11

JAVA

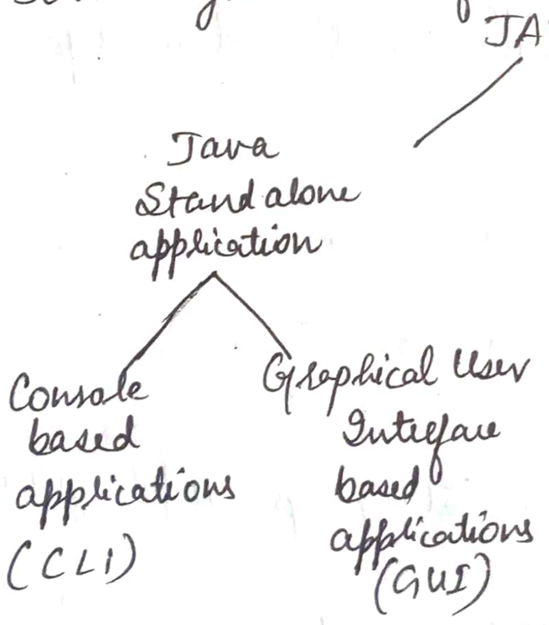
→ Java was conceived by James Gosling in 1995 (father of JAVA) at Sun Microsystems.

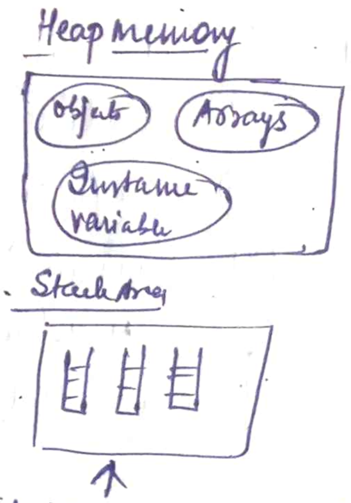
→ Java is a high-level, class-based, object oriented programming language.

→ It is a general-purpose programming language intended to let programmers "Write once, run anywhere (WORA)". which means that compiled Java code can run on all platforms that support Java without the need ~~of~~ to recompile.

→ Java is used to develop mobile apps, web apps, desktop apps, games and much more.

→ Java Programs is of two types


JAVA
Java Stand alone application
Web Applets
Console based applications (CLI)
Graphical User Interface based applications (GUI)



Total 5 types of Memory Areas

1. Method Area
2. Heap Area
3. Stack Area
4. PC Register
5. Native Method Area

→ Stand-alone application is a program written in Java to carry out certain task on local stand alone computer.

→ Web applets are small java programs developed for internet. (small app embedded in a webpage)

→ Stand-alone program (run without a web browser)

Stand-alone Application

→ Execution of Java Programs involve two step:

① compiling source code into bytecode using javac compiler

② Bytecodes are executed by using java interpreter.

Web Applets

Can run only in web browsers.

Implementation of JAVA Programs :-

Implementation of JAVA programs involve three steps:

① Creating a Program
② Compiling a Program
③ Running a Program

① Notepad Hello.java

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

or String[] args)

→ Source file

static var ↑ (PrintStream)

↓ class (java.lang package)

↓ method

② Compilation

```
      user
C:\>  Javac Hello.Java ↵ Compile (Source code to byte code)
```

→ creates bytecode with name Hello.~~java~~ class

③ Running

```
C:|user> Java Hello
```

Output -: Hello Java

$$\begin{tikzcd}
\boxed{\text{Hello.Java}} \arrow[d, "\text{Javac Hello.Java} \hookleftarrow \text{compile}", "\text{Compile}"'] & \text{Source code} \arrow[l] \\
\boxed{\text{Hello.class}} \arrow[d, "\text{Java Hello} \hookleftarrow \text{Interpreter}", "\text{Run}"'] & \text{Byte code} \arrow[l] \\
\boxed{\text{Hello World}} &
\end{tikzcd}$$

Explanation of the above syntax

① Public - So that JVM can execute the method from anywhere.

② Static - The main method is to be called without an object. The modifiers are public and static can be written in either order.

③ void - The main method doesn't return anything.

④ main () - Name configured in the JVM. The main method must be inside the class definition. The compiler executes the codes starting always from the main function.

⑤ String [ ] - The main method accepts a single argument, i.e. an array of elements of type String.

⑥ args - is the array name, & it is of String type. i.e. store group of string.

Java Tokens

Smallest individual units in a java program are known as java tokens. Java Programs are collection of tokens, comments, and whitespaces.

The different type of Java tokens are:

① Reserved Keywords → 60 Keywords; public, private, int, float, if, etc.

② Identifiers - naming of classes, methods, variables, interfaces, packages, all are identifiers.

③ literals - integer, float, char, etc, string, boolean

④ Operators - +, +, ++, --, *, /, %, <, >, ==, !=

⑤ Separators - (), {}, [], ;, ,, :

Q: Features of Java

Ex:

```
class Example {
   public static void main (String args [])
{
   int num;   // this declares a variable called num
  num = 100;  // this assigns num the value 100
 System.out.println ("This is num:" + num);  → appended to the string
  num = num * 2;
 System.out.println ("The value of num * 2 is ");
 System.out.println (num);
   }
 }
```

→ o/p This is num : 100
The value of num * 2 is 200

print   println → next line

Declaring Objects

By using new operator.

The new operator dynamically allocates (i.e. allocates at run time) memory for an object & returns a reference to it.

```
Classname  ref-var  =  new classname ();
```

For ex:

```
class A
{
  int x = 10;
  public static void main (String ... s)
{
  A a = new A();
  SOP ("value of x" + a.x);
                      ↓
                  // ref-var . variable name ;
}
}
```

O/P = value of x 10.

Method - sec-B - 18/9/21

→ A method is like a funⁿ i.e. used to expose the behavior of an object.

→ It is a set of codes that performs a particular task.

```
<access_modifier> <return_type> <method name> ( list of parameters)
{
   // body
}
```

[illegible] — not return any [illegible]

Diff b/w main & void in java.

```
int square () // instance method
{
 return 10*10;
}
```

$$\begin{tikzcd} & \text{return type} & \text{method\_name} & \text{parameter list} \\ \text{modifier} & \text{public} \arrow[l] & \text{int} \arrow[ul] & \text{max} \arrow[u] \; (\text{int } x, \text{ int } y) \arrow[u] \end{tikzcd}$$

```
public int max (int x, int y)
{
    if (x>y)
        return x;
    else
        return y;
}
```

} → body of the method

```
import java .io. *;
class Addition {
    int sum=0;    // Initially taking sum as 0
  public int addTwoInt ( int a, int b)  // Method to add 2 no's
  {
    Sum = a+b;    // Adding 2 Integers value
    return sum;   // Returning summation of 2 values
  }
}
```

Tuto!

6- WAP to implement calculater in java

7- WAP to implement print the number 1 to 100 in java
even or odd season

8- WAP to implement Armstrong number

```
public class Main {                    ——— then after 2 this main
  static void myMethod () {
    SOP (" I just got executed !");
}                                      ——— always run
                                           first this
public static void main (String[] args) {  main
   myMethod ();
   SOP (" Hey I just got executed!");
   myMethod ();
   myMethod ();
   myMethod ();
}
```

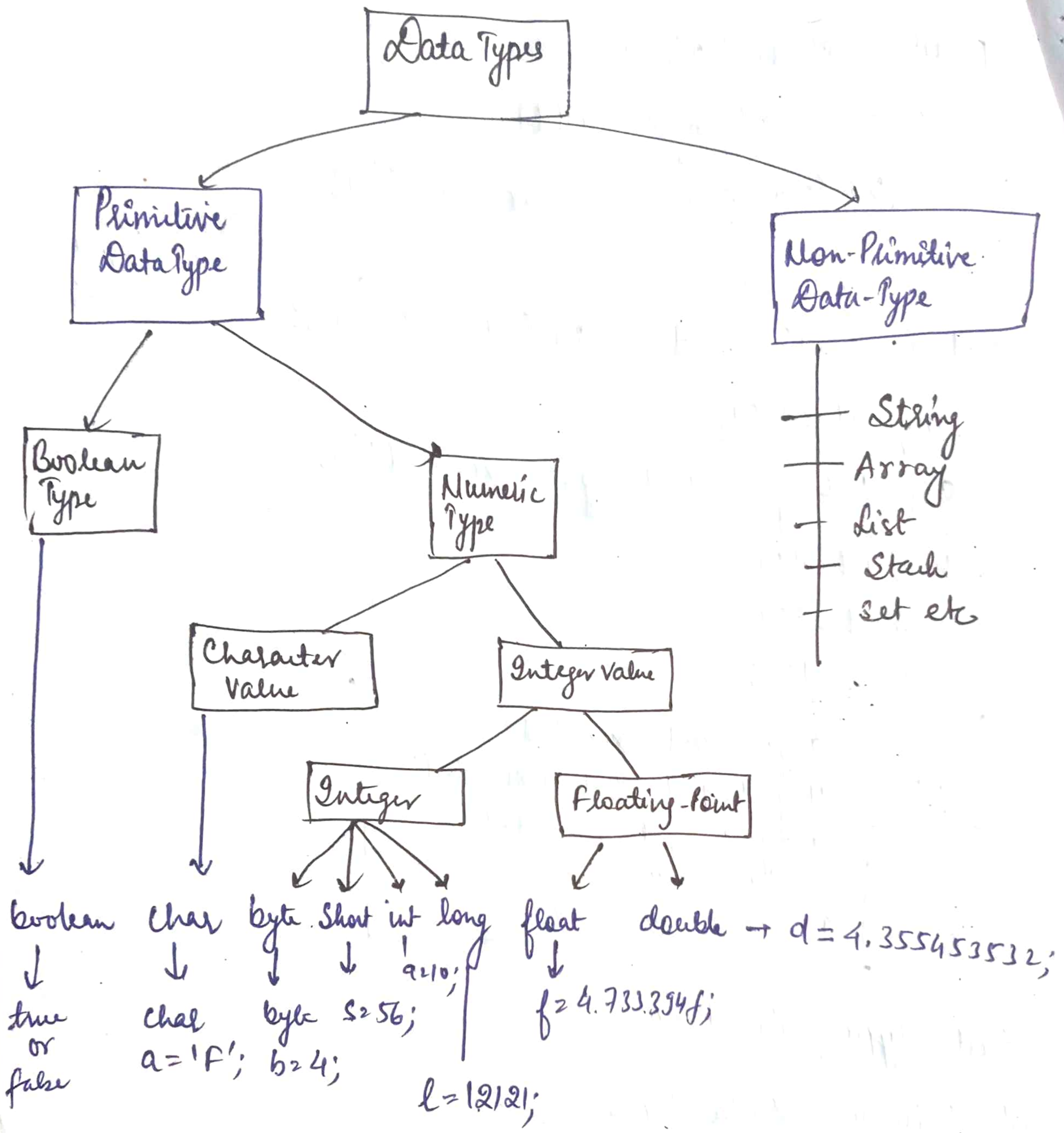
O/p:

```
I just got executed!
Hey I just got executed!
I just got executed!
I   "   "   "
I   "   "   "
```

Data Types in Java

① Primitive Data Types

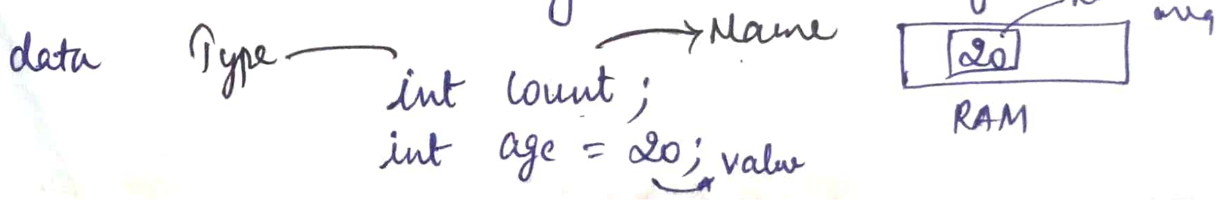
② Non - Primitive Data Types



Variables

Variables are the data containers that save the data values during Java program execution.

A variable is a memory location name for the



Void - ~~[illegible]~~ nothing (not even 0 or null)
means nothing

① local Variables ② Instance variable ③ Static variable

→ Local variables : Local variable are declared in methods, constructors or blocks.

stored
↓
Stack section

eg:
```
Void area ()
{
  int c; // local variables
}
```

→ Instance variables : Instance variables are declared in a class, but outside a method, constructor or any block.

stores in Heap - memory

```
class Employee
{
  public String name;   || instance variable
  private double salary;||
}
```

→ Static variable :
Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.

non-heap memory or
Static memory

```
class Employee {
  Static double salary; // Static variables
}static int m = 100;
```

```
class A
{
  int a;
}

A ob = new A();

ob. a;
```

How to access
1) directly
2) By any classname (A.b)
3) by any object ref. name

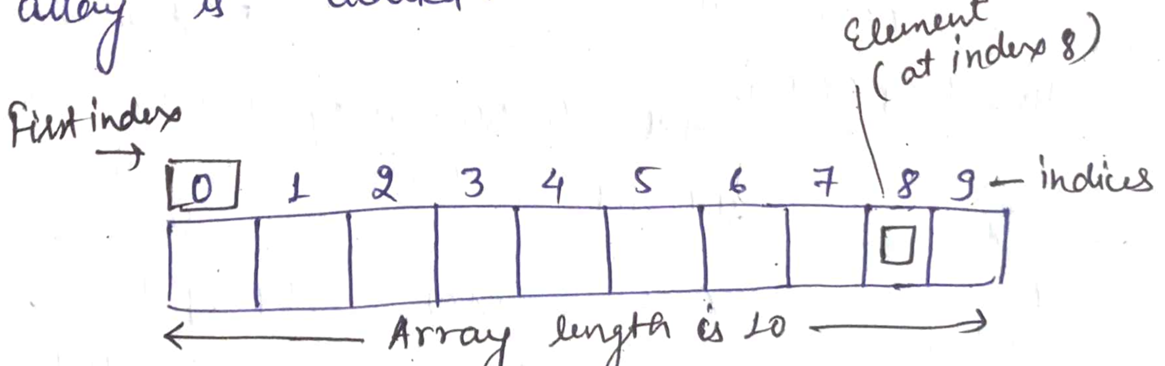
Ex:

```
    public class A
  {
   Static int m=200; // static variables
   void method ()
   {
    int n = 100; // local variable
  }
  Public static void main (String args[])
  {
   int data = 50;   // instance variable
  }
 } //end of the class
```

→ Diff b/w JDK, JRE, JVM
→ Control statements (if, for, if-else, if-else-if)

Arrays

An Array is a container object that holds a fixed number of values of a single type.
The length of an array is established when the array is created.

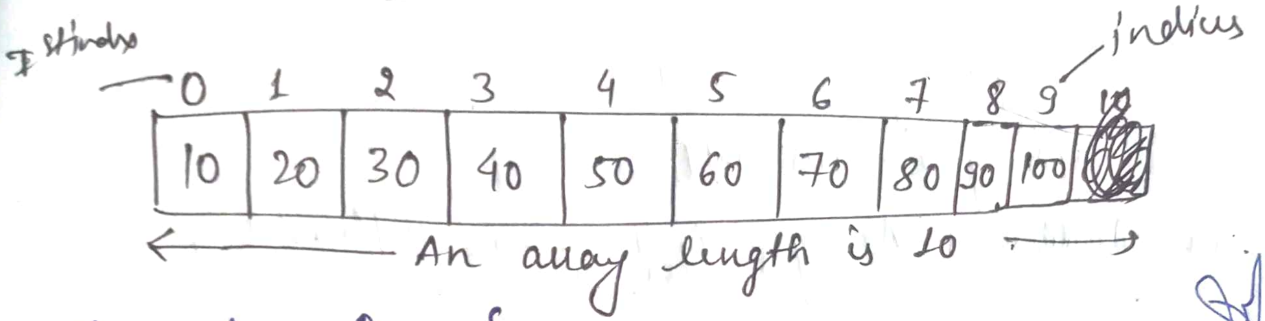


→ An array of 10 elements

```
          subscript operator
datatype []  var_name;
data type  var_name [];
```

```
int a[];
                  object ref
int a[] = new int[5];
```

→ Each item in an array is called an element, and each element is accessed by its numerical index.



```
class ArrayDemo {
  p s v m (String [] args) {
                                int anArray[]
    int[] anArray; // declares an array of integers
    anArray = new int[10]; // allocates memory for 10 integers

    anArray[0] = 10; // initialize first element
    anArray[1] = 20; // initialize second element
    anArray[2] = 30;     // and so forth
    anArray[3] = 40;
    anArray[4] = 50;
    anArray[5] = 60;
    anArray[6] = 70;
    anArray[7] = 80;
    anArray[8] = 90;
    anArray[9] = 100;
    S.O.P ("Element at index 0:" + anArray[0]);
    S.O.P ("   "   " index 1:" + anArray[1]);
      ⋮
    S.O.P ("Element at index 9:" + anArray[9]);
  }
}
```

looping statement

O/P

```
Element at index 0: 10
  "      " index 1 : 20
  "      "   "   2 : 30
  —    —    —    —
Element at index 9 : 100
```

SOP(a[3]);

Shortcut

int[] anArray = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

Here, the length of Array is determined by the no.s of values provided b/w braces & separated by commas (,).

int[] myArray = new int[5];

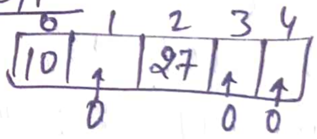
int myArray[] = new int[5];

We can Initialize Arrays in Java in 2 Different ways:

① Initialize using index

enter more values than the size of the array. it will raise an ArrayIndexOutOfBoundsException

Ex:
```
int[] myarr = new int[5];
myarr[0] = 10;
myarr[2] = 27;
```
O/P



② Initialize while declaring

By using curly brackets {}

```
int[] myArray = {1, 2, 3, 4, 5, 6}; // for integers
String[] myArray = {"A", "B", "C", "D", "E"} // for string
```

Scanner class

In Java, Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc and strings.

→ Using the SC in Java is the easiest way to read input in a Java Program, ~~though not very efficient~~.

→ An object of scanner class is created as

Scanner sc = new Scanner (System.in)

here an object of class Scanner named sc is created. & predefined object System.in is passed where system.in represents standard input stream.

→ Different methods of scanner class are

nextByte();	→ reads a Byte value from the user.
nextShort();	→ reads the Short " " " "
nextInt();	→ " " Integer " " " "
nextLong();	→ " " long " " " "
nextFloat();	→ " " float " " " "
nextDouble();	→ " " Double " " " "
nextLine();	→ " the String from the user until new line character is encountered.
next();	→ read the string value until the white space is encountered.

Scanner class

In Java, Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc and strings.

→ Using the SC in Java is the easiest way to read input in a Java Program, ~~though not very efficient~~.

→ An object of scanner class is created as

Scanner sc = new Scanner(System.in)

here an object of class Scanner named sc is created & predefined object System.in is passed where system.in represents standard input stream.

→ Different methods of scanner class are

nextByte(); → reads a Byte value from the user.

nextShort(); → reads the Short " " " "

nextInt(); → " " integer " " " "

nextLong(); → " " long " " " "

nextFloat(); → " " float " " " "

nextDouble(); → " " Double " " "

nextLine(); → " the String from the user until new line character is encountered.

next(); → read the string value until the white space is encountered.

next.charAt(0); → used to take a single character as input

Ex:

```
import java.util.Scanner;
class ScannerEx {
public static void main (String [] args)
{
Scanner sc = new Scanner (System.in);
System.out.println ("Enter the name :");
String name = sc.next~~Int~~Line ();
System.out.println (" Enter the roll no:");
int roll = sc.nextInt ();
System.out.println ("Enter the gender:");
char gender = sc.next().charAt(0);
SOP. (" Name :" + name);
S.O.P. (" Roll number : " + roll);
S.O.P. (" Gender : " + gender);
}
}
```

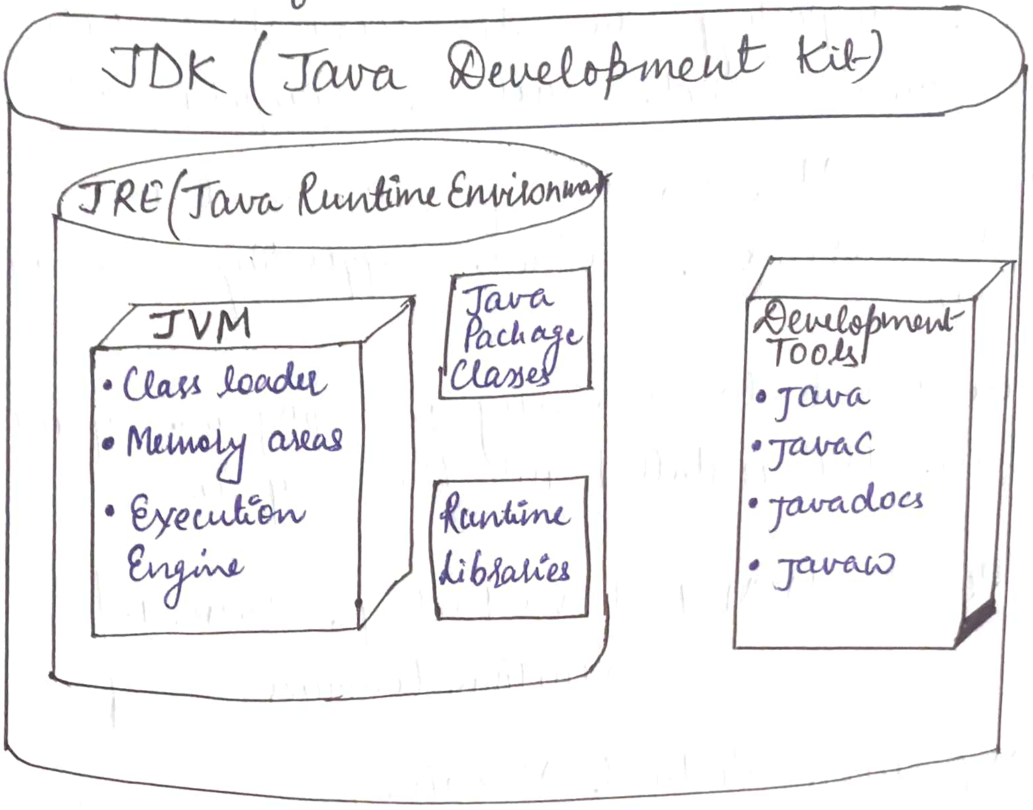
← this indexing of the element At(2)

O/p

```
Enter the name : XYZ
Name : XYZ
Enter the rollNo : 22
Roll Number : 22
Enter the gender : Female
Gender : female
```

JAVA

→ Architecture of JDK, JRE & JVM



JDK (Java Development Kit)

$\boxed{\text{JDK = JRE + Development Tools}}$

→ Java Developer Kit contains tools needed to develop the Java Programs, & JRE to run the programs.

→ The tools include compiler (javac.exe), Java application launcher (java.exe), Applet viewer, etc.

→ JDK is mainly targeted for java development. i.e. You can create a Java file (with the help of Java Packages), compile a Java file & run a java file.

JRE (Java RunTime Environment)

→ Java Runtime Environment contains JVM, class libraries, & other Supporting files. It does not contain any development tools such as compiler, debugger, etc.

→ Actually JVM runs the program, & it uses the class libraries, & other supporting files provided in JRE.

If you want to run any java program, you need to have JRE installed in the system.

$$\boxed{\text{JRE} = \text{JVM} + \text{Java Package Classes} + \text{RunTime Libraries}}$$

JVM (Java Virtual Machine)

JVM is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVM's are available for many h/w & s/w platforms. JVM is platform dependent because configuration of each OS differs & this makes Java Platform Independent.

JVM performs following main tasks:

→ Loads Code
→ Verifies Code
→ Executes Code
→ Provide runtime environment libraries