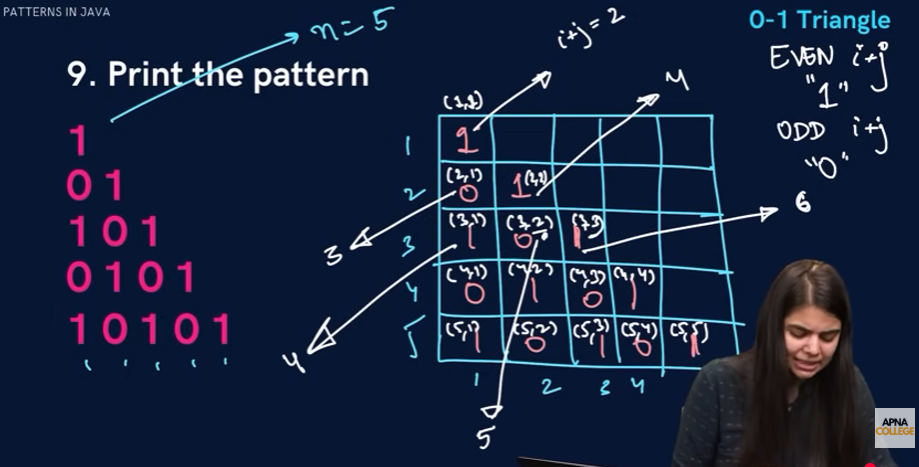
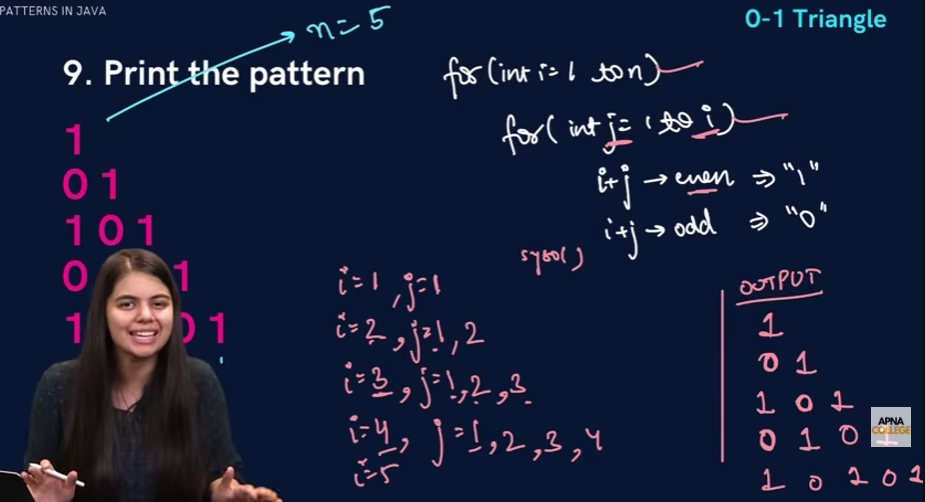


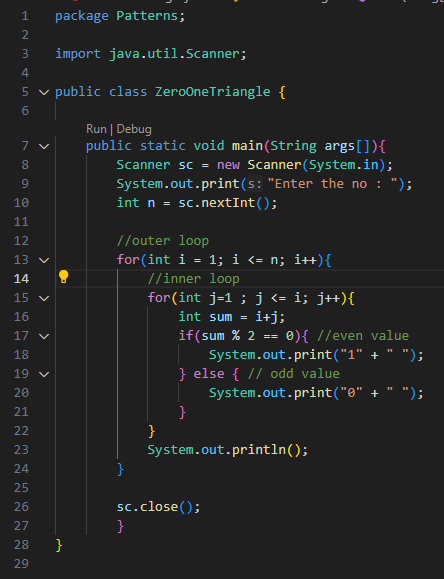
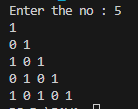
**Floyd’s Triangle**

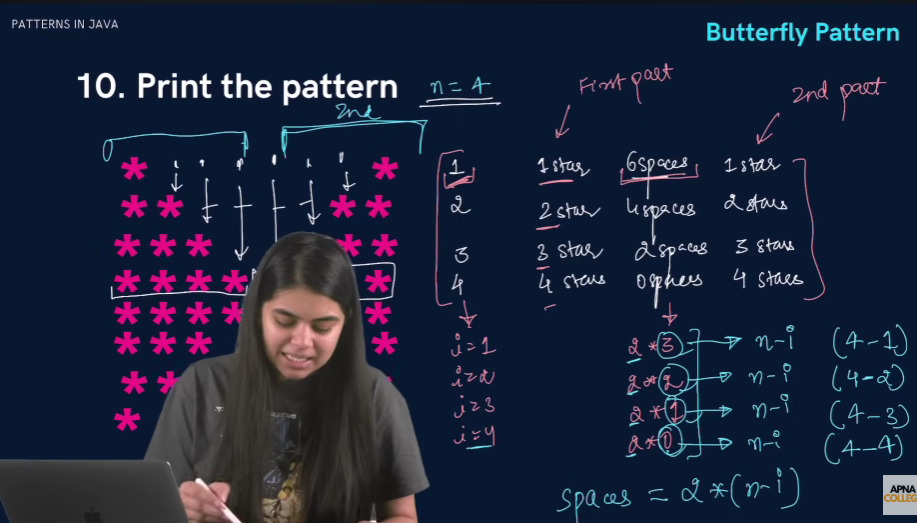
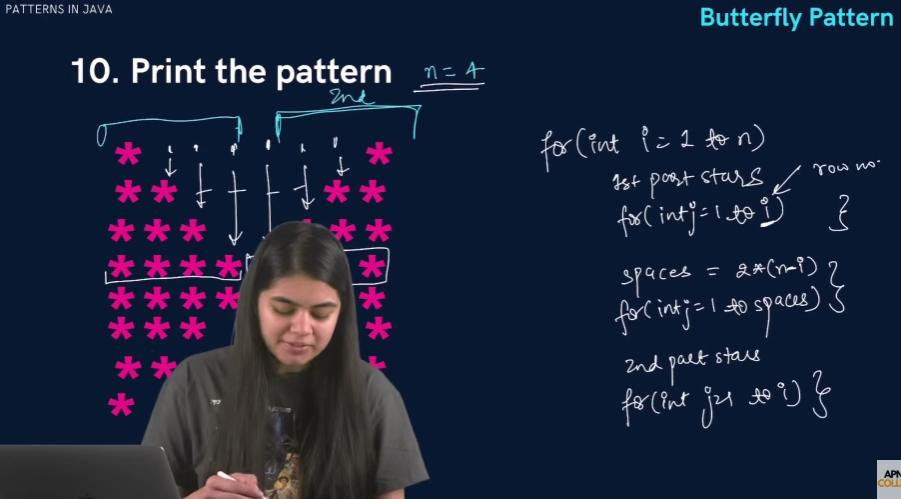


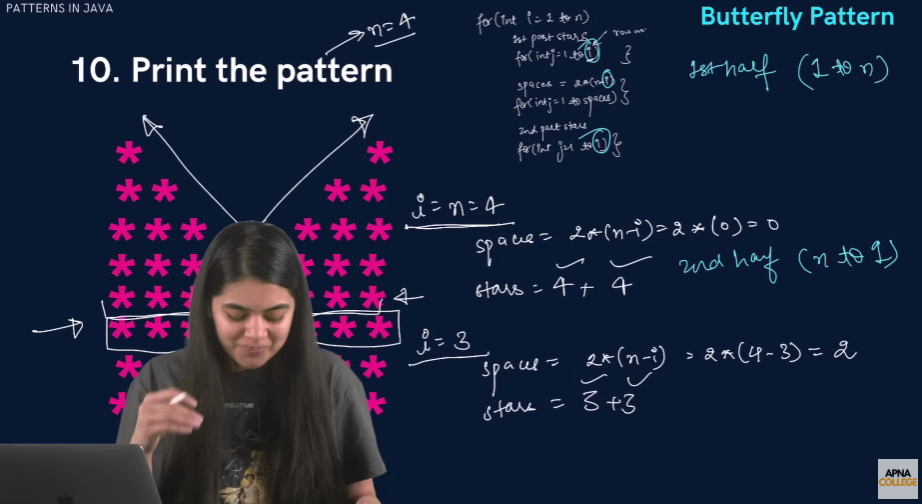
**0-1 Triangle**

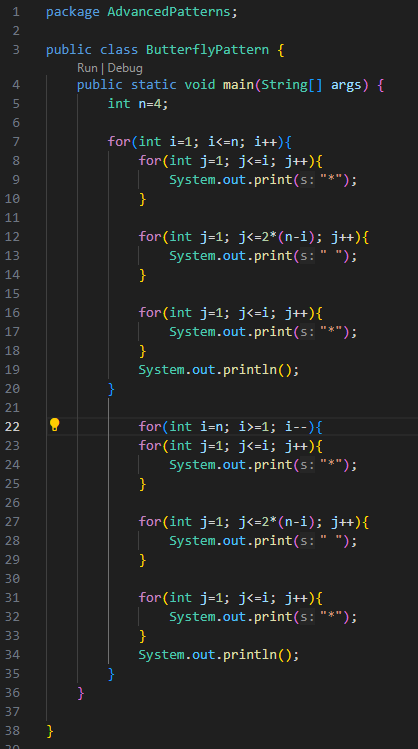
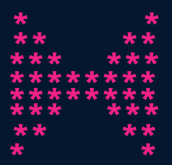
 

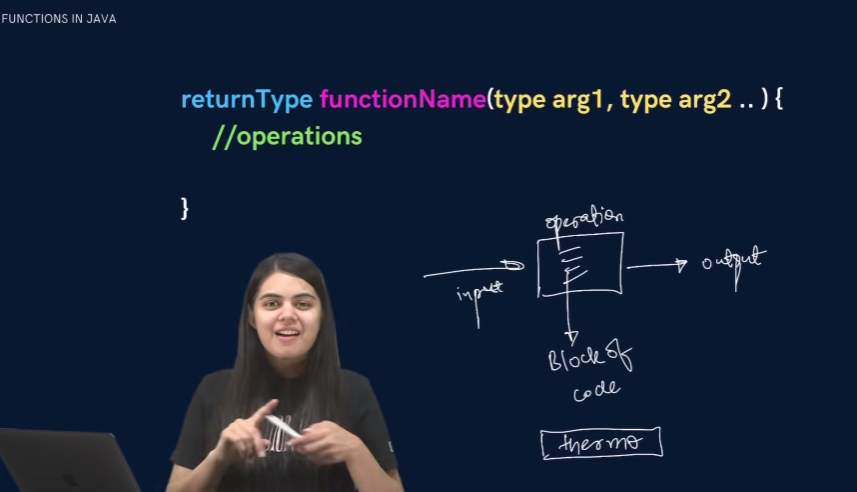
**Butterfly Pattern**



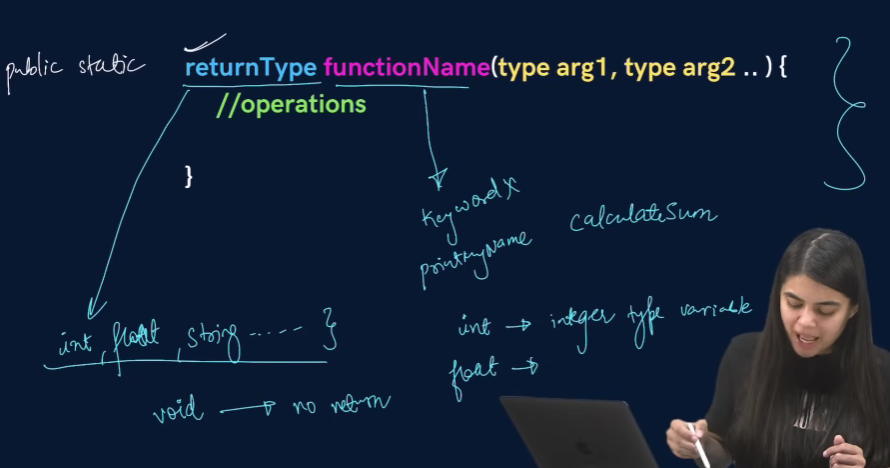
**Functions**



**A function is a block of code that performs a specific task.**

Why are functions used?

* If some functionality is performed at multiple places in software, then  rather than writing the same code, again and again, we create a function  and call it everywhere. This helps reduce code redundancy.
* Functions make maintenance of code easy as we have to change at one  place if we make future changes to the functionality.
* Functions make the code more readable and easy to understand.



The **syntax** for function declaration is :

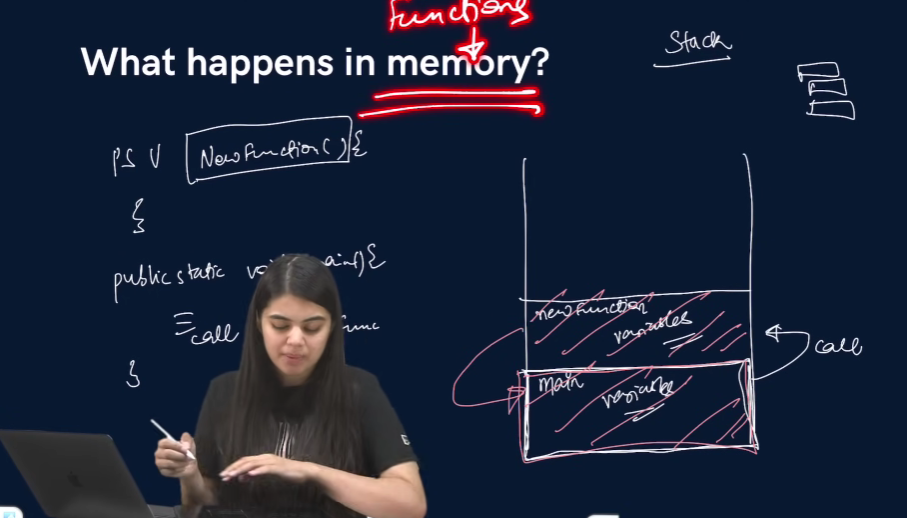
return-type function\_name (parameter 1, parameter2,  …… parameter n){

//function\_body

}

return-type

The **return type** of a function is the data type of the variable that that function  returns.



For eg - If we write a function that adds 2 integers and returns their sum then  the return type of this function will be ‘int’ as we will return a sum that is an  integer value.

When a function does not return any value, in that case the return type of the  function is ‘void’.

**function\_name**

It is the unique name of that function.

It is always recommended to declare a function before it is used.

**Parameters**

A function can take some parameters as inputs. These parameters are specified  along with their data types.

For eg- if we are writing a function to add 2 integers, the parameters would be  passed like –

int add (int num1, int num2)

**main function**

The main function is a special function as the computer starts running the code  from the beginning of the main function. Main function serves as the entry  point for the program.

**Example** :

package Functions;

public class Main{

public static void sum(int a, int b){

int sum = a + b;

System.out.println(sum);

}

public static void main(String args[]){

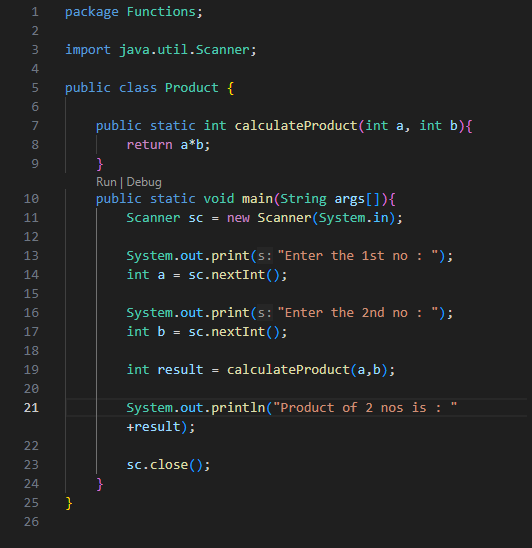
int a = 10;

 int b = 20;

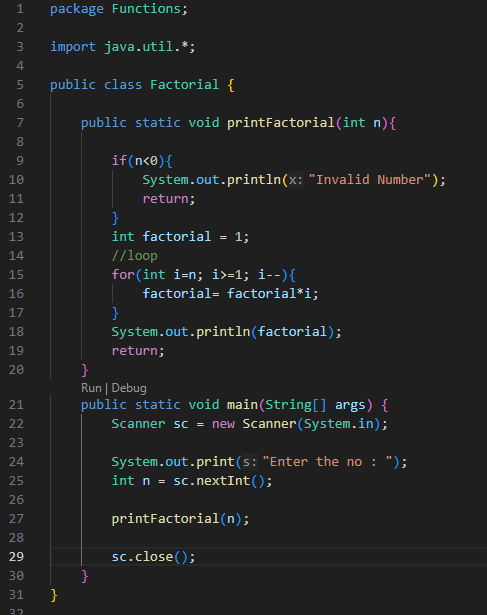
*sum*(a, b); // Function Call

}}

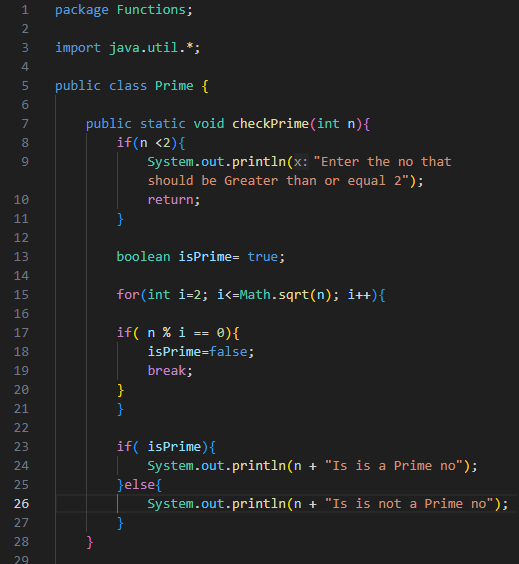
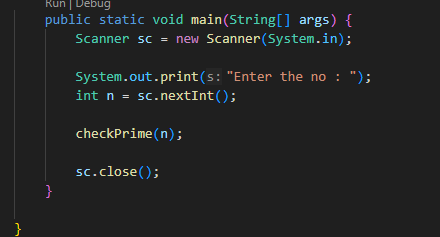
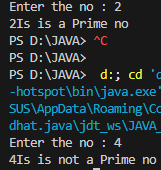
**Qs. Write a function to multiply 2 numbers.**



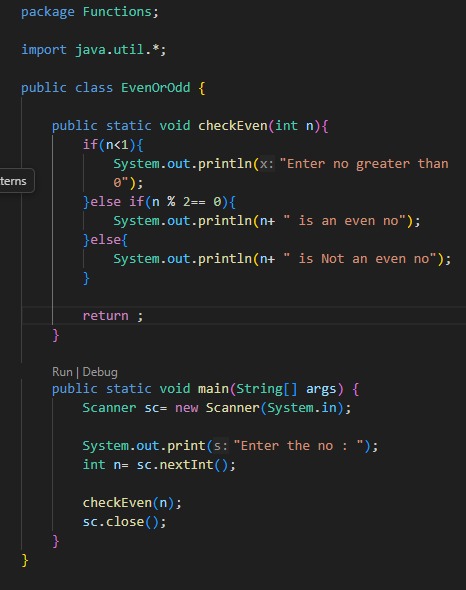
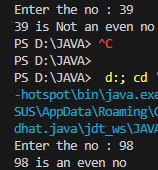
**Qs. Write a function to calculate the factorial of a number**

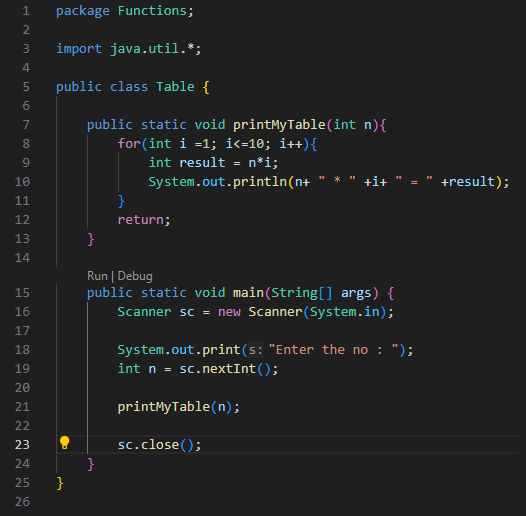
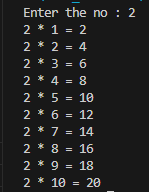
1. **Make a function to check if a number is prime or not.**

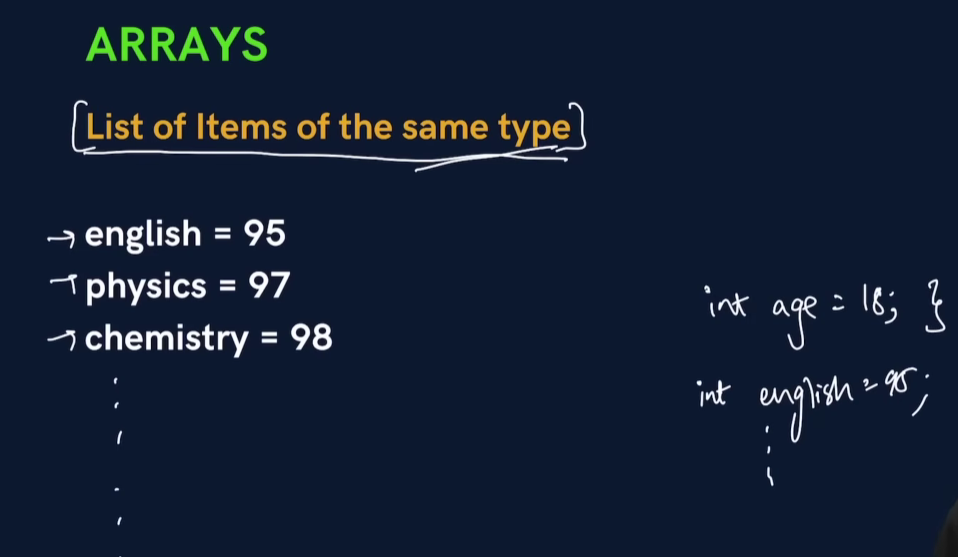
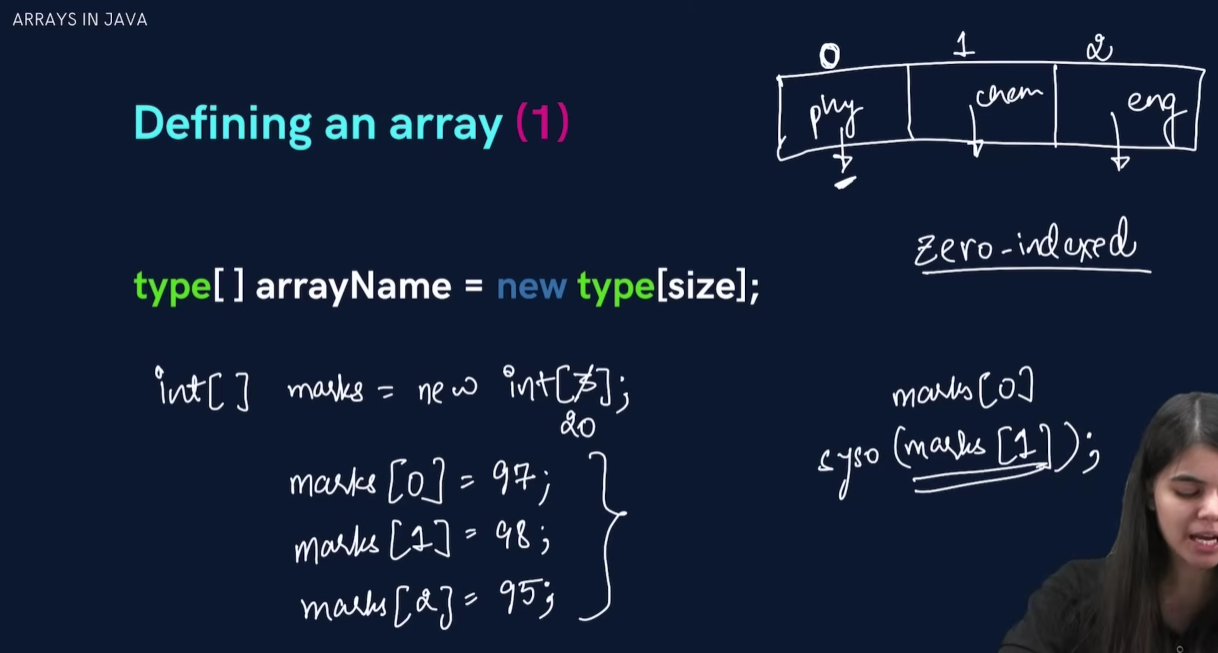
1. **Make a function to check if a given number n is even or not.**

**Make a function to print the table of a given number n.**

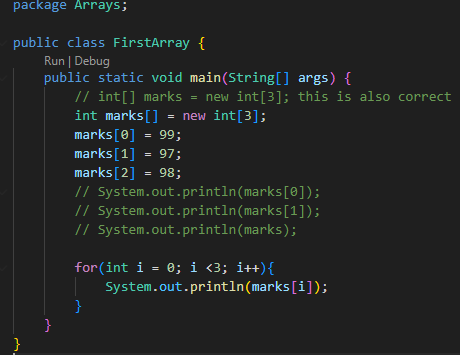
 

**ARRAYS**

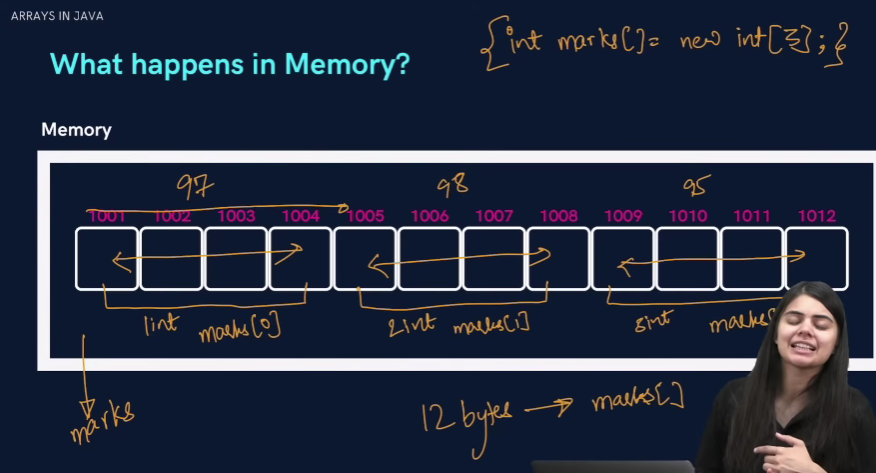
1. Arrays are **objects**
2. Printing array name prints **address (hashcode)**
3. Use Arrays.toString() to print all elements
4. Array index starts from **0**
5. Stored in **heap memory**

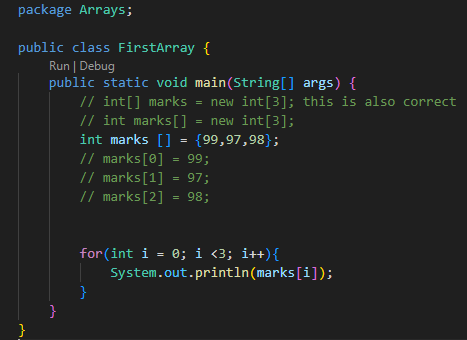
**EXAMPLE.**

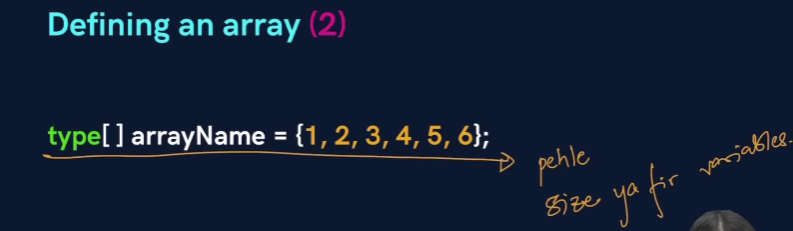
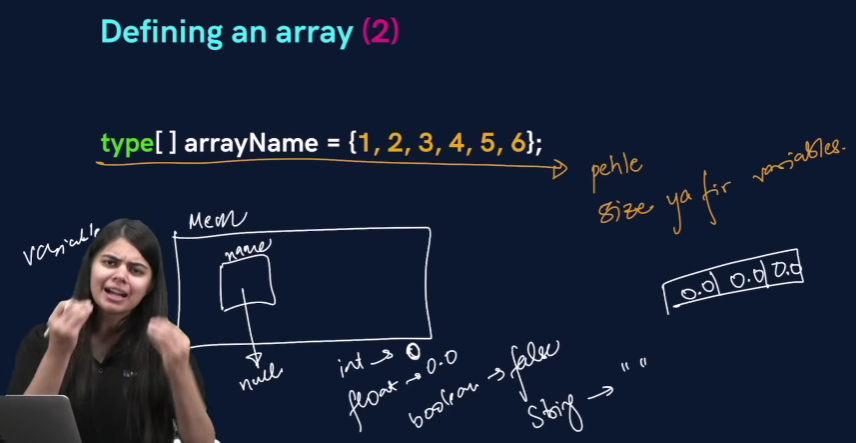
 

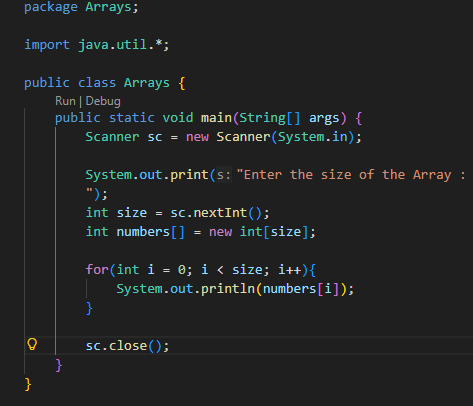
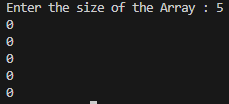
| **Command** | **Output** | **Represents** |
| --- | --- | --- |
| marks[0] | 99 | Element at index 0 |
| marks[1] | 97 | Element at index 1 |
| System.out.println(marks) | [I@hexCode | Array object reference |

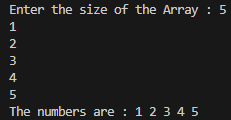
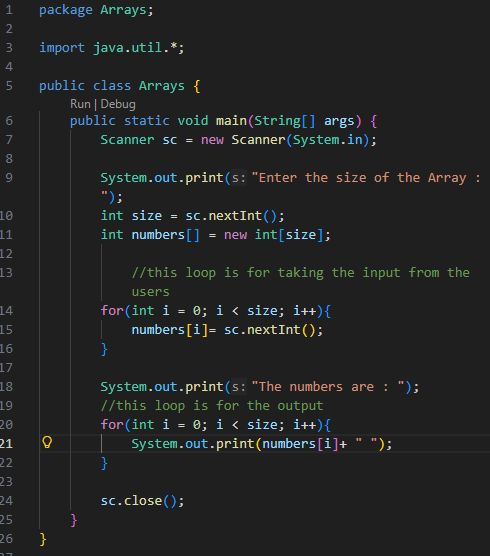
| **Part** | **Meaning** |
| --- | --- |
| [ | It’s an **array** |
| I | Type of array → **int** (I stands for Integer type in JVM) |
| @ | Separator |
| 2c7b84de | **Hashcode / memory reference** of array |





**Linear Search**

