• CODE:

```
import java.util.*;
class Employee {
        String name;
        int age;
        String phone_no;
        String address;
        double salary;

        double print_salary(){
            return salary;
        }
}
class Officer extends Employee {
        String specialization;
        public static void main(String args[]){
                Scanner s= new Scanner(System.in);
                Officer o=new Officer();
                System.out.println("Enter officer details:");
                System.out.println("Enter name:");
                o.name=s.nextline();
                System.out.println("Enter age:");
                o.age = s.nextInt();
                s.nextLine();
                System.out.println("Enter phone number:");
                o.phone_no =s.nextLine();
                System.out.println("Enter address:");
                o.address= s.nextLine();
                System.out.println("Enter salary:");
                o.salary = s.nextDouble();
                s.nextLine();
                System.out.println("Enter specialization:");
                o.specialization = s.nextLine();
                System.out.println("Officer details:");
```

EMPLOYEE - INHERITANCE

AIM: Write a Java program which creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'print-Salary()' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherits the 'Employee' class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an officer and a manager by making an object of both of these classes and print the same.

ALGORITHM:

Step 0: Start
Step 1: Create a class Employee; having name, age, phone_no, address, salary variables.
Step 2: Create a function print_Salary() to return value of salary.
Step 3: Create another class Officer which inherits from Employee Class.
Step 4: Create a variable specialization.
Step 5: Create an object o of Officer class.
Step 6: Read name, age, phone number, address, salary and specialization from user and store

```
        System.out.println("Name is: "+o.name);
        System.out.println("Age is: "+o.age);
        System.out.println("Phone number is: "+o.phone_no);
        System.out.println("Address is: "+o.address);
        System.out.println("Salary is: "+o.salary);
        System.out.println("Specialization is: "+o.specialization);
    }
}
class Manager extends Employee{
    String department;
    public static void main(String args[]){
        Scanner s=new Scanner(System.in);
        Manager m=new Manager();
        System.out.println("Enter manager details:");
        System.out.println("Enter name:");
        m.name=s.nextLine();
        System.out.println("Enter age:");
        m.age=s.nextInt();
        s.nextLine();
        System.out.println("Enter phone number:");
        m.phone_no=s.nextLine();
        System.out.println("Enter address:");
        m.address=s.nextLine()
        System.out.println("Enter salary:");
        m.salary=s.nextDouble();
        s.nextLine();
        System.out.println("Enter department:");
        m.department=s.nextLine();
        System.out.println("Manager details:");
        System.out.println("Name is:"+m.name);
```

in o.name, o.age, o.phone_no, o.address, o.salary and o.specialization respectively.

Step 7: Print name, age, phone number, address, salary, specialization.

Step 8: Create class Manager which inherits from class Employee.

Step 9: Create a variable department.

Step 10: Create an object m of Manager class.

Step 11: Read name, age, phone number, address, salary and department. Store in m.name, m.age, m.phone_no, m.address, m.salary and m.department respectively.

Step 12: Print name, age, phone number, address, salary, department.

Step 13: Stop

```
        System.out.println("Age is:"+m.age);
        System.out.println("Phone number is:"+m.phone_no);
        System.out.println("Address is:"+m.address);
        System.out.println("Salary is:"+m.salary);
        System.out.println("Department is:"+m.department);
    }
}
```

- OUTPUT:

```
Enter officer details:
Enter name:
Aabel
Enter age:
19
Enter phone number:
9718717709
Enter address:
Delhi
Enter salary:
342543
Enter specialization:
Front end
Officer details:
Name is:Aabel
Age is:19
Phone number is:9718717709
Address is:Delhi
Salary is:342543.0
Specialization is:Front end
```

• RESULT: The program has been completed successfully.

FACTORIAL USING RECURSION

- **AIM:** Write a Java program to find the factorial of a number using recursion.

- **ALGORITHM:**

Step 0: Start
Step 1: Create a class factorial_recursion.
Step 2: Create a function int fact (int n).
Step 3: Declare result. If n==0, return 1; else result= n*fact(n-1) and return value of result..
Step 4: In the main function, read the number whose factorial to be found and store in num.
Step 5: If num<0, print that factorial cannot be found.
Step 6: Else, create object f of class factorial_recursion.
Step 7: Initialize z=f.fact(num)
Step 8: Print value of z, which is the factorial of num.
Step 9: Stop

- **RESULT:** The program has been completed successfully.

40

- **CODE:**

```
import java.util.*;
class factorial_recursion {
        int fact (int n) {
                int result;
                if (n==0) {
                        return 1;
                }
                result = n* fact(n-1);
                return result;
        }
        public static void main(String args[]) {
                Scanner s = new Scanner (System.in);
                System.out.println ("Enter the number to find
                factorial:");
                int num = s.nextInt();
                if (num<0) {
                        System.out.println("Cannot find
                        factorial");
                } else {
                        factorial_recursion f= new factorial_recur
                                                        sion(
                        int z= f.fact (num);
                        System.out.println ("Factorial is:"+z);
                }
        }
}
```

dp 19/10

- **OUTPUT:**

```
Enter number to find its factorial:
5
Factorial is 120
```

• CODE:

```
import java.util.*;
class area_calculate{
    static double area;
    static final double pi=3.14;
    void Area(double radius){
        area= pi*radius*radius;
        System.out.println(area);
    }
    void Area(double length, double breadth){
        area = length* breadth;
        System.out.println(area);
    }
    void Area(double base, double height, double width){
        area = 0.5*base*height;
        System.out.println(area);
    }
}
class area{
    public static void main(String args[]){
        Scanner s=new Scanner(System.in);
        area_calculate ar=new area_calculate();
        System.out.print("Enter choice:\n1.Circle \n2.
        Rectangle \n3.Triangle \n");
        int choice = s.nextInt();
        switch(choice){
            case1:
                System.out.println("Enter radius:");
                double r= s.nextDouble();
                System.out.println("Area of circle is:");
                ar.Area(r);
                break;
```

Date: 31/10/23

Experiment No: 16

AREA OF SHAPES-METHOD OVERLOADING

- AIM: Write a Java program to calculate the area of different shapes namely circle, rectangle, and triangle using the concept of method ~~overloading~~ overloading.

- ALGORITHM:

Step 0: Start

Step 1: Create a class area_calculate.

Step 2: Create double area and finalize double pi=3.14

Step 3: Create a function void Area with a parameter radius.

Step 4: area= pi * radius * radius and print value of area.

Step 5: Create a function void Area with 2 parameters length and breadth.

Step 6: area= length * breadth and print area.

Step 7: Create another function void Area with 3 parameters base, height, width.

Step 8: area=0.5* base* height and print value of area.

Step 9: Create another class area.

Step 10: Create an object ar using area_calculate.

Step 11: Read the choice from user and store it in choice.

```
            case 2:
                System.out.println("Enter length:");
                double l=s.nextDouble();
                System.out.println("Enter breadth:");
                double b= s.nextDouble();
                System.out.println("Area of rectangle is:");
                ar.Area(l,b);
                break;
            case 3:
                System.out.println("Enter base:");
                double base=s.nextDouble();
                System.out.println("Enter height:");
                double h=s.nextDouble();
                System.out.println("Area of triangle is:");
                ar.Area(base,h,0);
                break;
            default:
                System.out.println("Invalid choice");
        }
    }
}
```

- **OUTPUT:**

```
Enter choice:
1. Circle
2. Rectangle
3. Triangle
1
Enter radius:
23
Area of circle is:
1661.06
```

Step 12: If choice=1, read radius and store in r.
Step 13: Call ar.Area(r) and break.
Step 14: If choice=2, read length l and breadth b.
Step 15: Call ar.Area(l,b) and break.
Step 16: If choice=3, read base base and height h of triangle
Step 17: Call ar.Area(base,h,0) and break.
Step 18: If any other value of choice, print invalid choice
Step 19: Stop

• RESULT: The program has been completed successfully.

•CODE:

```
import java.io.*;
class Employee {
        void display(){
                System.out.println("Name of class is Employee");
        }
        void calcSalary(){
                System.out.println("Salary of employee is 10000");
        }
}
class Engineer extends Employee{
        void display_employee() {
                super.display();
                super.calcSalary();
        }
        void display(){
                System.out.println("Name of class is Engineer");
        }
        void calcSalary(){
                System.out.println("Salary of engineer is 20000");
        }
}
class engineer_inheritance{
        public static void main(String[] args){
                Engineer e = new Engineer();
                e.display_employee();
                e.display();
                e.calcSalary();
        }
}
```

EMPLOYEE-ENGINEER INHERITANCE (USE OF SUPER KEYWORD)

• **AIM:** Write two Java classes Employee and Engineer. Engineer should inherit from Employee class. Employee class to have two methods display() and calcSalary(). Write a program to display the engineer salary and to display from Employee class using a single object instantiation (i.e, only one object creation is allowed).

- display() only prints the name of the class and does not return any value.
  Eg: "Name of class is Employee"
- calcSalary() in Employee displays "Salary of employee is 10000" and calcSalary() in Engineer displays "Salary of employee is 20000".

• **ALGORITHM:**

Step 0: Start
Step 1: Create a class Employee.
Step 2: Create 2 methods display() and calcSalary() and print the two statements given in question.
Step 3: Create a class Engineer which inherits from Employee class.
Step 4: Create a method display_employee() to

• OUTPUT:

```
Name of class is Employee
Salary of employee is 10000
Name of class is Engineer
Salary of engineer is 20000
```

print details of employee using super keyword.

Step 5: Create methods display() and calcSalary() and print engineer details.

Step 6: Create another class engineer_inheritance.

Step 7: Create object e of class Engineer.

Step 8: Call the functions e.display_employee, e.display and e.calcSalary to print employee and engineer details.

Step 9: Stop

- RESULT: The program has been completed successfully

•CODE:

```
import java.util.*;
abstract class Shape{
        abstract void numberOfSides();
}
class Rectangle extends Shape{
        void numberOfSides(){
            System.out.println("Rectangle has 4 sides");
        }
}
class Triangle extends Shape{
        void numberOfSides(){
            System.out.println("Triangle has 3 sides");
        }
}
class Hexagon extends Shape{
        void numberOfSides(){
            System.out.println("Hexagon has 6 sides");
        }
}
class Abstract{
        public static void main(String args[]){
            Scanner s= new Scanner(System.in);
            int choice=0;
            while(choice!=4){
                System.out.println("Enter choice: \n1.Rectangle
                \n2. Triangle \n3. Hexagon \n4.Exit \n");
                choice=s.nextInt();
                switch(choice){
                    case 1:
                        Rectangle obj1=new Rectangle();
                        obj1.numberOfSides();
                        break;
                    case 2:
```

Date: 09/11/2023 Experiment No: 18

ABSTRACT CLASS EXAMPLE

•AIM: Write a Java program to create an abstract class named Shape that contains an empty method named numberOfSides(). Provide three classes named Rectangle, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides() that shows the number of sides in the given geometrical structures.

•ALGORITHM:

Step 0: Start

Step 1: Create abstract class Shape with abstract method void numberOfSides().

Step 2: Create a class Rectangle which inherits from Shape. In function numberOfSides, print Rectangle has 4 sides.

Step 3: Create a class Triangle which inherits/ extends from Shape. In function numberOfSides, print Triangle has 3 sides.

Step 4: Create a class Hexagon which inherits from Shape. In function numberOfSides, print Hexagon has 6 sides.

Step 5: Create a class Abstract.

Step 6: Initialize choice=0 and run while loop

```
                Triangle obj2 = new Triangle();
                obj2.numberOfSides();
                break;
            - case 3:
                Hexagon obj3 = new Hexagon();
                obj3.numberOfSides();
                break;
            case 4:
                break;
            default:
                System.out.println("Enter valid choice");
        }
    }
}
```

- **OUTPUT:**

```
Enter choice:
1. Rectangle
2. Triangle
3. Hexagon
4. Exit
3
Hexagon has 6 sides
1
Rectangle has 4 sides
```

till choice != 4.

Step 7: Read choice from user and store it in choice.

Step 8: If choice = 1, create obj1 of class Rectangle and obj1.numberOfSides. Then break out of switch case.

Step 9: Else if choice = 2, create obj2 of class Triangle and obj2.numberOfSides. Break out of switch.

Step 10: Else if choice = 3, create obj3 of class Hexagon and obj3.numberOfSides. Break out of switch case.

Step 11: If choice = 4, break to terminate while loop.

Step 12: Else, print enter valid choice.

Step 13: Stop

• RESULT: The program has been completed successfully.

• CODE:

```
import java.io.*;
public class garbage_collection {
    public void finalize(){
        System.out.println("object is garbage
        collected");
    }
    public static void main(String args[]){
        garbage_collection s1=new garbage_collection();
        garbage_collection s2=new garbage_collection();
        s1=null;
        s2=null;
        System.gc();
    }
}
```

• OUTPUT:

```
object is garbage collected
object is garbage collected
```

Date: 09/11/2023 Experiment No: 19

GARBAGE COLLECTION

•AIM: Write a Java program to demonstrate the use of garbage collector.

•ALGORITHM:

Step 0: Start
Step 1: Create a class garbage_collection.
Step 2: Create a function void finalize().
Step 3: Inside the function, print object is garbage collected.
Step 4: In public static void main, create objects s1 and s2 of garbage_collection().
Step 5: s1 = null and s2 = null.
Step 6: System.gc(); for garbage collection
Step 7: Stop

•RESULT: The program has been completed successfully.

• **CODE:**

```
public class staticDemo{
    static int x=0;
    int y;
    static {
        System.out.println("Static block");
    }
    {
        System.out.println("Instance block");
    }
    static void staticMethod(){
        System.out.println("Static method");
    }
    void instanceMethod(){
        System.out.println("Instance method");
    }
    public static void main(String args[]){
        System.out.println("Static variable" +
                                    staticDemo.x);
        staticDemo.staticMethod();
        staticDemo obj1= new staticDemo();
        obj1.y=10;
        obj1.instanceMethod();
        staticDemo.x=5;
        System.out.println("Static variable through
        instance:" + obj1.x);
        obj1.staticMethod();
    }
}
```

Date: 31/10/2023

Experiment No: 63.20

STATIC VARIABLES, METHODS, BLOCKS

- **AIM:** Write a sample program to demonstrate the static variables, methods and blocks.

- **ALGORITHM:**

Step 0: Start
Step 1: Create a class staticDemo.
Step 2: Initialize a static variable x=0 and an instance variable y.
Step 3: Use static keyword and print Static block.
Step 4: Print Instance block.
Step 5: Create a static function void staticMethod() and print Static method.
Step 6: Create a function void instanceMethod() and print Instance method.
Step 7: In public static void main, print static variable and static method.
Step 8: Create object obj1 of staticDemo.
Step 9: Initialize obj1.y = 10 and call instance method.
Step 10: Initialize staticDemo.x = 5 and print value of static variable through instance (obj1.x).
Step 11: Call staticMethod().
Step 12: Stop

- OUTPUT:

Static block
Static variable 0
Static method
Instance block
Instance method
Static variable through instance: 5
Static  method

• RESULT: The program has been completed successfully and result obtained