Aim: Write a Java Program to Display Employee Details using Constructor and this key word.

```
package JAVA;

public class Employee
{
        int empId;
        String empName;
       // parameterized constructor with two parameters

          Employee(int id, String name)
           {
             this.empId=id;
             this.empName=name;
           }

        void info()
       {
           System.out .println(" Id:"+empId+"Name:"+empName);
        }

       public static void main(String args[])
      {
         Employee obj1=new Employee(10245,"Madhu");
         Employee obj2=new Employee(92232,"Sudha");
         obj1.info();
         obj2.info();
      }
}
```

OUT PUT:

Aim: Write a Java program to illustrate Encapsulation.

```
package JAVA;

public class EncapTest
{
     private String name;
     private int age;
     public int getAge() {
         return age;
    }
    public String getName() {
         return name;
    }
   public void setAge(int newAge) {
        age=newAge;
  }
  public void setName(String newName) {
        name=newName;
  }

public static void main(String args[]) {
     EncapTest encap=new EncapTest();
     encap.setName("MADHU");
     encap.setAge(35);
     System.out.println("Name:"+encap.getName()+"\nAge:"+encap.getAge());
  }
}
```

OUT PUT:

Aim: Write a Java Program to Illustrate try, catch and finally Statement.

```
package JAVA;

public class Example {

        public static void main(String[] args)
        {

          try {
          System.out.println("First statement of try block");
          int num=45/0;
          System.out.println(num);
        }
        catch(ArithmeticException e) {
          System.out.println("ArithmeticException");
         }
        finally {
          System.out.println("finally block");
        }
          System.out.println("Out of try-catch-finally block");

}
}
```

OUT PUT:

Aim: Write a java program to illustrate Two Dimensional Array.

```
public class TwoDimensionalArrayExample {
    public static void main(String[] args) {
        // Declare and initialize a 2D array
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        // Access and print values from the 2D array
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

OUT PUT:

Aim: Write a java program to illustrate the concept of Overloading in Constructor.

```
package JAVA;

class Fanu {
	int price,speed;
	String colour;
	Fanu(int a, int b,String c){
		price=a;
		speed=b;
		colour=c;
	}
	Fanu(){
		price=100;
		speed=10;
		colour="white";
	}
	Fanu(int a,int b){
		price=a;
		speed=b;
	}
	public void display(){
		System.out.println("i am manufacturing a Fan");
		System.out.println(price);
		System.out.println(speed);
		System.out.println(colour);
	}
}
	public class Fan{
		public static void main(String[] args) {
		Fanu f1=new Fanu(2,10,"white");
		Fanu f2=new Fanu(5000,200);
		Fanu f3=new Fanu();
		f1.display();
		f2.display();
		f3.display();
	}
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Encapsulation Concept in java.

```
package JAVA;

public class amount {
private int amount;
private int acc_no;

public void deposit(int a) {
        if(a<0)
        {
                System.out.println("No negative deposit");

        }
        else {
                System.out.println("Amount deposited is;"+a);
                amount=amount+a;

        }
}
public class user{
        public static void main(String[] args) {
                amount a1=new amount();
                a1.deposit(100);
}
}
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Conditional Statement in Java:

i. if Statement
ii. if-else Statement
iii. Nested-if Statement

```
public class ConditionalStatements {
public static void main(String[] args) {
    int number = 10;

    // Example of if statement
    if (number > 0) {
        System.out.println("Number is positive.");
    }

    // Example of if-else statement
    if (number % 2 == 0) {
        System.out.println("Number is even.");
    } else {
        System.out.println("Number is odd.");
    }

    // Example of nested if statement
    if (number > 0) {
        if (number % 2 == 0) {
            System.out.println("Number is positive and even.");
        } else {
            System.out.println("Number is positive and odd.");
        }
    } else {
        System.out.println("Number is negative or zero.");
    }
  }
}
```

OUT PUT:

Aim: Write a Java program to illustrate the Do-While loop Statement.

```
package JAVA;
import java.util.Scanner;
public class Dowhile
{

	public static void main(String args[]) {
		int n,a,m=0,sum=0;
		Scanner s=new Scanner(System.in);

		System.out.print("enter any number:");
		n=s.nextInt();
		do
		{
			a=n%10;
			m=m*10+a;
			sum=sum+a;
			n=n/10;
		}
		while(n>0);
		System.out.println("reverse:"+m);
		System.out.println("sum of digits:"+sum);
	}
}
```

OUT PUT:

Aim: Write a Java Program to illustrate One Dimensional Array.

```
package JAVA;

public class OneDimArray {

	public static void main(String[] args) {
		int a[]=new int[5];//declaration and instantiation
		a[0]=10;//initialization
		a[1]=20;
		a[2]=30;
		a[3]=40;
		a[4]=50;
		//traversing array
		System.out.println("Elements of array are");
		for(int i=0;i<a.length;i++)
			System.out.println(a[i]);

	}

}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Interface Concept.

```
// Define an interface
interface Drawable {
    void draw();
}

// Implement the interface in a class
class Circle implements Drawable {
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

// Implement the interface in another class
class Rectangle implements Drawable {
    @Override
    public void draw() {
        System.out.println("Drawing a rectangle");
    }
}

public class InterfaceExample {
    public static void main(String[] args) {
        // Create objects of Circle and Rectangle classes
        Circle circle = new Circle();
        Rectangle rectangle = new Rectangle();

        // Call the draw() method on the objects
        circle.draw();
        rectangle.draw();
    }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Overriding Concept in Inheritence.

```
// Parent class
class Animal {
    public void makeSound() {
        System.out.println("Animal is making a sound");
    }
}

// Child class inheriting from the parent class
class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Meow");
    }
}

public class OverridingExample {
    public static void main(String[] args) {
        Animal animal = new Animal();
        Cat cat = new Cat();

        animal.makeSound(); // Call method from Animal class
        cat.makeSound();    // Call overridden method from Cat class
    }
}
```

OUT PUT:

Aim : Write a Java program to Display the Student Information using constructors.

```
package JAVA;

class Student{
	float height;
	int weight,age,mobile_no;
	String name,city,reg_no,gender;

	Mohsin(float height,int age,int mobile_no,int weight,String name,String reg_no,String gender,String
city){
		this.weight=weight;
		this.height=height;
		this.age=age;
		this.mobile_no=mobile_no;
		this.name=name;
		this.city=city;
		this.reg_no=reg_no;
		this.gender=gender;
	}
	public void display(){
		System.out.println("Mohsin Details are:");
		System.out.println("Name:"+name);
		System.out.println("Age:"+age);
		System.out.println("Reg_no:"+reg_no);
		System.out.println("Gender:"+gender);
		System.out.println("Mobile number:"+mobile_no);
		System.out.println("City:"+city);
		System.out.println("Weight:"+weight);
		System.out.println("Height:"+height);
	}
}
public class Simple {
	public static void main(String[] args) {
	Student m1=new Student(6,18,79089,54,"Mohsin","146CS21023","MALE","Gurgunta");
	Student m2=new  Student(6,18,792545,54,"Darshan","146CS21010","MALE","Sindhanur");
	m1.display();
	m2.display();
}
}
```

OUT PUT:

Aim : write a Java program to illustrate the Switch Operation.

```
package JAVA;
import java.util.Scanner;

public class SwitchDemo {
	public static void main(String[] args) {
		int day;
		Scanner input=new Scanner(System.in);
		System.out.println("enter a day:/n");
		day=input.nextInt();
		switch(day)
		{
		case 1:
			System.out.println("monday");
			break;
		case 2:
			System.out.println("tuesday");
			break;
		case 3:
			System.out.println("wednesday");
			break;
		case 4:
			System.out.println("thursday");
			break;
		case 5:
			System.out.println("friday");
			break;
		case 6:
			System.out.println("saturday");
			break;
		case 7:
			System.out.println("sunday");
			break;
			default:
				System.out.println("wrong option");
				break;
		}
	}
}
```

OUT PUT:

Aim : Write a Java program to illustrate the Break Statement.

```
public class BreakExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        // Iterate over the numbers array
        for (int num : numbers) {
            // Check if the number is equal to 3
            if (num == 3) {
                // If the number is 3, break out of the loop
                break;
            }

            // Print the number
            System.out.println(num);
        }
    }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Continue Statement.

```
public class ContinueExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        // Iterate over the numbers array
        for (int num : numbers) {
            // Check if the number is divisible by 2
            if (num % 2 == 0) {
                // If the number is divisible by 2, skip the current iteration
                continue;
            }

            // Print the number
            System.out.println(num);
        }
    }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the finalize() function Concept.

```
public class FinalizeExample {
    public static void main(String[] args) {
        Book book1 = new Book("Java Programming");
        Book book2 = new Book("Python Programming");

        // Set book2 reference to null
        book2 = null;

        // Request garbage collection
        System.gc();
    }
}

class Book {
    private String title;

    public Book(String title) {
        this.title = title;
    }

    // Override the finalize() method
    @Override
    protected void finalize() throws Throwable {
        System.out.println("Finalizing " + title);
    }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Different Datatypes.

```
public class DataTypesExample {
    public static void main(String[] args) {
        // Integer types
        byte myByte = 10;
        short myShort = 20;
        int myInt = 30;
        long myLong = 40L;

        // Floating-point types
        float myFloat = 3.14f;
        double myDouble = 2.71828;

        // Character type
        char myChar = 'A';

        // Boolean type
        boolean myBoolean = true;

        // String type
        String myString = "Hello, world!";

        // Output values
        System.out.println("Byte: " + myByte);
        System.out.println("Short: " + myShort);
        System.out.println("Int: " + myInt);
        System.out.println("Long: " + myLong);
        System.out.println("Float: " + myFloat);
        System.out.println("Double: " + myDouble);
        System.out.println("Char: " + myChar);
        System.out.println("Boolean: " + myBoolean);
        System.out.println("String: " + myString);
    }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Class and Object in a program.

```
public class ClassAndObjectExample {
    public static void main(String[] args) {
        // Create objects of the Person class
        Person person1 = new Person();
        Person person2 = new Person();

        // Set values for the first person
        person1.name = "Darshan";
        person1.age = 18;

        // Set values for the second person
        person2.name = "Mohsin";
        person2.age = 18;

        // Call methods on the objects
        person1.sayHello();
        person2.sayHello();
    }
}

class Person {
    // Instance variables
    String name;
    int age;

    // Method to say hello
    void sayHello() {
        System.out.println("Hello, my name is " + name + " and I am " + age + " years old.");
    }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Different Variable Types: Local, Instance, Static.

```
public class VariableTypesExample {
    // Instance variable
    int instanceVariable;

    // Static variable
    static int staticVariable;

    public static void main(String[] args) {
        // Local variable
        int localVariable = 10;

        VariableTypesExample example = new VariableTypesExample();
        example.instanceVariable = 20;
        staticVariable = 30;

        System.out.println("Local Variable: " + localVariable);
        System.out.println("Instance Variable: " + example.instanceVariable);
        System.out.println("Static Variable: " + staticVariable);
    }
}
```

OUT PUT:

Aim : Write a Java program to illustrate the Different Types of Constructor.

```
class Student
{
        String name;
        int regno;
        Student() {                     //Constructor
                name="Raju";
                regno=1234;
         }
        Student(String n, int r) {      // parameterized constructor
                name=n;
                regno=r;
        }
         Student(Student s) {           // copy constructor
                name=s.name;
                regno=s.regno;
        }
        void display(){
                System.out.println(name + "\t" +regno);
        }
}

public class StudentInfo {
        public static void main(String[] args) {
                Student s1= new Student();
                s1.display();
                Student s2= new Student(1,”Rahul”);
                s2.display();
                Student s3= new Student(s2);
                s3.display();
        }
}
```

OUT PUT:

Aim: Code, execute and debug programs that uses a. static binding b. dynamic binding.

```
public class BindingExample {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.makeSound(); // Static binding

        Animal dog = new Dog();
        dog.makeSound(); // Dynamic binding

        Animal cat = new Cat();
        cat.makeSound(); // Dynamic binding
    }
}

class Animal {
    public static void makeSound() {
        System.out.println("Animal is making a sound");
    }
}

class Dog extends Animal {
    public static void makeSound() {
        System.out.println("Dog is barking");
    }
}

class Cat extends Animal {
    public void makeSound() {
        System.out.println("Cat is meowing");
    }
}
```

OUT PUT:

Aim: Code, execute and debug programs that Uses Abstract class to achieve Abstraction.

```
abstract class Shape {
    protected String color;

    public Shape(String color) {
        this.color = color;
    }

    public abstract double getArea();
    public abstract double getPerimeter();

     public void display() {
        System.out.println("Color: " + color);
        System.out.println("Area: " + getArea());
        System.out.println("Perimeter: " + getPerimeter());
    }
}

class Rectangle extends Shape {
    private double length;
    private double width;

    public Rectangle(String color, double length, double width) {
        super(color);
        this.length = length;
        this.width = width;
    }

    @Override
    public double getArea() {
        return length * width;
    }

    @Override
    public double getPerimeter() {
        return 2 * (length + width);
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(String color, double radius) {
        super(color);
        this.radius = radius;
    }
```

```
    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }



@Override
    public double getPerimeter() {
        return 2 * Math.PI * radius;
    }
}

public class AbstractionExample {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle("Red", 5, 4);
        rectangle.display();

        System.out.println();

        Shape circle = new Circle("Blue", 3);
        circle.display();
    }
}
```

OUT PUT:

Aim: Code, execute and debug programs that uses interface to achieve abstraction.

```
interface Vehicle {
    void start();
    void stop();
}
class Car implements Vehicle {
    @Override
    public void start() {
        System.out.println("Car started");
    }
    @Override
    public void stop() {
        System.out.println("Car stopped");
    }
}

class Bike implements Vehicle {
    @Override
    public void start() {
        System.out.println("Bike started");
    }
    @Override
    public void stop() {
        System.out.println("Bike stopped");
    }
}

public class AbstractionWithInterfaceExample {
    public static void main(String[] args) {
        Vehicle car = new Car();
        car.start();
        car.stop();
        System.out.println();
        Vehicle bike = new Bike();
        bike.start();
        bike.stop();
    }
}
```

OUT PUT:

Aim : Code, execute and debug programs that uses inheritance concept.

```
class Vehicle {
    private String brand;
    private String color;

    public Vehicle(String brand, String color) {
        this.brand = brand;
        this.color = color;
    }

    public void start() {
        System.out.println("Starting the " + brand + " vehicle");
    }

    public void stop() {
        System.out.println("Stopping the " + brand + " vehicle");
    }
}

class Car extends Vehicle {
    private int numberOfDoors;

    public Car(String brand, String color, int numberOfDoors) {
        super(brand, color);
        this.numberOfDoors = numberOfDoors;
    }

    public void accelerate() {
        System.out.println("Accelerating the car");
    }
}

public class InheritanceExample {
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Red", 4);
        car.start();
        car.accelerate();
        car.stop();
    }
}
```

OUT PUT:

Aim: Code, execute and debug programs that uses polymorphism.

```
class Animal {
    public void makeSound() {
        System.out.println("Animal is making a sound");
    }
}

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Dog is barking");
    }
}

class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Cat is meowing");
    }
}

public class PolymorphismExample {
    public static void main(String[] args) {
        Animal animal1 = new Animal();
        Animal animal2 = new Dog();
        Animal animal3 = new Cat();

        animal1.makeSound();
        animal2.makeSound();
        animal3.makeSound();
    }
}
```

OUT PUT:

Aim: Code, execute and debug program for expression evaluation.

```
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class ExpressionEvaluation {
    public static void main(String[] args) {
        ScriptEngineManager manager = new ScriptEngineManager();
        ScriptEngine engine = manager.getEngineByName("js");

        String expression = "3 + 4 * (2 - 1)";
        try {
            Object result = engine.eval(expression);
            System.out.println("Result: " + result);
        } catch (ScriptException e) {
            System.out.println("Error evaluating expression: " + e.getMessage());
        }
    }
}
```

OUT PUT:

Aim : Code, execute and debug programs to connect to database through JDBC and perform basic DB operations.

```
import java.sql.*;

Connection conn = null;
String url = "jdbc:mysql://localhost:3306/mydatabase";
String username = "your-username";
String password = "your-password";

try {
    conn = DriverManager.getConnection(url, username, password);
    System.out.println("Connected to the database");
} catch (SQLException e) {
    System.out.println("Failed to connect to the database: " + e.getMessage());
}

Statement stmt = null;
ResultSet rs = null;
try {
    stmt = conn.createStatement();
    String sql = "SELECT * FROM users";
    rs = stmt.executeQuery(sql);

    while (rs.next()) {
        int id = rs.getInt("id");
        String name = rs.getString("name");
        String email = rs.getString("email");

        System.out.println("ID: " + id + ", Name: " + name + ", Email: " + email);
    }
} catch (SQLException e) {
    System.out.println("Failed to execute query: " + e.getMessage());
} finally {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
            // Ignore
        }
    }

    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException e) {
            // Ignore
        }
```

```
    }
}
try {
    if (conn != null) {
        conn.close();
        System.out.println("Disconnected from the database");
    }
} catch (SQLException e) {
    System.out.println("Failed to close the database connection: " + e.getMessage());
}
```

OUT PUT:

Aim: Write a simple Java program that demonstrates the relational operator.

```
public class RelationalOperatorExample {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;

        // Equal to
        boolean isEqual = (a == b);
        System.out.println("Equal to: " + isEqual);

        // Not equal to
        boolean isNotEqual = (a != b);
        System.out.println("Not equal to: " + isNotEqual);

        // Greater than
        boolean isGreater = (a > b);
        System.out.println("Greater than: " + isGreater);

        // Less than
        boolean isLess = (a < b);
        System.out.println("Less than: " + isLess);

        // Greater than or equal to
        boolean isGreaterOrEqual = (a >= b);
        System.out.println("Greater than or equal to: " + isGreaterOrEqual);

        // Less than or equal to
        boolean isLessOrEqual = (a <= b);
        System.out.println("Less than or equal to: " + isLessOrEqual);
    }
}
```

OUT PUT:

Aim: Write a Java Program to perform Logical Operator.

```
public class LogicalOperatorExample {
    public static void main(String[] args) {
        boolean a = true;
        boolean b = false;

        // Logical AND (&&)
        boolean resultAnd = a && b;
        System.out.println("Logical AND: " + resultAnd);

        // Logical OR (||)
        boolean resultOr = a || b;
        System.out.println("Logical OR: " + resultOr);

        // Logical NOT (!)
        boolean resultNotA = !a;
        boolean resultNotB = !b;
        System.out.println("Logical NOT a: " + resultNotA);
        System.out.println("Logical NOT b: " + resultNotB);
    }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate increment and decrement operators.

```
public class IncrementDecrementOperatorExample {
   public static void main(String[] args) {
      int num = 5;

      // Increment Operator
      System.out.println("Original Value: " + num);
      num++; // Increment by 1
      System.out.println("After Increment: " + num);

      // Decrement Operator
      num--; // Decrement by 1
      System.out.println("After Decrement: " + num);

      // Prefix Increment
      System.out.println("Prefix Increment: " + (++num));

      // Postfix Increment
      System.out.println("Postfix Increment: " + (num++));
      System.out.println("After Postfix Increment: " + num);

      // Prefix Decrement
      System.out.println("Prefix Decrement: " + (--num));

      // Postfix Decrement
      System.out.println("Postfix Decrement: " + (num--));
      System.out.println("After Postfix Decrement: " + num);
   }
}
```

OUT PUT:

Aim: Write a Java Program to illustrate the Different Access Modifiers.

```
// Class with public access modifier
public class AccessModifierExample {

    // Field with private access modifier
    private int privateField = 10;

    // Method with default (package-private) access modifier
    void defaultMethod() {
        System.out.println("Default Method");
    }

    // Method with protected access modifier
    protected void protectedMethod() {
        System.out.println("Protected Method");
    }

    // Method with public access modifier
    public void publicMethod() {
        System.out.println("Public Method");
    }

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

        // Accessing private field (Compile-time error)
        // System.out.println(obj.privateField);

        // Accessing default method within the same package
        obj.defaultMethod();

        // Accessing protected method within the same package
        obj.protectedMethod();

        // Accessing public method
        obj.publicMethod();
    }
}
```

OUT PUT:

Aim: Write a Java Program to Illustrate the Non-Access Modifiers.

```
public class NonAccessModifierExample {

    // Static variable with the static modifier
    static int staticVariable = 10;

    // Final variable with the final modifier
    final int finalVariable = 20;

    // Static method with the static modifier
    static void staticMethod() {
        System.out.println("Static Method");
    }

    // Synchronized method with the synchronized modifier
    synchronized void synchronizedMethod() {
        System.out.println("Synchronized Method");
    }

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

        // Accessing static variable
        System.out.println("Static Variable: " + staticVariable);

        // Accessing final variable
        System.out.println("Final Variable: " + obj.finalVariable);

        // Calling static method
        staticMethod();

        // Calling synchronized method
        obj.synchronizedMethod();
    }
}
```

OUT PUT:

Aim: Write a Java Program to create a Package.

To create a package in Java, you need to follow a specific directory structure and naming conventions. Here's an example of how to create a package in Java:

1. Create a new directory on your computer where you want to store your package. For example, let's create a directory named "myPackage".

2. Inside the "myPackage" directory, create a new subdirectory with the name of your package. For example, let's create a subdirectory named "com/mypackage".

3. Create a Java source file inside the package subdirectory. For example, let's create a file named "MyClass.java" with the following code:

```
package com.mypackage;

public class MyClass {
    public void displayMessage() {
        System.out.println("Hello, World!");
    }
}
```

4. Save the "MyClass.java" file in the "com/mypackage" subdirectory.
5. Open a command prompt or terminal and navigate to the parent directory of the "myPackage" directory.
6. Compile the Java source file using the 'javac' command:

```
javac myPackage/com/mypackage/MyClass.java.
```

7. Once the compilation is successful, a compiled bytecode file named "MyClass.class" will be generated in the same package subdirectory.
8. You have successfully created a package in Java! Now, you can use the package by importing it in other Java classes. For example, let's create another Java class to use the "MyClass" from the package:

```
import com.mypackage.MyClass;

public class Main {
    public static void main(String[] args) {
        MyClass myObj = new MyClass();
        myObj.displayMessage();
    }
}
```

9.Compile and run the "Main.java" file using the following commands:

javac Main.java
java Main

10.you should see the output "Hello, World!" on the console, indicating that the package and class are successfully being used.

OUT PUT:

Aim: Write a Java Program to illustrate the Multi-Level Inheritence.

```
class Animal {
    public void eat() {
        System.out.println("Animal is eating.");
    }
}

class Dog extends Animal {
    public void bark() {
        System.out.println("Dog is barking.");
    }
}

class Labrador extends Dog {
    public void displayColor() {
        System.out.println("Labrador is of yellow color.");
    }
}

public class MultilevelInheritanceExample {
    public static void main(String[] args) {
        Labrador labrador = new Labrador();
        labrador.eat(); // Inherited from Animal class
        labrador.bark(); // Inherited from Dog class
        labrador.displayColor(); // Specific to Labrador class
    }
}
```

OUT PUT:

PART-A

PART-B