(4) Explain one dimensional and multi-dimensional array used in Java with suitable examples.

⇒ - Array is a collection of similar type of elements. Thus grouping of similar kind of element is possible using array.

- Typically arrays are written along with size of them

Syntax:

```
data_type array_name[size];
```

to allocate memory:

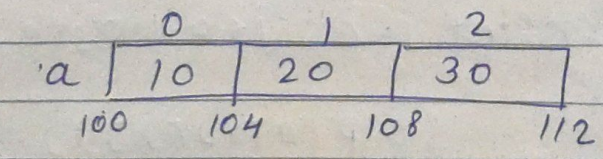
```
array_name = new datatype[size];
```

eg:

```
class A {
    public static void main(String s[]) {

        int[] a = new int[3];
        a[0] = 10;
        a[1] = 20;
        a[2] = 30;

        S.O.P("One dimensional array elements");
        S.O.P(a[0]);
        S.O.P(a[1]);
        S.O.P(a[2]);
    }
}
```



(ii) multidimensional Array:

- It means arrays of array. To declare a multidimensional array variable, specify each additional index using another set of square brackets.
- The two dimensional array are arrays in which elements are stored in rows as well as in columns.

for es-

rows \ column	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90

Syntax:

```
Datatype arrname[][] = new datatype[size][size];
```

es-

```
int a[][] = new int[3][3];
```

Program:

```
class multidimensional {
    public static void main(String[] s){

        int[][] a = { {0,1,2},
                      {3,4,5},
                      {6,7,8}
                    };

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

                S.O.P(a[i][j]);
            }
        }
    }
}
```

(12) Write a program to print area of circle by creating a class named 'Area' having two methods. First method named as 'setRadius' takes the radius of the circle as a parameter and the second method named as getArea returns the area of the circle. The radius of circle is entered through the keyboard.

⇒

```
import java.util.*;
class Area {

    private double radius;
    Float pi = 1.14f;
    public void setRadius(double r){
        radius = r;

    }

    public double getArea(){
        return pi * radius * radius;
    }
}

public class circle{
    public static void main(String s[]){

        Scanner sc = new Scanner(System.in);
        Area c = new Area();

        S.O.P("Enter Radius");
        double rad = sc.nextDouble();

        c.setRadius(rad);
        double area = c.getArea();
        S.O.P("Area of circle "+area);
    }
}
```

PAGE NO.
DATE:

Q. Describe Primitive data typer. List the primitive data typer in java & their respective storage capacity.

→

Data Typer:

Java is strongly typed language. It is used to specify type of data stored in variable. Depending upon this memory is allocated to that specefic variable.

Primitive data Typer:

- Primitive data typer are also called system defined data typer.

- There data typer are used to specify a variable into typer.

- memory is allocated accordingly.

There are 8 primitive data typer in java.

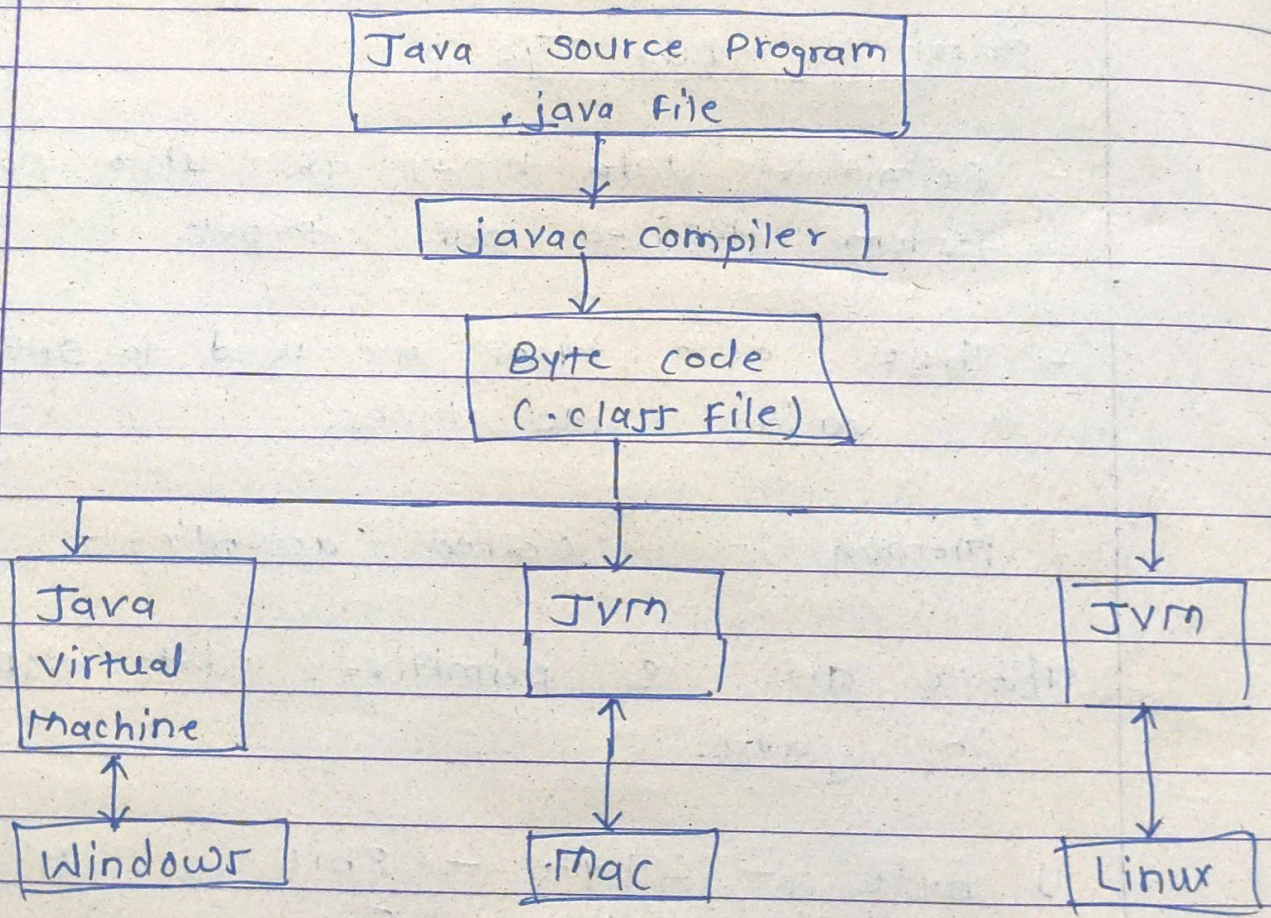
1) byte - 1 byte - 8 bit
2) short - 2 byte - 16 bit
3) int - 4 byte - 32 bit
4) long - 8 byte - 64 bit
5) float - 4 byte - 32 bit
6) double - 8 byte - 64 bit
7) char - 1 byte - 8 bit
8) boolean - 1 bit

Q. Write short notes on Java Virtual Machine (JVM) with diagram.

→

- JVM is a set of software & program components. As the name suggests, it is virtual computer that resides in real computer.

- Java can achieve its platform interdepen feature due to Java virtual machine.



PAGE NO.
DATE:

- When we want to write any Java Program, we write the source code & store it in a File. with extension .java.

- The .java file is compiled using javac & a class file gets created.

- The class file is byte code.

- The JVM takes byte code as input, reads it, interprets it, & then executes it.

- The JVM can generate an output corresponding to operating system.

PPL 3rd Unit

Q1]

a] Justify meaning of each characteristic of java in statement "java is simple, architecture, neutral, portable, interpreted and robust and secure programming language?

→

① * Java is simple and small programming language:-

- Java is simple programming language Even though you have no back ground of Programming you can learnt this language easily.
- The programmer who work C or C++ can learn this language more efficiently because syntax of java reassembles with C and C++.

② Java is Platform independent and portable programming language:-

- Platform independence is one of the important feature of java programming language that means the program of java can be run on various platform.

③ Java can be Compiled and Interpreted:-

- Normally programming language is either compiled or interpreted but java is both compiled and interpreted language.
- 1st our english language code is translated into byte code by compiler

- then that byte code is converted into machine language code.

④ Java is dynamic and extensible language :-

- This language is capable of dynamic linking of new class libraries, method and objects.
- It also support function written in C and C++ called Native method.

⑤ Java is robust and secure :-

- Following are reason due to which Java is robust and secure :-
  ① Java does not support concept of Pointer directely.
  ② memory management in Java is done automatically by garbage collector.
  ③ The output of Java is not executable code it is byte code.

② Define String in Java Programming? Explain following operation of class String in Java with example :-
  ① To find lenght of String
  ② To compare two String
  ③ Extraction of Character From String.

→ - String is a collection of Character - String is represented by using String class in Java which is in Java. lang Package.

- Sequence of character enclosed within double quote is called string.

1. Finding length of string:-
   - To find the lenght of string we use length() function

- Syntax:-

```
str.length();
```

- example:-

```
class str_length2
    psvm(String args[]){
      String str = new String("Gondh
                               arv");

      int a = str.length();
      S.o.println("lenght of String:"+a);
    }
}
```

2. Compare two String:-
   - Sometimes we need to check whether the two string are equal or not for that purpose we use equals() method which is of Boolean type.

syntax:-

- boolean equals(String str);

- example:-

```
class StringCompa2
    Public sVm (String args[]){
      String str1 = new String("INDIA");
      String str2 = new String("India");
```

```
if (str1.equals(str2)==true){
    S.O.P("String are equal");
}
else{
    S.O.P("String are not equal");
}
}
```

③ Extraction of Character From String :-

- As we know string is collection of character string class provide facility to extract the character from string object.
- we can exact the character using charAt(index).

* For example :-

```
String Fruit = new String("mango");
Char ch;
ch = Fruit.charAt(2);
S.O.O.P(ch);
```

③ Define constructor. Show the example about overlooding, default, parametrized and copy constructor.

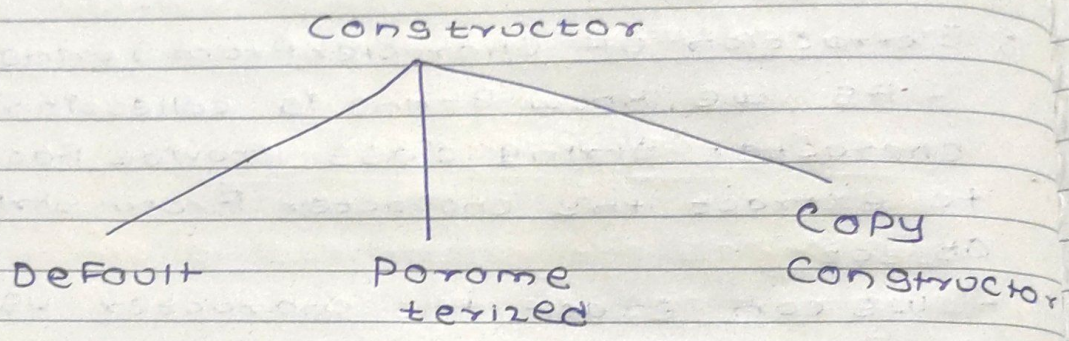
→

* Constructor :-

- Constructor is specialized method for initialized class object.
- The classname and name of constructor is same.

eg:- class test2
{
test()
{
}
}

- whenever object of class is created constructor invokes automatically.



1) Default constructor:-

- The constructor for which no argument or parameter is passed is called default constructor

eg:-

```
Public class test2
{
    int a;
    Test()
    {
        a=0;
    }
}
class main
{
    psvm (String args[])
    {
        Test obj = new Test();
    }
}
```

② Parametrized Constructor:-

- The parametrized constructor is a constructor for which the parameter is passed in constructor function.

For example:-

```
Public class Test{
    int a;
    Test(int x){
        a=x;
    }
}
class main{
    psvm (String args[]){
        Test obj = new Test(10);
    }
}
```

③ Copy Constructor:-

- Copy Constructor in Java is special type of constructor that creates a new object by copying value of existing object.

* features :

1. simple.
   - Java is designed to be easy to learn and use
   - It's syntax is stright forward and clean.
   - It remove pointer and operator overloading.

2. security :-
   - Java provide a secure execution environment.
   - It include security architecture that include bytecode verification
   - Bytecode verification ensure code safety

3. platform Independent :-
   - Java achives platform Independent through "write once Run Anywhere" capability.
   - Java program are compile into bytecode which platform independent internal.
   - It can work on any platform through JVM.

4. object-oriented :-
   - Java is fully object oriented language which means use object to represent data & method's to handle that data.
   - feature's like inheritance, polymorphism

abstraction are present in Java.

5. portability :
- Java portability is same from its platform independence.
- Java code compiled into byte code that can be executed on any JVM. doesn't consider harware or OS.
- This portability increase Java environment.

6. Architecture Nature
- Java is designed to be architecture neutral meaning that it does not rely on any specific machine architecture.
- This is achived through the use of the JVM which abstract or hide the internal things.
- Java compiler generate byte code that run on any machine.

7. Distrubuted :
- Java is designed with networking capabilities built into its core libraries making it ideal for building distrubuted applications.
- Java provide support for networking through Java.net. and RMI (Remote method Invocation)
- It help to communiate in network.

8. Interpreted :-
- Java code is initially compiled into bytecode which is then interpreted by the JVM
- This two-stage process allow Java programs to be highly portable and adaptable.
- The interpretation of bytecode at runtime provide flexibility.

9. Robust.
- Java increase reliability and robustness through several features.
- It include strong memory managm with automatic garbage collection, to prevent memory leaks.
- Java also provide exception handling to reduce runtime errors.

1. Grabage collector :-
- The Grabage collector in Java is responsible for automatically managing memory by free the memory which is not in use by program.
- Java program create objects that consume memory and when these object are no longer needed the garbage collector identify and remove ~~ten~~ them, to free space.
- It prevent memory leaks.

* Example.

```
public class A {
    public static void main(String[] args)
    {
        A obj = new A;
        obj = Null;
        system.gc();
    }
}
```

2. this
- In Java 'this' is a reference variable that refers the current object within an instance method or constructor.
- It is use for diffentiate instance variable and local variable.
- It iroke current object method's or pass current object as argument.

* Example :-

```
public class Demo{
    public int a;
    public Demo(int a){
        this.a = a;
    }
    public void display(){
        System.out.println(this.a);
    }
    public static void main(String[] args)
    {
        Demo D1 = new Demo(5);
        D1.display();
    }
}
```

3. final :

- The 'final' keyword is used with variable, method or classes.
- It means we cannot change the entity.
- A final variable can not be reasign
- A final method can not be overrid
- A final class cannot be subclass.

* Example :-

```
class final class Demo{
        final int a=10;
    final void display () {
            System.out.println(a);
    }
public static void main (String[] args){
    Demo obj = new Demo();
    obj.display();
}
}.
```

4. Static :
- Static keyword in Java indicates that a particular member belong to the class itself, rather than to intances of class.
- Static variable & methods can access without creating object.

* Example :

```
public class Demo {
    Static int a = 10;

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

5. finalize()

- The 'finalize()' method is called by garbage collector before an object is destroyed It allow the object to clean up resources before removing from memory.
- It is not commonly used due to problems.

* Example :-

```
public class Demo{

    protected void finalize() throws
     throwable{

        System.out.println("finallized
                            methed call"
    }

public static void main (string[] args)
{
    Demo obj = new Demo();
    obj = null;
    System.gc();
}
}
```

* Explain the Grabage collection (GC) in Java programming with example.

→

* Garbage Collection (GC)
  - Garbage collection (GS) in Java is an automatic memory managment process that helps in reclaiming memory occupied by objects that are no longer in use by the application.
  - this process is essential for preventing memory leaks and optimizing the usage of available memory.
  - It prevent memory leakage.

  - we have seen earlier an object is created in the memory using new operator,
  - Constructor is used to initialize the properties of that object
  - when an object is no more required. it must be removed from the memory so that memory can be reused for other objects.
  - In the language like c++ it is performed manually using destructor
  - Different approach taken by Java for it It handle deallocation automatically and this technique that completes this is called garbage collection.

* Example :-

```
public class Demo{
    int a=10;

    public static void main (String[]
                                args)
    {
        Demo obj = new Demo();
        obj = NULL;

        System.gc();
    }

}
```

Page No.
Date

Q.1] Different Access controls in Java.

→ Access control in java refers to the mechanisms used to restrict access to classes, methods and fields.

1. Private : Accessible only within same class other classes cannot access private members directly.

2. Public : Accessible from anywhere public members can be accessed by any other class

3. Default (Package-private) : Accesseble only with no access modifier are consider package-private.

4. protected : Accessible within the same package and subclasses subclasse in different packages can access protected members only if they subclass the class in which the members are declare.

* situation if you remove static modifier.
- If you remove the 'static' modifier from the 'main' method in java it means the method belongs to an instance of the classe rather than the class itself.
- But problem is the 'main' method is where java starts execution and it doesn't create objects to call 'main' so removing 'static' would not work.

Q. 2. Define string in Java

String - string is a sequence of characters represented by the 'string' class.
- It is immutable.

(i) To find length of string.
- "length()" method is used
- Example :-

```
String myString = "Hello"
int length = myString.length()  //5.
```

(ii) To compare two strings :
- equals() → true or false.
- compareTO() → 0 otherwise comparison
- Example :-

```
String str1 = "Hello";
String str2 = "word";

boolean equal = st1.equals(str2) //false.
```

(iii) extract character from string :-
- charAt() → get character of specefic index
- Example :

```
String str = "Hello";
char ch = str.charAt(1) //e
```

(iv) to concate two strings :-
- +
- concate()
- Example :

```
String str1 = "Hello";
String str2 = "word";

String str3 = str1+str2.
```

Q.3]

→ constructor()

- A constructor in Java is a special type of method that is automatically called when an object of a class is created.

1. Default constructor()

- This constructor is automatically created by Java if no other constructor is defined explicitly. It initialize object with default value.

- Example :

```
public class Person {
    string name;
    int age;

    public person() {
        name = "Pranav";
        age = 19;
    }
    public void display() {
        System.out.println("Name :" +name);
        System.out.println("Age :" + age);
    }
    public Static void main (string[] args){
        person p = new person();
        per p.display();
    }
}
```

2. Parameterized constructor():
- This constructor allows you to pass argument when creating an object, allowing you to initialize the object with specific values.
- Example.

```
public class Person {
        string name;
        int age;
    public person (string n, int a){
        name = n;
        age = a;
    }
    public void display () {
        System.out.println ("Name:" +name)
        System.out.println ("Age:" +age);
    }
    public static void main (string[] args ){
        person p = new person (pranav, 19);
        p.display();
    }
}
```

Q.4] Use of static variables & methods in Java. Restrictions that are declared static

→ Use of static variables:
- used to maintain common values accross all instance of the class They are initialized only once when class loaded on to memory
- we can access without creating object.

Page No.
Date

static method use:
- used for operation that do not require any instance speceific data They can be invoked without creating an instance of the class.

* Restrictions of static method.
1. cannot access instance variable
   - static method cannot access instance variables directly because they donot belong to any speceific instance of the class

2. can not use 'this' keyword:
   - they can not use 'this' keyword to refer to the current instance because they are not associated with any instance

3. can not be overriden:
   - They can not be overriden because they are implicitly final.

4. They cannot call Non-static Method.
   - static method can not call non-static method directly as they dont have access to instance -speceific data.

28] write short Note

→ i) LAMBDA Fuction
- In LISP the "LAMBDA" fuction is used to create anonymous fuctions.
- It allows you to define a fuction without giving it a name
- syntax :
  LAMBDA fuction syntax :
  (lambda (parameters) body)
  - where parameters are the input aguments. to the fuction and body is code.

ii) ARRAY :
- ARRAY in LISP are 1-dimensional sequence of elements
- They are used to store collection of data in a structured manner.
- In LISP arrays are indexed starting from 0
- It can hold any type of data.

(iii) Property List :
- Property Lists (or plists) are a data structure in LISP used to associate keys and values.
- they consists of list of alternating keys and value where each keys is followed by it corresponding value.
- mostly used to store symbolic data.

```java
//Write a program which receives n integers. Store the integers in an array. Program outputs the number of odd and even numbers present in this array.

import java.util.*;
class Program4{
	public static void main(String[] args){
		Scanner sc=new Scanner(System.in);
		System.out.print("Enetr Size : ");
		int size=sc.nextInt();
S
		int arr[]=new int[size];

		System.out.println("Enter Array elements : ");
		for(int i=0;i<arr.length;i++){
			arr[i]=sc.nextInt();
		}
		int ec=0;
		int oc=0;
		for(int i=0;i<arr.length;i++){
			if(arr[i]%2==0){
				ec++;
			}else{
				oc++;
			}
		}
		System.out.println("Even Count  = "+ec);
		System.out.println("Odd Count = "+oc);
	}
}
```

```java
//Write a program in java to perform addition of two matrices and set the diagonal elements of resultant matrix to 0.

import java.util.*;
class prg5{
	public static void main(String [] args){
		Scanner sc = new Scanner (System.in);
		System.out.print("Enter No of Row " );
		int row=sc.nextInt();
		System.out.print("Enter No of Column " );
		int col=sc.nextInt();
		int arr1[][]=new int [row][col];
		int arr2[][]=new int [row][col];
		int res[][]=new int [row][col];

		System.out.println("Enter Elements of Matrix 1 : ");
		for(int i=0;i<row;i++){
			System.out.print("Enter Elements for Row "+i+" : ");
			for(int j=0;j<col;j++){
				arr1[i][j]=sc.nextInt();
			}
		}
		System.out.println("Enter Elements of Matrix 2 : ");
		for(int i=0;i<row;i++){
			System.out.print("Enter Elements for Row "+i+" : ");
			for(int j=0;j<col;j++){
				arr2[i][j]=sc.nextInt();
			}
		}
		System.out.println("Resultant Matrix = ");
		for(int i=0;i<row;i++){
			for(int j=0;j<col;j++){
				if(i==j){
					res[i][j]=0;
				}else{
					res[i][j]=arr1[i][j]+arr2[i][j];
				}
				System.out.print(res[i][j]+"\t");
			}
			System.out.println();
		}

	}
}
```

```java
//Write a program in java using switch case statements to perform addition, subtraction, multiplication and division of given two numbers and print the result

import java.util.Scanner;
class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the first number:");
        double num1 = scanner.nextDouble();

        System.out.println("Enter the second number:");
        double num2 = scanner.nextDouble();

        System.out.println("Calculator ");
        System.out.println("1. Addition");
        System.out.println("2. Subtraction");
        System.out.println("3. Multiplication");
        System.out.println("4. Division");
        int choice = scanner.nextInt();

        double result;

        switch (choice) {
            case 1:
                result = num1 + num2;
                System.out.println("Result: " + result);
                break;
            case 2:
                result = num1 - num2;
                System.out.println("Result: " + result);
                break;
            case 3:
                result = num1 * num2;
                System.out.println("Result: " + result);
                break;
            case 4:
                if (num2 != 0) {
                    result = num1 / num2;
                    System.out.println("Result: " + result);
                } else {
                    System.out.println("Cannot divide by zero!");
                }
                break;
            default:
                System.out.println("Invalid choice!");
        }
    }
}
```