MODULE: 2

- **<u>Practical-1</u>: Write a program to create a "distance" class with methods where distance is computed in terms of feet and inches, how to create objects of a class.**

```
import java.util.Scanner;
class Distance1{
   float feet;
   float inches;
   void distCalc(int length){
      feet = (length/(12f*2.54f));
      inches = (length*2.54f);
      System.out.println("Feet: " + feet + "\nInches: " + inches);
   }
}
public class OneDistance {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the length: ");
      int length = sc.nextInt();
      Distance1 d1 = new Distance1();
      d1.distCalc(length);
   }
}
```

Output:

```
Enter the length: 320
Feet: 10.498688
Inches: 812.8
```

- **<u>Practical-2</u>: Modify the "distance" class by creating constructor for assigning values (feet and inches) to the distance object. Create another object and assign second object as reference variable to another object reference variable. Further create a third object which is a clone of the first object.**

```
import java.util.Scanner;

class distance1 {
   double feet, inch;

   distance1() {
      double feet, inch;
      System.out.println("In default constructor");
   }

   void printData() {
      System.out.println("The distance is: " + feet + "ft.");
      System.out.println("The distance is: " + inch + "in.");
   }

   void setData(double f, double i) {
      feet = f;
      inch = i;
   }
}

class distance2 {
      public static void main(String[] args) {
         distance1 D1 = new distance1();
         distance1 D2 = new distance1();

         distance1 D3 = D1;
```

```
        //taking input of values...
        Scanner ob = new Scanner(System.in);

        System.out.print("Enter the distance in feet: ");
        double f = ob.nextDouble();

        System.out.print("Enter the distance in inch: ");
        double i = ob.nextDouble();

        //setting and printing data for D1
        D1.setData(f,i);
        D1.printData();

        D2.setData(f,i);
        D2.printData();

        //setting data of D3 as 50 feet..
        D3.feet = 50;

        System.out.println("The feet of D1: " + D1.feet);
    }
}
```

Output:

```
Constructor
Clone Constructor
Feet: 7
Inch: 5
Feet: 3
Inch: 5
Feet: 7
Inch: 5
```

- **<u>Practical-3</u>: Write a program in Java in which a subclass constructor invokes the constructor of the super class and instantiate the values.**

```
package src;
class Animal {
   // No-arg constructor
   Animal() {
      System.out.println("I am an animal");
   }
   // parameterized constructor
   Animal(String type) {
      System.out.println("Type: "+type);
   }
}
class Dog extends Animal {
   // default constructor
   Dog() {
      // calling parameterized constructor of the superclass
      super("Animal");
      System.out.println("I am a dog");
   }
}
public class ThreeSubSupConst {
   public static void main(String[] args) {
      Dog dog1 = new Dog();
   }
}
```

Output:

```
Type: Animal
I am a dog
```

- **Practical-4: Write a program in Java to develop overloaded constructor. Also develop the copy constructor to create a new object with the state of the existing object. (Class Point)**

```
class Point1{
   int x,y,z;
   Point1(){
      x=0;
      y=0;
      z=0;
      System.out.println("Invoked Default constructor");
      System.out.println(x + ", " + y + ", " + z);
   }

   Point1(int ix, int iy, int iz){
      x = ix;
      y = iy;
      z = iz;
      System.out.println("Invoked Parameterized constructor");
      System.out.println(x + ", " + y + ", " + z);
   }

   Point1(Point1 p){
      x = p.x;
      y = p.y;
      z = p.z;
      System.out.println("Invoked Copy Constructor");
      System.out.println(x + ", " + y + ", " + z);
   }
}

public class PDEU_Constructor {
   public static void main(String[] args) {
```

```
        Point1 p1 = new Point1();
        Point1 p2 = new Point1(5,10,15);
        Point1 p3 = new Point1(p2);
    }
}
```

Output:

```
Invoked Default constructor
0, 0, 0
Invoked Parameterized constru
5, 10, 15
Invoked Copy Constructor
```

- **<u>Practical-5</u>: Write a program to show the difference between public and private access specifiers. The program should also show that primitive data types are passed by value and objects are passed by reference and to learn use of final keyword.**

```
class Distance{
   public int feet, inch;
   private int meter, centimeter;
   final int kilometer = 1500;
   Distance(int f, int i){
      this.feet = f;
      this.inch = i;
      meter = 10, centimeter = 100;
   }
   void display(){
      System.out.println("Feet: " + feet);
      System.out.println("Inch: " + inch);
      System.out.println("Meter: " +meter);
      System.out.println("Centimeter: " + centimeter);
      kilometer = 2000; // kilometer can't be changed as it is final
   }
}
public class FiveAccessFinal {
   public static void main(String[] args) {
    Distance d1 = new Distance(23, 12);

      Distance d2 = new Distance(20, 30);
      d1.display(), d2.display();

      System.out.println(meter); // As it's Private, can't be called
      System.out.println(centimeter); // As it's Private, can't be called
   }
}
```

Output

```
Constructor
Constructor
Feet: 23
Inch: 12
Feet: 20
Inch: 30
```

- **Practical-6: Write a program to show the use of static functions and to pass variable length arguments in a function.**

```
class Demo {
   //non-static function
   void display() {
      System.out.println("A non-static function is called.");
   }
   //static function
   static void show() {
      System.out.println("The static function is called.");
   }
   //Varargs method
   static void displayVarargs(String... values){
      System.out.println("Displaying Varargs method ");
      for(String s:values){
         System.out.println(s);
      }
   }
}
public class SixStaticVarargs {
   public static void main(String args[]) {
      Demo obj = new Demo();
      obj.display();
      Demo.show();
      Demo.displayVarargs("This","is","Object","Oriented", "Programming");

   }
}
```

Output

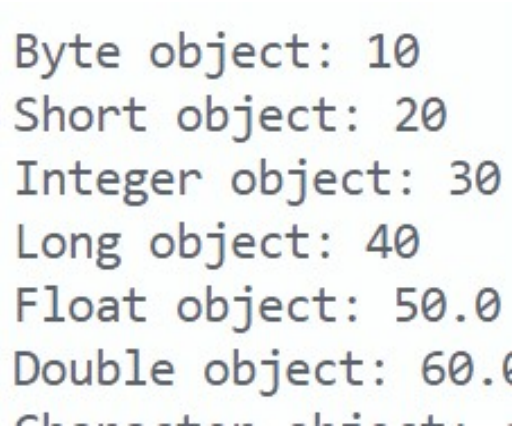
```
A non-static function is called.
The static function is called.

Displaying Varargs method
This
is
Object
Oriented
Programming
```

- **<u>Practical-7</u>: Write programs in Java to use Wrapper class of each primitive data types.**

```
public class SevenWrapperClass {

   public static void main(String[] args) {

      byte b = 10;

      short s=20;

      int i=30;

      long l=40;

      float f=50.0F;

      double d=60.0D;

      char c='a';

      boolean b2=true;


      Byte byteobj=b;

      Short shortobj=s;

      Integer intobj=i;

      Long longobj=l;

      Float floatobj=f;

      Double doubleobj=d;

      Character charobj=c;

      Boolean boolobj=b2;


      System.out.println("\nByte object: "+byteobj);

      System.out.println("Short object: "+shortobj);

      System.out.println("Integer object: "+intobj);

      System.out.println("Long object: "+longobj);

      System.out.println("Float object: "+floatobj);
```

```
        System.out.println("Double object: "+doubleobj);
        System.out.println("Character object: "+charobj);
        System.out.println("Boolean object: "+boolobj);
    }
}
```

Output:



- **<u>Practical-8</u>: Write a program that implements two constructors in the class. We call the other constructor using 'this' pointer, from the default constructor of the class**

```
class Point{
   int x,y,z;
   Point(){
      this.x=0;
      this.y=0;
      this.z=0;
      System.out.println("Invoked 1st constructor");
   Point(int x, int y, int z){
      this.x = x;
      this.y = y;
      this.z = z;
      System.out.println("Invoked 2nd constructor");
   }
}
class EightThisPointer {
   public static void main(String[] args) {
      Point p1 = new Point();
      Point p2 = new Point(5,10,15);
      System.out.println(p1.x+", "+p1.y+", "+p1.z);
      System.out.println(p2.x+", "+p2.y+", "+p2.z);
   }
}
```

Output:

```
Invoked 1st constructor
Invoked 2nd constructor
0, 0, 0
5, 10, 15
```

- **<u>Practical-9</u>: Write a program in Java to demonstrate single inheritance, multilevel inheritance and hierarchical inheritance.**

```
import java.util.Scanner;
public class singleInheritance {
    public static void main(String[] args) {
        programmer p = new programmer();
        p.setData();
        p.printData();
        System.out.println("Bonus of programmer is " + p.bonus);
    }
}
class employee {
    Scanner input = new Scanner(System.in);
    int salary;
    int empId;
    void setData(){
        System.out.print("Enter employee salary: ");
        salary = input.nextInt();
        System.out.print("Enter Employee Id: ");
        empId = input.nextInt();
    }
    void printData(){
        System.out.println("Salary of employee is " + salary);
        System.out.println("Employee ID of employee is " + empId);
    }
}
class programmer extends employee {
```

```
Enter employee salary: 65000
Enter Employee Id: 23
Salary of employee is 65000
Employee ID of employee is 23
Bonus of programmer is 100
```

```
    int bonus = 100;
}
    // Multilevel inheritance
    import java.util.Scanner;
    public class multilevelInheritance {
      public static void main(String[] arguments) {
        Cube cube = new Cube();
        cube.display();
        cube.area();
        cube.volume();
      }
    }
    class Shape {
      int areaofshape;
      void display() {
        System.out.println("Inside display");
      }
    }
    class Rectangle extends Shape {
      int l , d;
      Scanner input = new Scanner(System.in);
      void area() {
        System.out.print("Enter length of Rectangle: ");
        l = input.nextInt();
        System.out.print("Enter width of Rectangle: ");
        d = input.nextInt();
        areaofshape = l * d;
```

```
    System.out.println("Area of rectangle is " + areaofshape);
  }
}
class Cube extends Rectangle {
  int h;
  void volume() {
    System.out.print("Enter height of cube: ");
    h = input.nextInt();
    int vol = l * d * h;
    System.out.println("Volume of cube is " + vol);
  }
}
```

Output:

```
Inside display
Enter length of Rectangle: 65
Enter width of Rectangle: 32
Area of rectangle is 2080
Enter height of cube: 6
Volume of cube is 12480
```

// Hierarchical Inheritance

```
import java.util.Scanner;
public class hierachicalInheritance {
   public static void main(String[] args) {
      Twowheeler b = new Twowheeler();
      b.finalPrice();
      Fourwheeler c = new Fourwheeler();
      c.finalPricefour();
   }
}
class Vehicle {
   Scanner input = new Scanner(System.in);
   double baseprice = input.nextDouble();
}
class Twowheeler extends Vehicle {
```

```
    double increasePriceby = 0.2;
    void finalPrice() {
        baseprice = baseprice + (baseprice * increasePriceby);
        System.out.println("Final price of Two Wheeler is " + baseprice);
    }
}
class Fourwheeler extends Vehicle {
    double increasePriceby = 0.5;
    void finalPricefour() {
        baseprice = baseprice + (baseprice * increasePriceby);
        System.out.println("Final price of Four Wheeler is " + baseprice);
    }
}
```

Output:

```
30000
Final price of Two Wheeler is 36000.0
```

- **<u>Practical-10</u>: Java Program to demonstrate the real scenario (e.g., bank) of Java Method Overriding where three classes are overriding the method of a parent class.**

```
// SuperClass
class Bank {
  void loanInterest() {
     System.out.println("Interest Rate is 8%");
  }
}

// SubClass-1
class SBI extends Bank {
  void loanInterest() {
     System.out.println("SBI Bank Rate of Interest: 8.2%");
  }
}

// SubClass-2
class Axis extends Bank {
void loanInterest() {
     System.out.println("Axis Bank Rate of Interest: 9.8%");
  }
}

// SubClass-3
class ICICI extends Bank {
```

```
  void loanInterest() {
        System.out.println("ICICI Bank Rate of Interest: 9%");
    }
  }

 // Main Class
public class TenMethOverride
{
  public static void main(String[] args) {
     SBI s = new SBI();
     ICICI i = new ICICI();
     Axis a = new Axis();
     s.loanInterest();
     i.loanInterest();
     a.loanInterest();
   }
}
```

Output:

```
SBI Bank Rate of Interest:
ICICI Bank Rate of Interest
Axis Bank Rate of Interest:
```

- **<u>Practical-11</u>: Write a program that implements simple example of Runtime Polymorphism with multilevel inheritance. (e.g., Animal or Shape).**

```
class Animal{
   void sound(){System.out.println("Animals hae different sounds");}
}
class Cat extends Animal{
   void sound(){System.out.println("Meow");}
}
class Kitten extends Cat{
   void sound(){System.out.println("Meowww");}
}
public class ElevenMultiLevelInheritance {
   public static void main(String args[]) {
      // Creating Reference of Animal
      Animal a1;
      Cat c1 = new Cat();
      Kitten k1 = new Kitten();

      a1 = c1;
      a1.sound();
      a1 = k1;
      a1.sound();
   }
}
```

Output:

```
Meow
Meowww
```

- **<u>Practical-12</u>: Write a program to compute if one string is a rotation of another. For example, pit is rotation of tip as pit has same character as tip.**

```
import java.util.Scanner;


public class Twelve {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter word 1: ");
      String a = sc.next();
      System.out.print("Enter word 2: ");
      String b = sc.next();
      int count = 0;
      for(int i = 0; i < a.length(); i ++) {
         if(a.charAt(i) == (b.charAt(b.length() - i - 1))) {
            count ++;
         }
      }
      if(count == a.length()) {
         System.out.println(a + " is rotation of " + b);
      }
      else {
         System.out.println(a + " is not rotation of " + b);
      }
   }
}
```

Output

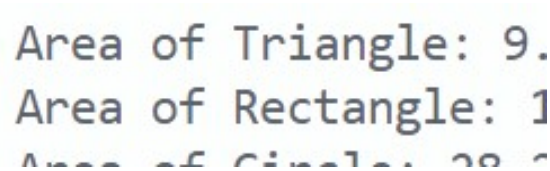
```
Enter word 1: JAVA
Enter word 2: AVAJ
JAVA is rotation of AVAJ
```

- **<u>Practical-13</u>: Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e. area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.**

```
abstract class Shape {
   abstract void area();
}

class Triangle{
   void area(float b, float h) {
      System.out.println("Area of Triangle: " + (0.5f*b*h));
   }
}

class Rectangle{
   void area(int l, int b) {
      System.out.println("Area of Rectangle: "+(l*b));
   }
}

class Circle{
   void area(float r) {
      System.out.println("Area of Circle: " + ((float)(3.14*r*r)));
   }
}

public class ThirteenAreaMethOverride {
```

```
    public static void main(String[] args) {
        Triangle t = new Triangle();
        Rectangle r = new Rectangle();
        Circle c = new Circle();
        t.area(3.3f,5.6f);
        r.area(3,6);
        c.area(3);
    }
}
```

Output:



- **<u>Practical-14</u>: Write a program in Java to demonstrate multiple inheritance.**

```
Interface BankA{
   float rateOfInterest();
}
interface BankB{
   float securities(long amount);
}
class SBI implements BankA, BankB{
   public float rateOfInterest() {
      return 9.75f;
   }
   public float securities(long amount){
      return (float)0.1f*amount;
   }
}
class ICICI implements BankA, BankB{
   public float rateOfInterest() {
      return 8f;
   }
   public float securities(long amount){
      return (float)0.2f*amount;
   }
}
class Axis implements BankA, BankB{
   public float rateOfInterest() {
      return 9f;
```

```
    }
    public float securities(long amount){
        return (float)0.3f*amount;
    }
}
public class multipleInheritanceInterfacing {
    public static void main(String[] args) {
        SBI s = new SBI();
        BankA I = new ICICI();
        BankB a = new Axis();

        System.out.println(s.rateOfInterest());
        System.out.println(s.securities(5000863));
        System.out.println(i.rateOfInterest());
        System.out.println(a.securities(8645167));
    }
}
```

Output:

```
9.75
500086.3
8.0
```

- **Practical-15:**

a) Write an application that illustrates method overriding in the same package and different packages.

```
class A {
   void show() {
      System.out.println("Inside Class-A");
   }
}
class B extends A {
   void show() {
      System.out.println("Inside Class-B");
   }
}
public class FifteenA {
   public static void main(String arg[]) {
      B obj=new B();
      obj.show(); //show() method of class-B will called.
   }
}
```

Output:

```
Inside Class-B
```

b) Also demonstrate accessibility rules in inside and outside packages.

package package1;

```
public class accessModifiers {
   public static void printData() {
      System.out.println("This is public access specifier");
   }
   protected static void print() {
      System.out.println("This is protected access specifier");
   }
   public static int a = 4;
   protected static int b = 3;
   int c = 2;
}
class A extends accessModifiers {
   void printdata() {
      System.out.println(c);
   }
}
```

```
package package2;
import package1.accessModifiers;
public class accessModifiers1 extends accessModifiers {
    public static void main(String[] args) {
        printData();
        print(); // Will give error
        System.out.println(a);
        System.out.println(b); // Will give error
    }
}
```

Output:

This is public access specifier