**Assisted Practice: 3.8 Class, Objects, and Pillars of OOPs**

* Write a program in Java to demonstrate the uses of classes, objects, and the object-oriented pillars in Java
* Use Eclipse (the popular text editor for Java programs)
* Push code to Git

This lab has seven sub-sections, namely:

* + 1. Creating a new project in Eclipse
    2. Writing a program in Java to demonstrate the uses of classes and objects
    3. Writing a program in Java to demonstrate the uses of polymorphism
    4. Writing a program in Java to demonstrate the uses of inheritance
    5. Writing a program in Java to demonstrate the uses of encapsulation
    6. Writing a program in Java to demonstrate the uses of abstraction
    7. Pushing the code to your GitHub repositories

**Step 3.8.1:** Creating a new project in Eclipse

* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* Type in any project name and click on “Finish.”
* Select your project and go to File -> New -> Class.
* Type in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

**Step 3.8.2:** Writing a program in Java to demonstrate the uses of classes and objects

class Student{

int id;

String name;

}

class TestStudent2{

public static void main(String args[]){

Student s1=new Student();

s1.id=2527519;

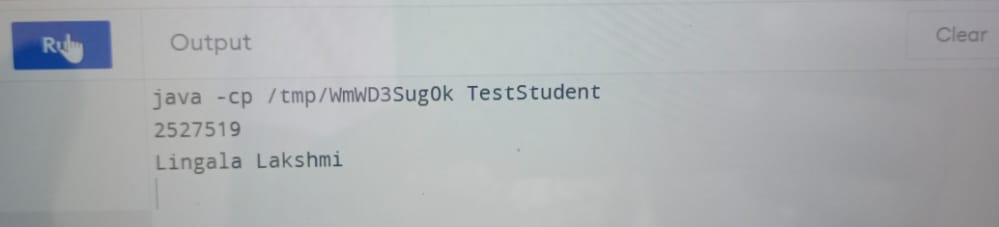
s1.name="Lingala Lakshmi";

System.out.println(s1.id+"\n"+s1.name);//printing members with a white space

}

}

**Output:**

****

**Step 3.8.3:** Writing a program in Java to demonstrate the uses of polymorphism

class Shape{

void draw(){System.out.println("drawing...");}

}

class Rectangle extends Shape{

void draw(){System.out.println("drawing rectangle...");}

}

class Circle extends Shape{

void draw(){System.out.println("drawing circle...");}

}

class Triangle extends Shape{

void draw(){System.out.println("drawing triangle...");}

}

class TestPolymorphism2{

public static void main(String args[]){

Shape s;

s=new Rectangle();

s.draw();

s=new Circle();

s.draw();

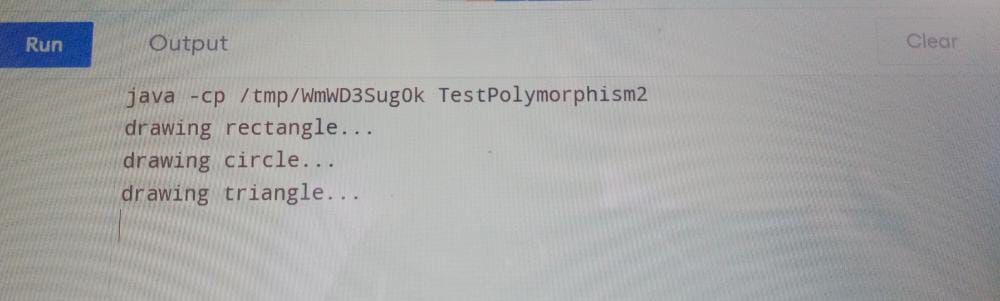
s=new Triangle();

s.draw();

}

}

**Output:**



**Step 3.8.4:** Writing a program in Java to demonstrate the uses of inheritance

class Animal{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

class TestInheritance{

public static void main(String args[]){

Dog d=new Dog();

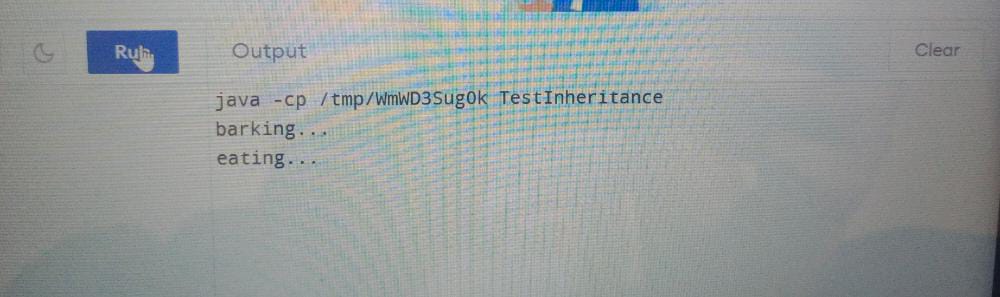
d.bark();

d.eat();

}

}

**Output:**

****

**Step 3.8.5:** Writing a program in Java to demonstrate the uses of encapsulation

class Area {

// fields to calculate area

int length;

int breadth;

// constructor to initialize values

Area(int length, int breadth) {

this.length = length;

this.breadth = breadth;

}

// method to calculate area

public void getArea() {

int area = length \* breadth;

System.out.println("Area: " + area);

}

}

class Main {

public static void main(String[] args) {

// create object of Area

// pass value of length and breadth

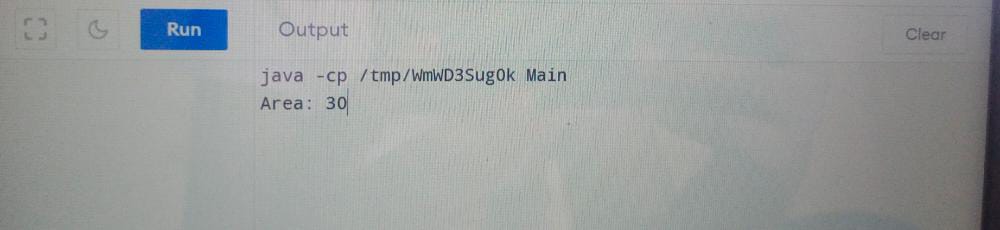
Area rectangle = new Area(5, 6);

rectangle.getArea();

}

}

**Output:**

****

**Step 3.8.6:** Writing a program in Java to demonstrate the uses of abstraction.

abstract class Car{

abstract void accelerate();

}

//concrete class

class Suzuki extends Car{

void accelerate(){

System.out.println("Suzuki::accelerate");

}

}

class Main{

public static void main(String args[]){

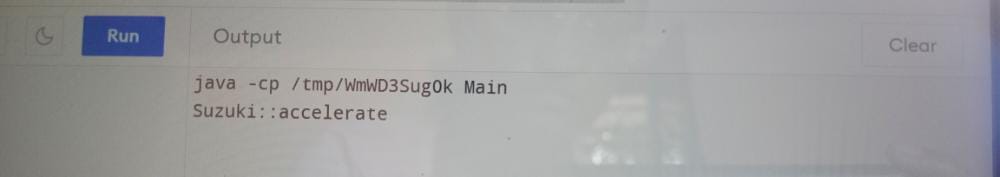
Car obj = new Suzuki(); //Car object =>contents of Suzuki

obj.accelerate(); //call the method

}

}

**Output:**

****