Practical No 1

Aim:- To study various Software do Perform Java application.

Practical No.1

Page No :
Date : 1 1

Name of Practical

Aim:- To study various software to perform Java application

Theory:-

1. What is "oop"?
2. What is "JAVA"?
3. various types of software and tools in Java and which are they?
4. State the features of tools and software in Java?

(1) What is oop?

→ OOPs - Object oriented programming System It's a programming ~~pap~~ paradigm that organices software design around data or objects rather than actions and logic, key principles includes Encapsulation, Inheritance, polymorphism and abstraction, which help in creating modular maintainable and reusable code oop languages, such as Java, c++ and python, use these principle to

Teacher's Signature

Page No :
Date : 1 1

Name of Practical

structure programs and facilitate efficient developement.

- Need of oop :-

- It make it easier to understand how a program works by bringing together data and it's behaviour in a single bundle called object.
- Enhanced the code structure and flexibility
- Increase code reusability.
- In oop the debugging is easy
- But it is slower.
- Works on bottom up approach.

(2) What is Java?

→ Java - Java is a high-level object. oriented programming language developed by Sun Microsystem (now-owned by oracle) It was first released in 1995 and has since becomes one of the most popular programming tag languages due to its platform

Teacher's Signature

Page No :
Date : 1 1

I N Name of Practical

independence, which means JAVA programs can run on any device that has Java virtual machine (JVM) Java is known for it's simplicity, versatility and robustness making it suitable for a wide range of applications, from web development to mobile apps and enterprise system.

Some key points of Java :-

(1) Java is primarily an object-oriented language. which means in it focuses on objects and data rather than just functions and logic.

(2) Portability :- Java programs are compiled into bytecode, which can be executed on any system with a JVM installed this makes Java application platform independent.

Teacher's Signature

Page No :
Date : 1 1

| N: Name of Practical

(3) Rich standard library - Java provides a comprechensive set of libraries for taks such as networking I/O, data structure and more.

(3) Various types of software and tools in Java and which are they?

→ Java, being versatile and widely used programming language has a variety of software and tools that cater to different aspects of development-testing.

1) Eclipse :-

A widely-used open source integrated development Enviroment (IDE) that ~~development~~ supports multiple programming languages.

- Features :-

* Plugin architecture.
* code refactoring tools
* Integrated debugging.

Teacher's Signature

Page No :
Date : 1 1

Name of Practical

Advantages :-
* Free and open soure
* large community support.

Disadvantages.

* can be resource - intensive

(2) Intelij IDEA.

Intelij IDEA is a highly regrarded IDE for java developement known for it's intelligent coding assitance and evgonomic's design

Features :-
* Smart code completion.
* on-the-fly code analysis.
* Refactoring tools

Advantages :-
* Excellent user interface.
* Advanced code navigation and search.

Teacher's Signature

Page No.
Date : 1 1

Name of Practical

Disadvantages
* can be heavy on system resources
* commercial version can be expesive.

(3) Net Beans
Net Beans is an Open-source IDE for JAVA and other languages, developed by the Apache Software Foundation.

Features :-
* project management tools
* code editing with sy

Teacher's Signature

Date :

Name of Practical

④ Junit :-

Junit is an source testing framework specifically designed for Java application. It provids a platform to write & run automatical tests, allowing developers to verify the correctness of their code of the unit level.

⑤. Linux :-

① Groovy :- Is a powerful language with static - type & compilation capabilities. It is aimed at improving develop productivity. The primary goal of this Groovy. is to increase the production speed.

Name of Practical

* Features :- dynamic typing, closures, easy object navigation & more compact syntax for working with lists & maps.

* Limitations :- It is a little slower than the many other object-oriented programming languages.

* Features of Junit :-

(1) It is an open-source framework for Java used to write and run test cases.

(2) It runs tests by providing test runners.

(3) It gives assertions to test expected results.

Teacher's Signature

Page No :
Date : / /

Name of Practical

(6) Mockito :- Mockito is an open source Java mocking & unit testing tools.
Mockito is a popular open sources Java mocking. framework. It helps developers to write well designed & loosely coupled code.

- Features of mockito :-

i) mockito allows writing of relaxed tests.

ii) seamlessly integrates with Junit.

Teacher's Signature

Page No :
Date : / /

Name of Practical

(7) Groovy :-

Groovy an object-oriented Programming language buit on the Java virtual machine has been Powerful yet often underrated player in the world of Programming language.

- Features of groovy :-

i) syntax highlighting

ii) code folding

iii) Add missing import

iv) Intellectool errors.

Teacher's Signature

Page No :
Date : / /

Name of Practical

(8) Jenkins :-

Jenkins is one of the top Develops tools because it is free, open-source and modulor and can integrate with pretty othero develops dool out there

- Features of Jenkins :-

i) Continious Integration.

ii) Intrastructure as Code.

iii) Monitoring and reporting.

Teacher's Signature

Page No :
Date : \ \

Name of Practical

(9) Echcache :- Is an open source Java distributed cache for general purpose caching, Java EE & light weight containers. Echcache was developed by Greg luck starting in 2003.

* Features :- Scales from in process caching, all the way to mixed in-process / out-of-process deployments with tera byte-sized caches.

* Limitations :- Ehcache is limited in size, eventually some data already in the cache will have to be removed to make room for new data that the application most recently accessed.

Teacher's Signature

Page No :
Date : / /

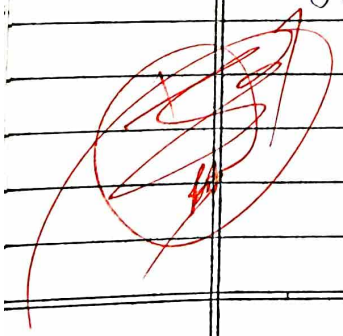
N. Name of Practical

(10) Visual IVM :- Visual IVM is a virtual tool for integrating command line JDK tools. It also offers light weight capabilities, It is design for both development & production time use.

* Features of visual IVM :-

(1) Display both local & remote Java Processes.
(2) visualize Process Threads.
(3) Help do analyze care Dumps.

Conclusion :- we studied various Software Java application do perform.



Teacher's Signature

Practical No. 2.

Page No :
Date : / /

Name of Practical

Aim :- Develop a Program to Print "Hello world" by using Java.

Theory :- To Print Java Hello world Program as follow.

1) Public :- It is an acess specifiers used to specify that code can be called from anywhere, main() is declared Public beause it is called by coder outside the class.

2) static :- It is declared static because it allows main() to be called without having to instentiate the class.

3) Void :- It does not return a value the keyword void simply tells the compiler that main() does not return anything back to the callers.

Teacher's Signature

Page No :
Date : / /

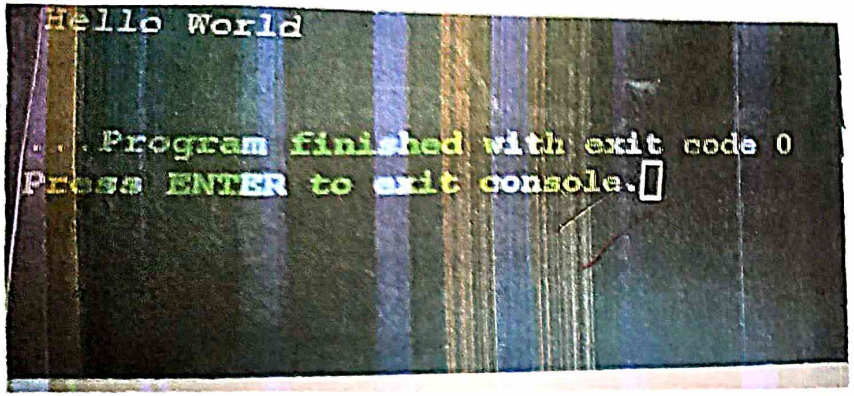
No Name of Practical

4) String args[] :- It holds optional command line arguments passed to the class through the Java command line.

Program code :-

```
class HelloWorld {
    // Program run under util //
Public static void main (String[] args)
// Public class has to be main for Code
{                                    to work //
    System.out.Println ("Hello world");
                    //write your code here //
    } // Print is the function is get
                                    output //
}
```

Teacher's Signature



Output :-

Hello World.

Conclusion :- we have succesfully runned the hello world Program by Java

Page No :
Date : / /

Name of Practical

Conclusion :-

we have succesfully runned the hello world Program by Java.

Teacher's Signature

Page No :
Date : 1 1

Practical No 3

Name of Practical

Aim :- Develop a program to study implement looping statement belonging to Java.

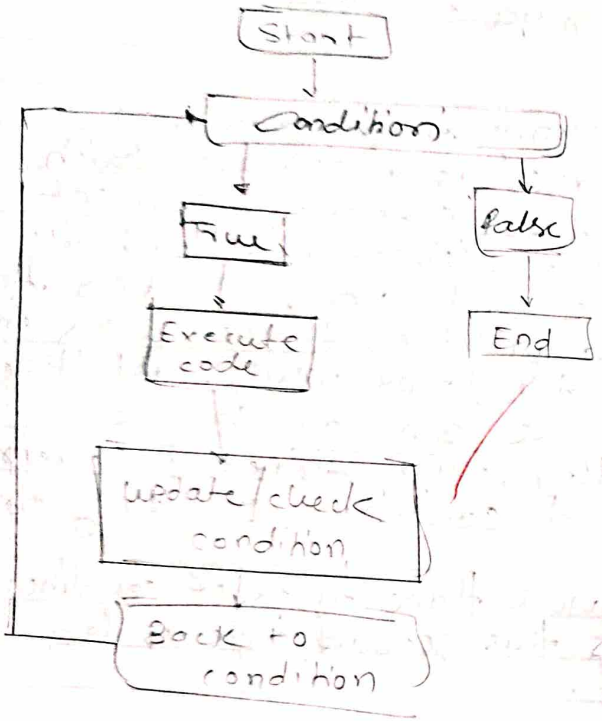
Theory :-

looping in programming languages is a feature which facilities the execution of a set of instructions/function repeatedly while some condition evaluates to true. Java provides three ways for executing the loops while. all the ways provide similar basic functionality, they differ in their syntax. and condition checking time.

Java provides three types of conditional statements this second type is loop statement.

Teacher's Signature

Flow chart for a while loop :-



Page No :
Date : / /

Name of Practical

(1) While loop :-

A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement

Syntax :-

```
while (boolean condition)
{
  loop statements. . .
}
```

Teacher's Signature

```java
public class WhileLoopExample
{
    public static void main(String[] args)
    {
        int i = 0; // Initialization

        // While loop that prints numbers from 1 to 5
        while (i < 5)   // Condition
        {
            System.out.println(i); // Body
            i++; // Update
        }
    }
}
```

```
java -cp /tmp/Sopomankpm/WhileLoopExample
0
1
2
3
4

=== Code Execution Successful ===
```

output :-

0
1
2
3
4

Page No :
Date : / /

Name of Practical

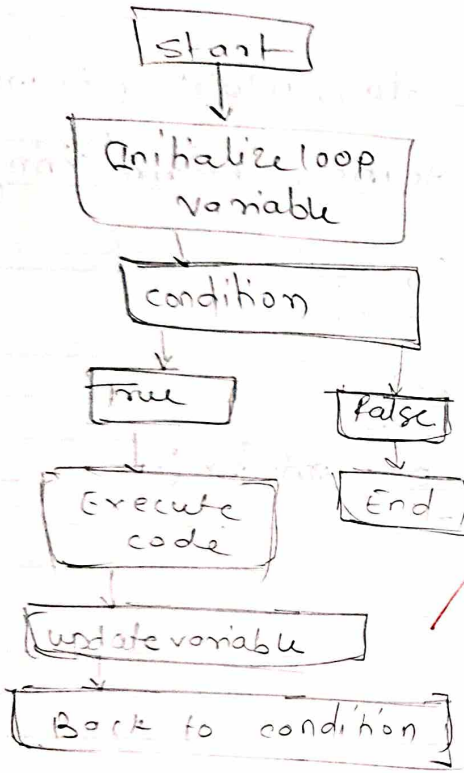
Program :-

(2) while loop

```java
public class whileloopExample
{
public static void main(string args[])
{
    int i=0; // Initialization
// while loop that prints numbers 1 to 5
  while (i<5) // condition
{
    system.out.println(i); // body
      i++; // update
}
}
}
```

Teacher's Signature

(2) Flowchart for a for loop:-



Page No :
Date : / /

Name of Practical

(2) For loop:-

For loop provides a concise way to writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition and increment/decreament in one line thereby providing a shorter, easy to debug structure of looping.

Syntax:-

```
for (initialization condition;
      testing
condition; increment / decrement)
{
    statement (s)
}
```

Teacher's Signature _____

```java
public class ForLoopExample
{
    public static void main(String[] args)
    {
        // A for loop that prints numbers from 1 to 5
        for (int i = 1; i <= 5; i++)
        {
            System.out.println(i);
        }
    }
}
```

Output

```
java -cp /tmp/tfzK8o2xBl/ForLoopExample
1
2
3
4
5

=== Code Execution Successful ===
```

output :-

1
2
3
4
5

Page No :
Date : / /

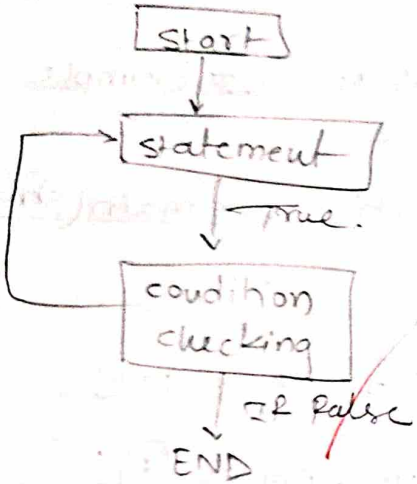
Name of Practical

Program :-
For loop :-

```
Public class for loop Example
        // class to be declared
{
Public static void main(string[]args)
{
    // A for loop that prints no from
        1 to 5
    for(int i=1; i<=5; i++)
{
        system.out.println(i); // body.
    }
}
}
```

Teacher's Signature

Flowchart :-



Page No :
Date : 1 1

Name of Practical

3) do while :-

do while loop is similar to while loop with only difference that it checks for condition after executing the statement and therefore is an example of Exit control loop.

Syntax :-

```
do
{
   statements:-
}
while (condition);
```

Teacher's Signature

```java
public class DoWhileLoopExample
{
    public static void main(String[] args)
    {
        int i = 1; // Initialization

        // Do-while loop that prints numbers from 1 to 5
        do
        {
            System.out.println(i); // Body
            i++; // Update
        } while (i <= 5); // Condition
    }
}
```

```
java -cp /tmp/lzHSVLKPgD/DoWhileLoopExample
1
2
3
4
5

=== Code Execution Successful ===
```

Output :-

1
2
3
4
5.

Conclusion :- We have studied looping statement in Java.

Name of Practical

(3) Program :-

3) Do while

```
public class DoWhileLoopExample
{
public static void main(String[]args)
{
    int i=1; // Initialization.

// Do-while loop that prints numbers 1 to 5
    do
{
    system.out.println(i);
        i++;
{
    while (i<=5);
{
{
```

Conclusion :- We have studied looping statement in Java.

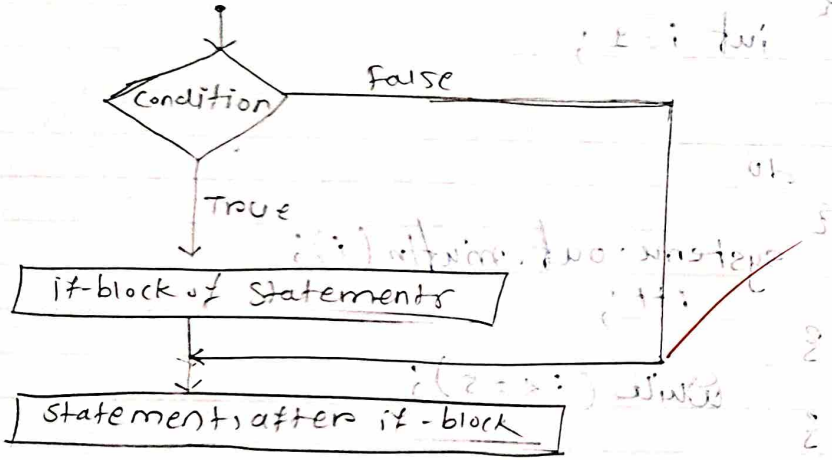
Page No :
Date : \ \

Teacher's Signature

Practical No 04.

Aim:- Develop a Program to study & impliment selection statements belonging to JAVA.

- Flow of execution :-



Syntax :-

if (condition)

Page No :
Date : 1 1

Practical No4.

Name of Practical

Aim :- Develop a program to study & impliment selection statments belonging to JAVA.

Theory :-

The selection statements in JAVA as.

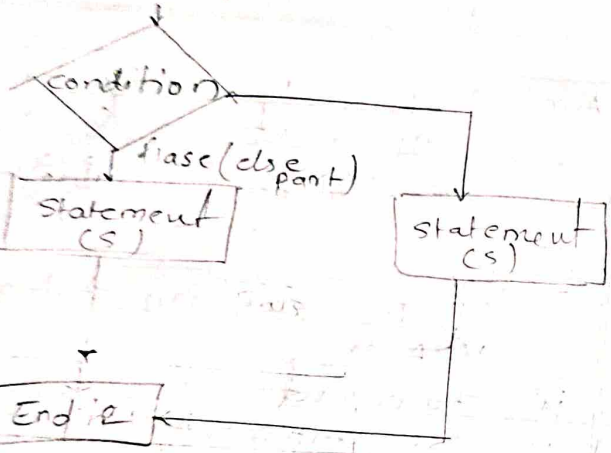
1) if statement
2) If-else statement
3) Nested if statement
4) if-else-if statement
5) switch statement

1) If statement :-

In Java, we use the if statement to test a condition and decide the execution of a back of statement based on that cond^n results. The if sd statements checks the given cond^n the decides the execution of a block of statements. If the cond^n is true, then the block of statements is executed and if it is false, then statements is ignored

Teacher's Signature

• Flow of execution :-



• Syntax :-

```
if (condition) {
    true-block of statements;
    ....
}
else {
    false-block of statements;
    .....
}
    statement after if-block;
```

Page No :
Date : / /

Name of Practical

(2) If-else statement :-

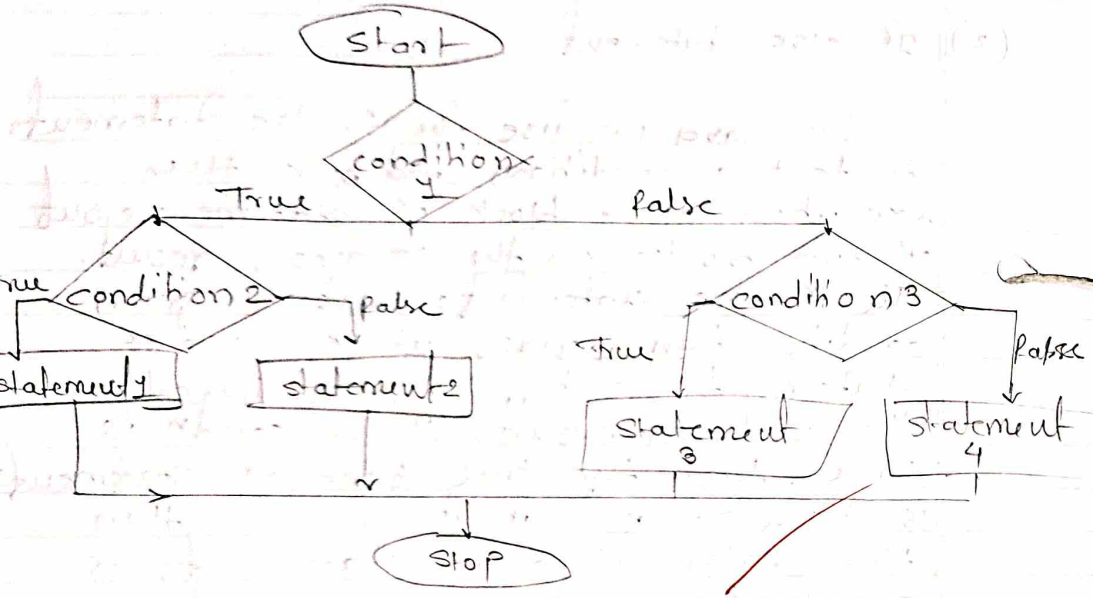
In Java we use the if-else statements to test a condition and pick the execution of a block of statements out of two blocks on the condition result. The if-else statements check the given condition then decides which block of statements to be executed. based on the condition result. If the condn is True, then the true block of statements is executed and if it is false, then the false block of statements is executed. The syntax and execution flow of if-else statement is as follows

• Syntax :-

```
if (condition){
        true-block of statements;
    ...}
else {
        false-block of statements;
    ...}
    statement after if-block;
```

Teacher's Signature

- Flowchart :-



Page No :
Date : 1 1

Name of Practical

(3) Nested if statement :-

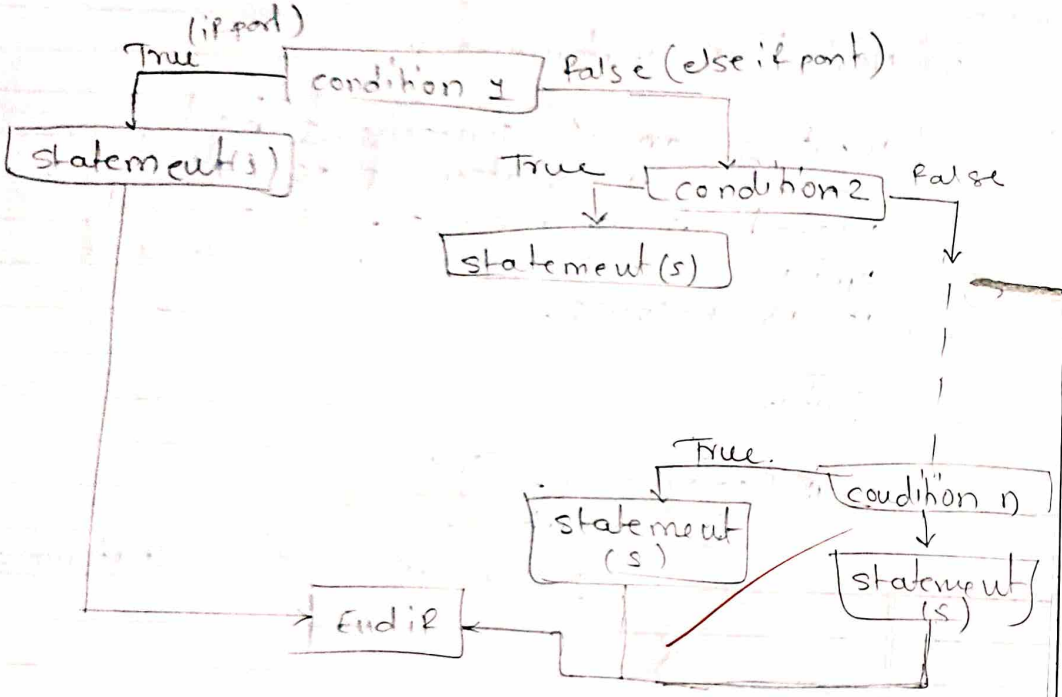
Writing an if statement inside another if-statement is called nested if statement. The general syntax of the nested if-statement is as follows.

Syntax :-

```
if (condition-1) {
    if (condition-2) {
        inner if-block of statements;
    }
    ...
}
```

Teacher's Signature

Flowchart:-



Page No :
Date : / /

Name of Practical

4) if-else if statement:-

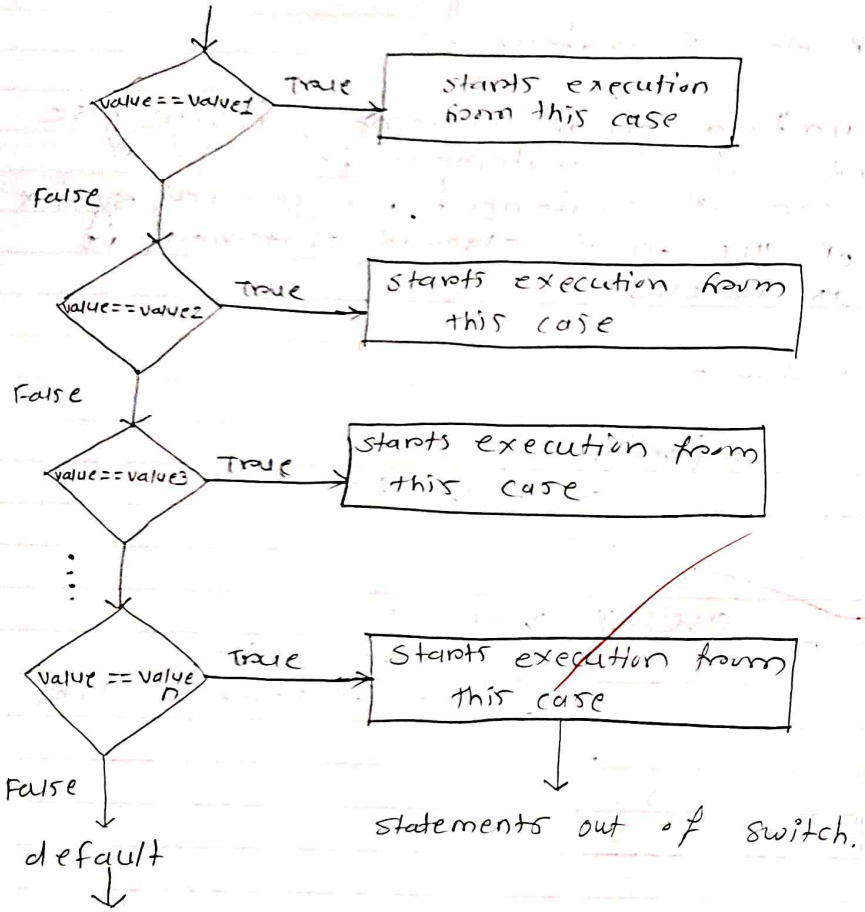
Writing an if-statement inside else or an if statement is called if-else if statement. The general syntax of the an if-else-if statement is as follows:-

syntax:-

```
if (condition-1){
    condition-1 true-block
    . . . .
}
else if (condition-2){
    condition-2 true-block;
    condition-1 false-block too;
    . . . .
}
```

Teacher's Signature

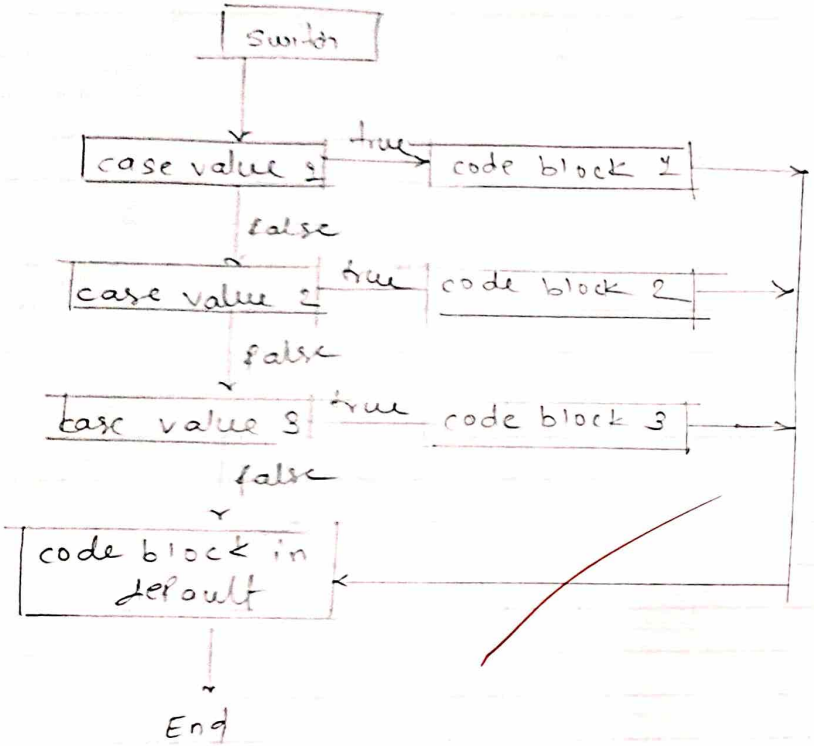
· Flow Diagram ·—



Page No :
Date : / /

Name of Practical

Teacher's Signature



Page No :
Date : 1 1

Name of Practical

5) switch statement in JAVA :-

Using the switch statement one can select only one option from more number of option very easily. In the switch statement. we provide a value that is to be compared with a value associated with each option. Whenever the given value matches the value associated with an option, the execution starts from that option. In the switch statement, every option is defined as a case.

⊙ Syntax :-

```
switch (expression or value)
{ case value 1 :- set of statement ;
  case value 2 :- set of statement ;
  case value 3 :- set of statement ;
  case value 4 :- set of statement ;
  ...
    default :- set of statements ;
}
```

Teacher's Signature

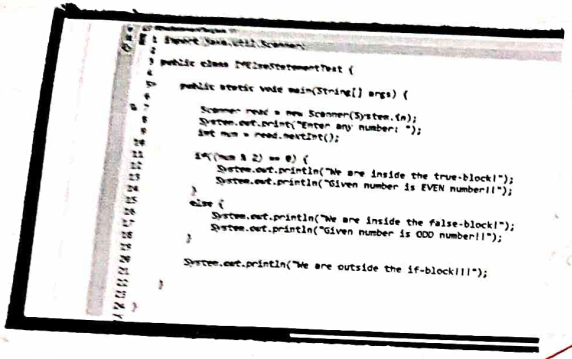
Page No :
Date : 1 1

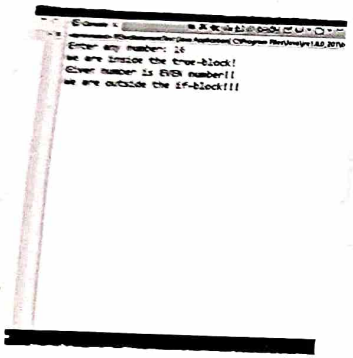
Name of Practical

1) if statement :-

```
import java.util.scanner;
public class IfStatementTest {//program using selection statement
public static void main (string[] args) {
// Declaration of Public class
Scanner read.= new scanner (system.in);
system out. print ("Enter any number:");
int num = read.nextInt ();

if ((num % 5) } == 0 {
    system.out.println("we are inside the
                        if-block!");
system.out.println("Given number is
                    divisible by 5!!");
}

    system.out.println("we are outside the
                        if-block!!!");
  }
} // Conventional method to display
                        output.
}
```

Teacher's Signature





Output :-

Enter any number : 16
We are inside the true-block
Given number is EVEN number!!
We are outside the if-block!!!

Page No :
Date : / /

Name of Practical

2]. if-else Statement :-

```
import java.util.Scanner;
Public class IfElseStatementTest {   //program to find numb er odd are even
Public static void (string [] args) {
  Scanner read = new Scanner (System.in);
  System.out.Print("Enter any number:");
   Int num = read.nextInt();
if ((num %.2) = = 0) {   //* main is the starting point of program
System.out.Println("we are inside the true block!");
System.out.Println ("Given number is EVEN
                                    Number!!");
}
  else {
     System.out.Println("we are inside the
                                    false-block!");
systemout.Println("Given number is ODD number!!");
}
   systemout.Println("we are outside the
                                    if-block!!!");
   } // Print method.
}
```

Teacher's Signature _____

```java
import java.util.Scanner;

public class NestedIfStatementTest {

    public static void main(String[] args) {

        Scanner read = new Scanner(System.in);
        System.out.print("Enter any number: ");
        int num = read.nextInt();

        if (num < 100) {
            System.out.println("\nGiven number is below 100");
            if (num % 2 == 0)
                System.out.println("And it is EVEN");
            else
                System.out.println("And it is ODD");
        } else
            System.out.println("Given number is not below 100");

        System.out.println("\nWe are outside the if-block!!!");
    }
}
```

```
Enter any number: 75

Given number is below 100
And it is ODD

We are outside the if-block!!!
```

Output:-

Enter any number: 75
Given number is below 100
And it is ODD
We are outside the if-block !!!

Page No :
Date : 1 1

Name of Practical

3). Nested if statements :-

```java
import java.util.Scanner; //declaration of Public Class
public class NestedIfStatement Test {
 public static void main (String [] args) {

 Scanner read = new Scanner (System.in);
 system.out.print ("Enter any number;");
 int num = read.nextInt();

 if (num < 100) {
    System.out.println ("\n Given number is
                         below 100");
 if (num %2 == 0) //code executed if condition is true
    system.out.println("And it is Even");
 else
    system.out.println ("Add it is ODD");
 } else // executed if condition is false
    system.out.println ("Given no. is not
                         below 100");
 system.out.println ("\n we are outside the
                      if-block !!! ");
 } // Print Statement.
```

Teacher's Signature

```
import java.util.Scanner;

public class IfElseIfStatementTest {

    public static void main(String[] args) {

        int num1, num2, num3;
        Scanner read = new Scanner(System.in);
        System.out.print("Enter any three numbers: ");
        num1 = read.nextInt();
        num2 = read.nextInt();
        num3 = read.nextInt();

        if( num1>num2 && num1>=num3)
            System.out.println("\nThe largest number is " + num1) ;

        else if (num2>=num1 && num2>=num3)
            System.out.println("\nThe largest number is " + num2) ;

        else
            System.out.println("\nThe largest number is " + num3) ;

        System.out.println("\nWe are outside the if-block!!!");
    }
}
```

```
Enter any three numbers: 17 39 14

The largest number is 39

We are outside the if-block!!!
```

Output:-

Enter any three numbers: 17 39 14

The largest number is 39

We are outside the if-block!!!

Page No :
Date : 1 1

Name of Practical

4. if-else if statements:-

```
import java.util.Scanner; //switch case example
public class IfElseIfStatementTest {
//Declaration of public class
public static void main(String[] args) {

int num1, num2, num3;
Scanner read = new Scanner(System.in);
System.out.print("Enter any three no:");
num1 = read.nextInt();
num2 = read.nextInt();
num3 = read.nextInt();
if (num1>num2 && num1>=num3)
    System.out.println("\nThe largest no is"
                + num1);
else if (num2>=num1 && num2>=num3)
    System.out.println("\nThe largest no is"
                + num2);

else // code executed.
    System.out.println("\nThe largest no is"+num3);
System.out.println("\nWe are outside the
                if-block!!!");
} // Print Statement end
}
```

Teacher's Signature

```
import java.util.Scanner;

public class SwitchStatementTest {

    public static void main(String[] args) {

        Scanner read = new Scanner(System.in);
        System.out.print("Press any digit: ");

        int value = read.nextInt();

        switch( value )
          {
            case 0: System.out.println("ZERO") ; break ;
            case 1: System.out.println("ONE") ; break ;
            case 2: System.out.println("TWO") ; break ;
            case 3: System.out.println("THREE") ; break ;
            case 4: System.out.println("FOUR") ; break ;
            case 5: System.out.println("FIVE") ; break ;
            case 6: System.out.println("SIX") ; break ;
            case 7: System.out.println("SEVEN") ; break ;
            case 8: System.out.println("EIGHT") ; break ;
            case 9: System.out.println("NINE") ; break ;
            default: System.out.println("Not a Digit") ;
          }

    }
```

```
Press any digit: 8
EIGHT
```

Output :-

press any digit : 8
EIGHT

Page No :
Date : / /

Name of Practical

5). switch statements :-

```
import java.util.scanner ;//switch case example.
Public class switchStatementTest {
public static void main(string[] args) {
 scanner read = new scanner(system.in);
 System.out.Print("Press any digit:");
 int value = read.nextInt();
  switch (value)
  {
  case0 : system.out.Println("ZERO"); break;
  //executed if variable equals to0
  case1 : System.out.println("ONE"); break;
  //executed if variable equals to1
  case2 : system.out.Println("TWO"); break;
  //executed if variable equals to2
  case3 : System.out.println("THREE"); break;
  //executed if variable equals to3
  case4 : system.out.Println("FOUR"); break;
  //executed if variable equals to4
  case5 : system.out.Println("FIVE"); break;
  //executed if variable equals to5
  case6 : system.out.Println("SIX"); break;
  //executed if variable equals to6
  case7 : system.out.Println("SEVEN"); break;
  //executed if variable equals to7
  case8 : system.out.println("EIGHT"); break;
  //executed if variable equals to8
  case9 : system.out.Println("NINE"); break;
  //executed if variable equals to 9
  }

 } // Print statement.

}
```

Teacher's Signature

- Conclusion :- We studied four types of selection statements in Java and perform it using Java.

Page No :
Date : 1 1

Name of Practical

- Conclusion :- We studied four types of selection statements in Java and perform it using Java.

Teacher's Signature

Practical No. 5

Aim:- Develop a program to demostrate the concept of class method and objects.

Name of Practical

Practical No 5

Page No :
Date : 1 1

Aim:- Develop a program to demostrate the concept of class, method & objects

Theory:- In Java, classes and objects are basics concepts of objects oriented programming (OOP's) that are used to represent real-world concepts and entities. The class represents a group of objects having similar properties & behaviour.

For example:- The animal type Dog is a class while a particular dog named Tommy is an object of the Dog class.

Teacher's Signature _____

class Declaration in Java

```
access-modifier class
  <class_name>
{
   data member;
   Method;
   Construction;
   Nested class;
   interface;
}
```

Page No :
Date : / /

Name of Practical

1) Java classes:-
A class in Java is a set of objects which shares common characteristics / behavior and common properties / attributes. It is a user-defined blueprint or prototype from which objects are created. For ex student is a class while a perticular student named Ravi is an object.

- Properties of Java classes:-

(1) class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.

(2) class does not occupy memory.

(3) class is a group of variables of different data types and a group of methods.

Teacher's Signature

Page No :
Date : 1 1

Name of Practical

(2) Java objects :-

An objects in Java is a basic unit of object-oriented programming and represents real-life entities, objects are the instances of a class that are created to use the attributes and methods of a class. A typical java program creates, many objects, which as you know interact by invoking methods.

(1) properties :-

(4) State :- It is represented by attributes of an object. It also reflects the properties of an objects:

(2) Behavior :- It is represented by the methods of an object. It also reflects the response of an object with other objects.

Teacher's Signature

```java
// Java Program for class example
class Student {
    // data member (also instance variable)
    int id;
    // data member (also instance variable)
    String name;

    public static void main(String args[])
    {
        // creating an object of
        // Student
        Student s1 = new Student();
        System.out.println(s1.id);
        System.out.println(s1.name);
    }
}
```

Output 1

```
0
null
```

Output 2

```
GeeksForGeeks
```

Output 3

```
GeeksForGeeks
```

O/P1:-
0
null
Output2:-
GeeksforGeeks
Output3:-
GeeksforGeeks.

Page No :
Date : 1 1

Name of Practical

(3) Identity:- It gives a unique name to an object and enables one object to interact with other objects.

```
// Java program for class example
class student{
//data member(also instance variable)
 int id;
//data member(also instance variable)
 string name;

 public static void main (string args[])
 {
        //creating an object of
        // student
 student s1 = new student();
 system.out.println (s1.id);
 system.out.println (s1.name);
 }
}
```

Teacher's Signature

Page No :
Date : 1 1

N Name of Practical

```
//class Declaration
public class Dog{
        //Instance variable
    string name;
    string breed;
    int age;
    string color;

    //constructor Declaration of class
Public Dog(string name, string breed, int age
            string color)
{
    this.name = name;
    this.breed = breed;
    this.age = age;
    this.color = color;
}

public string getName(){return name;}
public string getBreed(){return breed;}
public int getAge(){return age;}
public string getColor(){return color;}
```

Teacher's Signature

```java
// Class Declaration
public class Dog {
    // Instance Variables
    String name;
    String breed;
    int age;
    String color;

    // Constructor Declaration of Class
    public Dog(String name, String breed, int age,
               String color)
    {
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }

    // method 1
    public String getName() { return name; }

    // method 2
    public String getBreed() { return breed; }

    // method 3
    public int getAge() { return age; }

    // method 4
    public String getColor() { return color; }

    @Override public String toString()
    {
        return ("Hi my name is " + this.getName()
                + ".\nMy breed,age and color are "
                + this.getBreed() + "," + this.getAge()
                + "," + this.getColor());
    }

    public static void main(String[] args)
    {
        Dog tuffy
            = new Dog("tuffy", "papillon", 5, "white");
        System.out.println(tuffy.toString());
    }
```

Output

```
Hi my name is tuffy.
My breed,age and color are papillon,5,white
```

utput :-
my name is tuffy
breed, age and color are papillon,
5, white

Page No :
Date : / /

Name of Practical

@ Override public string toString()
{
return ("Hi my name is" + this.getName()
+ ".\n my breed, age and color are" +
this.getBreed() + "," + this.getAge()
+ "," + this.getColor());
}
public static void main(String[]args)
{
Dog tuffy.
= new Dog("tuffy", "papillon", 5, "white");
System.out.println(tuffy.toString());
}
}

Teacher's Signature

Page No :
Date : / /

Name of Practical

```
//Define the class
class car {
// Instance variables (attributes)
String model;
String color;
int year;
// constructor to initialize the object
car (String model, String color, int year) {
this.model = model;
this.color = color;
this.year = year;
    // method to display car details (object method)

void displayDetails () {
System.out.println("car model: " + model);
system.out.println("Car color: " + color);
system.out.println("Car year: " + year);
} // class method (static method)

static void showWelcomeMessage () {
System.out.println("Welcome to the car
                    showroom!");
}

}
```

Teacher's Signature _____

output :-

```
Welcome to the Car showroom!
Car model : Toyota Camry
Car Color : Red
Car Year : 2020
```

Conclusion :-

Hence we have perform to demosh class object and method in Java to print required output.

Name of Practical

Date : / /

```
//main class to test the Car Class
Public class main{
Public static void main (String[] args){
//call the class method (without creating
                        an object)
Car.ShowWelcomeMessage();
//Create an object of the Car class
Car myCar = new Car("Toyota Camry", "Red",
                        2020);

// call the object method to display details.
  myCar.displayDetails();
  }
}
```

Conclusion :-

Hence we have perform to demostrate class object and method in Java to print required output.

Teacher's Signature

Practical No. 6

Aim:- Develop a program to study of implement the concept of method overloading.

Page No :
Date : 1 1

Practical No. 6

Name of Practical

Aim :- Develop a program to study & implement the concept of method overloading.

Theory :- Java provides a powerful feature called "method overloading", which allows programer's to define multiple methods with the same name but diff parameter. This makes the code readable and provides flexibility and convenience when working diff datatypes or combination of input whether a beginner or an experienced Java developer, understand method overloading can greatly enchance your programming skills & help. You write more efficient and effective code.

Teacher's Signature

```
class Method_Overloading {
    double figure(double l, int b) { //two
        parameters with double type
    {
        return (l*b);
    }
    float figure(int s) { //one parameter with
        float return type
    {
        return (s*s);
    }

    public static void main(String[] args) {
        Method_Overloading obj = new
            Method_Overloading();
        System.out.println("Area of Rectangle:
            "+obj.figure(5.55, 6));
        System.out.println("Area of Square:
            "+obj.figure(3));
    }
}
```

```
javac -cp /tmp/PTK9DCZaPQ/Method_
Area of Rectangle: 33.3
Area of Square: 9.0

==> Code Execution Successful ==
```

Output:-

Area of Rectangle: 33.3

Area of square: 9.0

Page No :
Date : / /

Name of Practical

Program code:-

```
// A simple mode up the program to
   study method overload.
class method - overloading {
    double figure (double l, int b)
// two parameters with double type.
{
    return (l*b);
}
float fig. (int s) // one parameter with
                   float return type.
{
    return (s*s);
}
Public static void main (string [] args){
    method - overloading obj = new
    method - overloading ();
    system.out.println ("Area of Rectangle: "
        + obj. figure (5.55, 6));
    system.out.println ("Area of square: "
        + obj.figure (3));
}
}
```

Teacher's Signature

Conclusion :- Thus we have develop a program to study & implement the concept of method overloading.

Page No :
Date : / /

Name of Practical

Conclusion :- Thus we have develop a program to study & implement the concept of method overloading.

Teacher's Signature

Practical No. 7

Aim:- Develop a program to study & implement the concept of constructor.

Practical No. 7

Page No :
Date : 1 1

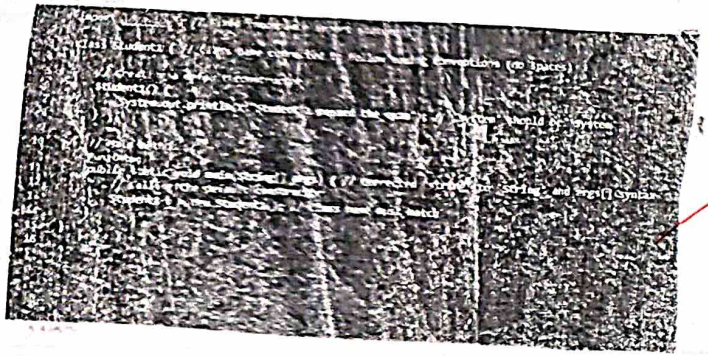
Name of Practical

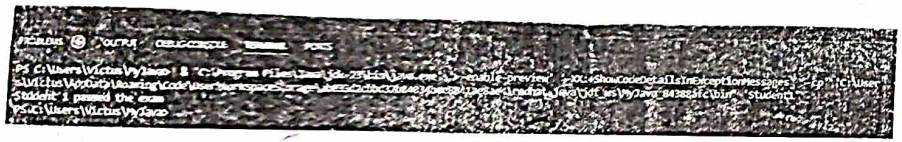
Aim :- Develop a program to study & implement the concept of constructor.

Theory :-

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor memory for the object is allocated in the memory. There are two type of constructor. There are two type in Java Default constructor and Parameterized constructors. It is a special type of method which is used to initialize the object. It calls a default constructor if there is no constructor available in the class. In such case, Java Compilers Provide a default constructor by default.

Teacher's Signature _____





output :- student1 passed the exam.

Conclusion :- Thus we have studied and implement the concept of constructor.

Page No :
Date : \ \

Name of Practical

Program code :-

```
import java.util.*;

class student1 {
// creating a default constructor
student1()
  { system.out.Println("student1 passed
                        the exam"); }
   // main method
Public static void main (string args[])
  {
  //calling a default constructor.
    student1 b = new student1();
    }

}
```

conclusion :- Thus we have studied and implement the concept of constructors.

Teacher's Signature

Practical No. 08.

Aim:- Develop a program to study and implement constructor overloading in Java.

Page No :
Date : / /

Practical No. 8

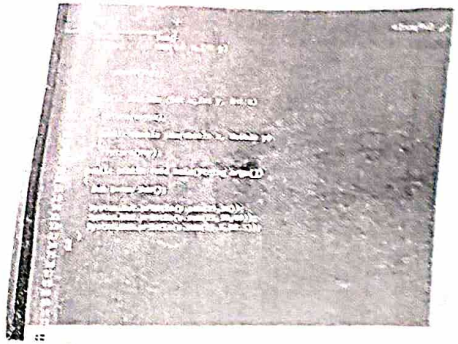
Name of Practical

Aim:- Develop a program to study and implement constructors overloading in Java.

Theory :-

In Java, we can overload constructors like methods the constructors overloading can be defined as the concept of having more than one constructors with different parameters so that every constructors can Deform a different task. Constructors for a class have the same name as that of the class but they can have different signature constructors are differentioned on the basis of argument passed to them.

Teacher's Signature



utPut :-
83
152
840

clusion :- Thus we have studied & implemented the concept of constructors overloading.

Page No :
Date : / /

Name of Practical

```
Public class sum { // class is Sum
public int sum (int x, int y)
{
  return (x+y); // returning the sum
                   of x and y
}
 Public int sum (int x, int y, int z)
{
   return (x+y+z);
}
 Public double sum (double x. double y)
{
   return (x+y);
}
 public static void main (String args[])
{
   Sum s = new Sum();
   // object creation
System.out.println(s.sum (25,58));
System.out.println(s.sum (25, 58, 69));
System.out.println (s.sum (25.5, 58.5));
}} // printing Sum.
```

Conclusion :- Thus we have studied & implemented the concept of constructor overloading.

Teacher's Signature

Practical No. 09

Aim :- Develop a program to study and implement arrays in Java.

Name of Practical

Page No :
Date : / /

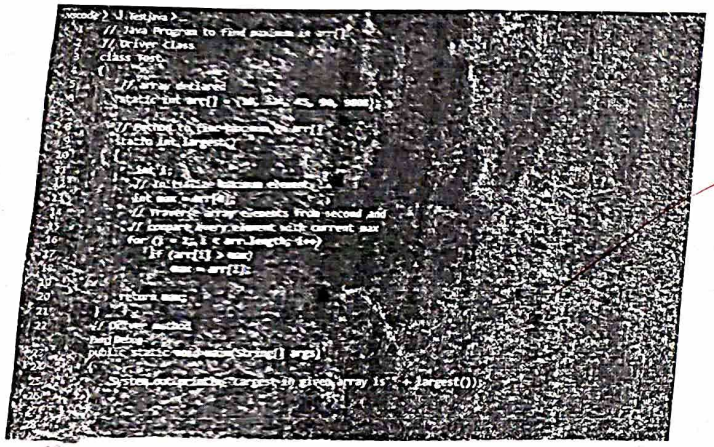
Practical No. 9

Aim :- Develop a program to study and implement arrays in Java.

Theory :-

An array is a data structure consisting of a collection of elements of the same memory size, each identified by at least one array index or key. An array is a linear data structure that stores similar elements that are stored in contigues memory locations. This article provides a variety of programs on arrays, including examples of operations such as sorting, merging, insertion, and deletion of elements in a single-dimensional array.

Teacher's Signature _____



Page No :
Date : / /

Name of Practical

Program code :-

```
// Java Program to find maximum in arr[]
// Driver class
class Test
{
    // array declared
    static int arr[] = {10, 324, 45, 90, 9808};
// method to find maximum in arr[]
static int largest()
{
    int i;
// Initialize maximum element
    int max = arr[0];
// Traverse array elements from second and
// compare every element with current max
for (i=1; i<arr.length; i++)
    if (arr[i] > max)
        max = arr[i];
return max;
}
// Driver method.
```

Teacher's Signature

o/p :- Largest in given array is 9808

Conclusion :- Thus we have studied and implement the concept of array in Java.

Page No :
Date : / /

Title of Practical

```
public static void main (String [] args)
{
    System.out.Println ("Largest in given
        array is " + largest ());
}
}
```

Conclusion :- Thus we have studied and implement the concept of array in Java.

Teacher's Signature

Page No :
Date : / /

Practical No. 10

e of Practical

Aim :- Develop a Program to study and implement the inheritance in Java.

Theory :-

Java, Inheritance is an important pillar of oop. It is the mechanism is Java by which one class is allowed to inherit the features of another class. In Java, Inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of the class. In addition, you can add new fields and methods to your current class as well.

Teacher's Signature _____

Page No :
Date : / /

Name of Practical

- Important Terminologies used in Java Inheritance.

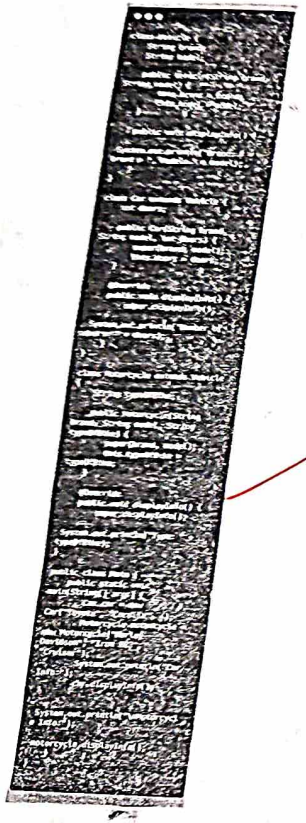
* Class :- Class is a set of objects which shares common characteristics / behavior and common properties / attributes.

* Super class / Parent class :- The class whose feature are inherited is known as a superclass.

* Sub class / child class :- The class that inherits the other class is known as a subclass.

* Reusability :- Inheritance supports the concept of 'reusability', i.e when we want to create a new class and there is already a class that includes some of the code that we want. Drive our new class from the existing class.

Teacher's Signature _____



Page No :
Date : / /

me of Practical

Program code :-

```
class vehicle {// properties of the base class
        string brand ;
        string model ;
// constructor for the base class
Public vehicle (string brand, string
                                model){
    this. brand = brand ;
    this. model = model ;
}. // method of the base class
  Public void displayInfo() {

System.out.Println("Brand :" + brand +",
    model :" + mode ) ;
  }
}
  // Doived class
  class car extends vehicle {
      int doors ;

Public car (string brand, string model,
    int doors) {
```

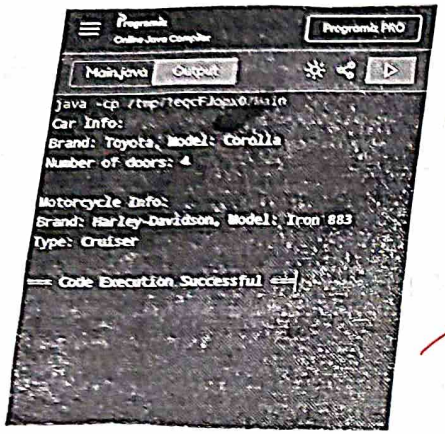
Teacher's Signature

Page No :
Date : / /

me of Practical

```
super (brand, model) {
    this.doors = doors;
}

@override
Public void displayInfo() {
super.displayInfo();

System.out.println("Number of doors:"
                    +doors);
}
}

class motorcycle extends vehicle.
{// Additional Property of the derived class
    string type of Bike;
// constructors for the derived class
Public motorcycle (string brand, string
model, string typeofBike) {

    super (brand, model);
    this.typeofBike = typeofBike;
} // method specific to the derived
                                    class.
```

Teacher's Signature _____



output :-

Car Info :

Brand : Toyota , model : corolla
Number of doors : 4

Motorcycle Info :
Brand : Harley - Devidson, Model : Iron 883
TYPE : Cruiser.

Page No :
Date :

Name of Practical

```
// overriding a method from the base
@override
Public void displayInfo(){
    super. displayInfo();

system.out.Println("TYPE:"+ typeofBike);
} // main class
  Public class main {
  Public static void
main (string[] args) {// create an object of
        Car car=new the vehicle class
car ("Toyota", "corolla", 4);
    motorcycle motorcycle = new
motorcycle ("Harley-Davidson", "Iron
  883", "cruiser");
system.out.Println("car Info :");
    Car.displayInfo();// calls the overridden
                          method.

System.out.Println("\n motorcycle Info:");
  // Display the Properties.
motorcycle.displayInfo();

    }

}
```

Teacher's Signature

conclusion :- Thus we have studied and implement the concept of inheritance in Java.

Page No :
Date : / /

Name of Practical

conclusion :- Thus we have studied and implement the concept of inheritance in Java.

Teacher's Signature

Practical No. 11

Aim:- Develop a Program to study and implement the concept of method overriding in Java.

Page No :
Date : / /

Practical No. 11

Name of Practical

Aim :- Develop a Program to study and implement the concept of method overriding in Java.

Theory :-

In Java, overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of it's super-classes or parent classes. when a method in a subclass has the same same, that same parameters or signature, and the same return type (or sub-type) as a method in it's super-class, then the method in the subclass is said to override the method in the super-class.

Teacher's Signature

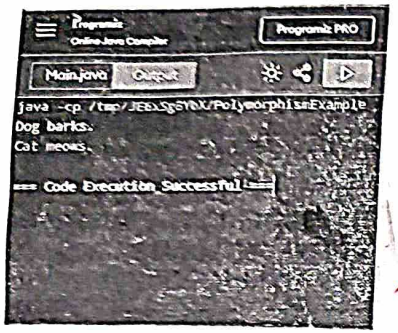
Page No :
Date : \ \

Name of Practical

Program Code :-

```
class Animal {
    public void sound() {
system.out.println("Animal 5 make
                    sounds");
    }
}
class Dog extends Animal {
    @override
    public void sound() {
system.out.println("Dog braks");
    }
}
class cat extends Animal {
    @override
    public void sound() {
```

Teacher's Signature



outPut :-

Dog barks

Cat meows

Page No :
Date : / /

Name of Practical

```
System.out.Println("Cat meows");
    }
}
public class main{
    public static void
main (string [] args) {
    Animal animal = new
Animal();
        Animal dog = new
Dog();
        Animal cat = new
Cat();
        animal.sound();
//calls Animals sound method
        dog.sound();
//calls Dog's overridding sound
                    method
    cat.sound();
//calls cat's overridden sound
                method
    }
}
```

Teacher's Signature

conclusion :- Thus we have studied and implement the concept of method overidding.

Page No :
Date : / /

Name of Practical

conclusion :- Thus we have studied and implement the concept of method overidding.

Teacher's Signature

Practical No. 12.

Aim :- Develop a program to study and implement the concept of polymorphism in Java.

me of Practical

Page No :
Date : / /

Practical No. 12.

Aim :- Develop a program to study and implement the concept of polymorphism in Java.

Theory :- The word 'polymorphism' means 'having many forms'. In simple words, we an define Java polymorphism as the ability of a message to be displayed in more than one form. In polymorphism is considerd one of the important features of object-oriented programming polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word "poly" means many and. "morphs" means form. So it means many-forms.

Teacher's Signature

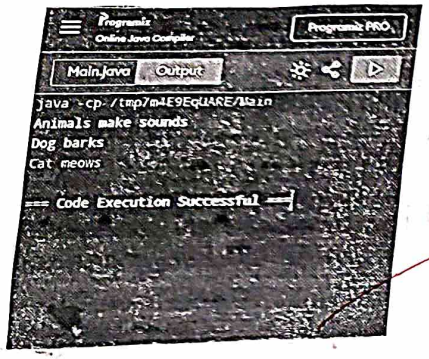
Page No :
Date : / /

Name of Practical

Program code :-

```
class Animal {
    void sound() {
System.out.println("This is a generic
        animal sound.");
    }
}
class Dog extends Animal
{
    @override
    void sound() {
System.out.println("Dog barks.");
    }
}
class cat extends Animal
{
    @override
    void sound() {
```

Teacher's Signature



output :-

Animals make sounds
Dog barks
cat meows

Name Practical

Page No :
Date : / /

```
System.out.Println("Cat meows.");
    }
}
public class
PolymorphismExample {
    public static void
main(String[] args) {
    Animal animal;
// Reference of Animal class
    animal = new

Dog();  // Dog object
    animal.sound();
// call's Dog's sound method
    animal = new
Cat();  // cat object

    animal.sound();
// call's cat's sound method.
    }
}
```

Teacher's Signature

conclusion :- Thus we have studied and implement the concept of polymorphism.

Name of

Page No :
Date : / /

Name of Practical

Conclusion :- Thus we have studied and implement the concept of polymorphism.

Teacher's Signature