Assignment 3

1.

```
//Write a program to demonstrate static variables and methods.

public class p1 {
   static int a=100;
   static void disp(){
    System.out.println(a);
   }
   public static void main(String args[]) {
     disp();
   }
}
```

2.

```
//Write a program for reuse class.

class add{

    void adder(int a,int b){
        System.out.println("sum is "+(a+b));
    }
}


public class p2 {
   int a=100;
   int b=300;

   public static void main(String args[]) {
     add A=new add();
     add B=new add();
     A.adder(10,5);
     A.adder(100,50);
   }
}
```

3.

```
//Write a program to give the example for method overriding concepts.

class add{

    void adder(int a,int b){
        System.out.println("Addition is "+(a+b));
    }
}


public class p3 extends add {

   void adder(int a,int b){
        System.out.println("substraction is "+(a-b));
    }

   public static void main(String args[]) {
     p3 A=new p3();
     A.adder(100,50);
   }
}
```

4.

```
//Write a program to give the example for ‘super’ keyword.

class Animal{
    void eat(){
        System.out.println("eating...");
    }
}
class Dog extends Animal{
    void eat(){
        System.out.println("eating bread...");
    }
    void bark(){
```

```
        System.out.println("barking...");
    }
    void work(){
    super.eat();
    bark();
    }
}
class p4{
    public static void main(String args[]){
        Dog d=new Dog();
        d.work();
    }
}
```

5.

```
//Write a program to create a class named shape. In this class we have three
sub classes
// circle, triangle and square each class has two member function named draw
()
//  and erase (). Create these using polymorphism concepts.

class Shape
{
    void draw()
    {
        System.out.println(“Shape draw()”);
    }
    void erase()

    {
        System.out.println (“ Shape erase()”);
    }
}
class Circle extends Shape
{
    void draw()
    {
```

```
        System.out.println (“Circle draw()”);
    }
    void erase()
    {
        System.out.println (“Circle erase()”);
    }
}
class Triangle extends Shape
{
    void draw()
    {
        System.out.println(“Triangle draw()”);
    }
    void erase()
    {
        System.out.println (“Triangle erase()”);
    }
}
class Square extends Shape
{
    void draw()
    {
        System.out.println (“Square draw()”);
    }
    void erase()
    {
        System.out.println (“Square erase()”);
    }
}
public class p5
{
    public static Shape randshape()
    {
        switch((int)(Math.random()*3))
        {
            case 0: return new Circle();
            case 1: return new Square();
```

```
            case 2: return new Triangle();
            default : System.out.println(“default”);
            return new Shape();
        }
    }
    public static void main (String arg[])
    {
        Shape s[] = new Shape[9];
        for(int i = 0;i< s.length; i++) s[i] = randshape();
        for(int i= 0;i < s.length; i++) s[i].draw();
    }
}

// 6.   Write a program to create interface A in this interface we h
// Define the interface A
interface A {
    void meth1();
    void meth2();
}

// Implement the interface in MyClass
class MyClass implements A {
    @Override
    public void meth1() {
        System.out.println("Executing meth1");
        // Add your code here for meth1
    }

    @Override
    public void meth2() {
        System.out.println("Executing meth2");
        // Add your code here for meth2
    }
}

// Main class to test the implementation
public class Main {
```

```java
    public static void main(String[] args) {
        // Create an instance of MyClass
        MyClass myObj = new MyClass();

        // Call the methods from the interface
        myObj.meth1();
        myObj.meth2();
    }
}

// 7.   Write a program to give example for multiple inheritance in Java.

// Define the first interface
interface A {
    void methodA();
}

// Define the second interface
interface B {
    void methodB();
}

// Implement both interfaces in a class
class MyClass implements A, B {
    @Override
    public void methodA() {
        System.out.println("Executing methodA");
        // Add your code here for methodA
    }

    @Override
    public void methodB() {
        System.out.println("Executing methodB");
        // Add your code here for methodB
    }
}
```

```java
// Main class to test the implementation
public class Main {
    public static void main(String[] args) {
        // Create an instance of MyClass
        MyClass myObj = new MyClass();

        // Call the methods from both interfaces
        myObj.methodA();
        myObj.methodB();
    }
}



// 8.   Write a program to create interface named test. In this interface the member function is square.
//      Implement this interface in arithmetic class.
//     Create one new class called ToTestInt in this class use the object of arithmetic class.

interface test{
    void square();
}
class arithmetic implements test{
    public void square(){
        System.out.println("Square method");
    }
}
public class toTestint{
    public static void main(String[] args) {

        arithmetic a=new arithmetic();
        a.square();
    }
}
```

```java
// 9.   Create an outer class with a function display, again create another
//      class inside the outer class named inner with
//      a function called display and call the two functions in the main
class

public class outer {
    void display()
    {
        System.out.println("Outer class display method");
         inner a=new inner();
        a.display();
    }
    class inner{
        void display()
        {
            System.out.println("Inner class display method");

        }
    }
    public static void main(String[] args) {
        outer o=new outer();
        o.display();


    }
}


// 10.  Write a program to give the example for ‘this’ operator.
//      And also use the ‘this’ keyword as return statement.

class Student {
    String name;

    Student(String name) {
        this.name = name;
    }
```

```
    String display() {
        return this.name;
    }
}

public class thiskey {
    public static void main(String args[]) {
        Student s1 = new Student("ankit");
        Student s2 = new Student("sumit");
        System.out.println(s1.display());
        System.out.println(s2.display());
    }
}
// 11.  Create a base class Building that stores the number of floors of a
building,
//      number of rooms and it’s total footage. Create a derived class House
//      that inherits Building and also stores the number of bedrooms and
bathrooms.
//      Demonstrate the working of the classes.

// Building class
class Building {
    private int floors;
    private int rooms;
    private double footage;

    public Building(int floors, int rooms, double footage) {
        this.floors = floors;
        this.rooms = rooms;
        this.footage = footage;
    }

    public int getFloors() {
        return floors;
    }
```

```
    public int getRooms() {
        return rooms;
    }

    public double getFootage() {
        return footage;
    }
}

// House class (derived from Building)
class House extends Building {
    private int bedrooms;
    private int bathrooms;

    public House(int floors, int rooms, double footage, int bedrooms, int
bathrooms) {
        super(floors, rooms, footage);
        this.bedrooms = bedrooms;
        this.bathrooms = bathrooms;
    }

    public int getBedrooms() {
        return bedrooms;
    }

    public int getBathrooms() {
        return bathrooms;
    }
}

// Demonstration of the classes
public class classderive {
    public static void main(String[] args) {
        // Create a building
        Building building = new Building(5, 20, 5000.0);
        System.out.println("Building - Floors: " + building.getFloors() + ",
Rooms: " + building.getRooms() + ", Footage: " + building.getFootage());
```

```
        // Create a house
        House house = new House(2, 6, 2000.0, 3, 2);
        System.out.println("House - Floors: " + house.getFloors() + ", Rooms:
" + house.getRooms() + ", Footage: " + house.getFootage() +
                           ", Bedrooms: " + house.getBedrooms() + ",
Bathrooms: " + house.getBathrooms());
    }
}


// 12.  In the earlier program, create a second derived class Office that
inherits
// Building and stores the number of telephones and tables.
// Now demonstrate the working of all three classes.

// Building class
class Building {
    private int floors;
    private int rooms;
    private double footage;

    public Building(int floors, int rooms, double footage) {
        this.floors = floors;
        this.rooms = rooms;
        this.footage = footage;
    }

    public int getFloors() {
        return floors;
    }

    public int getRooms() {
        return rooms;
    }

    public double getFootage() {
```

```
        return footage;
    }
}

// House class (derived from Building)
class House extends Building {
    private int bedrooms;
    private int bathrooms;

    public House(int floors, int rooms, double footage, int bedrooms, int
bathrooms) {
        super(floors, rooms, footage);
        this.bedrooms = bedrooms;
        this.bathrooms = bathrooms;
    }

    public int getBedrooms() {
        return bedrooms;
    }

    public int getBathrooms() {
        return bathrooms;
    }
}

// Office class (derived from Building)
class Office extends Building {
    private int telephones;
    private int tables;

    public Office(int floors, int rooms, double footage, int telephones, int
tables) {
        super(floors, rooms, footage);
        this.telephones = telephones;
        this.tables = tables;
    }
```

```
    public int getTelephones() {
        return telephones;
    }

    public int getTables() {
        return tables;
    }
}

// Demonstration of the classes
public class classderive2 {
    public static void main(String[] args) {
        // Create a building
        Building building = new Building(5, 20, 5000.0);
        System.out.println("Building - Floors: " + building.getFloors() + ",
Rooms: " + building.getRooms() + ", Footage: " + building.getFootage());

        // Create a house
        House house = new House(2, 6, 2000.0, 3, 2);
        System.out.println("House - Floors: " + house.getFloors() + ", Rooms:
" + house.getRooms() + ", Footage: " + house.getFootage() +
                           ", Bedrooms: " + house.getBedrooms() + ",
Bathrooms: " + house.getBathrooms());

        // Create an office
        Office office = new Office(10, 50, 10000.0, 50, 100);
        System.out.println("Office - Floors: " + office.getFloors() + ",
Rooms: " + office.getRooms() + ", Footage: " + office.getFootage() +
                           ", Telephones: " + office.getTelephones() + ",
Tables: " + office.getTables());
    }
}


// 13.  Write a Java program which creates a base class Num and contains an
integer
```

```java
// number along with a method shownum() which displays the number. Now create
a
// derived class HexNum which inherits Num and overrides shownum() which
displays
// the hexadecimal value of the number. Demonstrate the working of the
classes.

// Num base class
class Num {
    int number;

    public Num(int number) {
        this.number = number;
    }

    public void showNum() {
        System.out.println("Number: " + number);
    }
}

// HexNum derived class (inherits Num)
class HexNum extends Num {
    public HexNum(int number) {
        super(number);
    }

    @Override
    public void showNum() {
        System.out.println("Hexadecimal Value: " +
Integer.toHexString(super.number));
    }
}

// Demonstration of the classes
public class p13 {
    public static void main(String[] args) {
        // Create an instance of Num class
```

```
        Num num = new Num(42);
        num.showNum();

        // Create an instance of HexNum class
        HexNum hexNum = new HexNum(42);
        hexNum.showNum();
    }
}



// 14.  Create a base class called “vehicle” that stores number of wheels and
speed.
//  Create the following derived classes – “car” that inherits “vehicle” and
//  also stores number of passengers. “truck” that inherits “vehicle” and
also
//   stores the load limit. Write a main function to create objects of these
//   two derived classes and display all the information about “car” and
“truck”.
//    Also compare the speed of these two vehicles - car and truck and
display which one is faster.

// Vehicle base class
class Vehicle {
    int wheels;
    double speed;

    public Vehicle(int wheels, double speed) {
        this.wheels = wheels;
        this.speed = speed;
    }

    public void displayInfo() {
        System.out.println("Wheels: " + wheels + ", Speed: " + speed + "
km/h");
    }
}
```

```java
// Car derived class (inherits Vehicle)
class Car extends Vehicle {
    int passengers;

    public Car(int wheels, double speed, int passengers) {
        super(wheels, speed);
        this.passengers = passengers;
    }

    public void displayInfo() {
        super.displayInfo();
        System.out.println("Passengers: " + passengers);
    }
}

// Truck derived class (inherits Vehicle)
class Truck extends Vehicle {
    double loadLimit;

    public Truck(int wheels, double speed, double loadLimit) {
        super(wheels, speed);
        this.loadLimit = loadLimit;
    }

    public void displayInfo() {
        super.displayInfo();
        System.out.println("Load Limit: " + loadLimit + " tons");
    }
}

// Demonstration of the classes
public class p14 {
    public static void main(String[] args) {
        // Create a car object
        Car car = new Car(4, 120.0, 4);
        System.out.println("Car Information:");
```

```
        car.displayInfo();

        // Create a truck object
        Truck truck = new Truck(6, 80.0, 10.0);
        System.out.println("Truck Information:");
        truck.displayInfo();

        // Compare speeds
        if (car.speed > truck.speed) {
            System.out.println("The car is faster than the truck.");
        } else if (car.speed < truck.speed) {
            System.out.println("The truck is faster than the car.");
        } else {
            System.out.println("The car and truck have the same speed.");
        }
    }
}
```