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

This section will guide you to:

* 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

public class Dog

{

String name;

String breed;

int age;

String color;

public Dog(String name, String breed, int age, String color)

{

this.name = name;

this.breed = breed;

this.age = age;

this.color = color;

}

public String getName()

{

return name;

}

public String getBreed()

{

return breed;

}

public int getAge()

{

return age;

}

public String getColor()

{

return color;

}

@Override

public String toString()

{

return("Hi my name is "+ this.getName()+ ".\nMy breed,age and color are " + this.getBreed()+", " + this.getAge()+ ", and"+ this.getColor() + ".");

}

public static void main(String[] args)

{

Dog scott = new Dog("Scott","papillon", 5, "black");

System.out.println(scott.toString());

}

}

**Output:**



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

class Sum

{

public int sum(int x, int y)

{

return (x + y);

}

public int sum(int x, int y, int z)

{

return (x + y + z);

}

public double sum(double x, double y)

{

return (x + y);

}

public static void main(String args[])

{

Sum s = new Sum();

System.out.println(s.sum(10, 20));

System.out.println(s.sum(10, 20, 30));

System.out.println(s.sum(10.5, 20.5));

}

}

**Output:**



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

class Bicycle

{

public int gear;

public int speed;

public Bicycle(int gear, int speed)

{

this.gear = gear;

this.speed = speed;

}

public void applyBrake(int decrement)

{

speed -= decrement;

}

public void speedUp(int increment)

{

speed += increment;

}

public String toString()

{

return("No of gears are " + gear + "\n" + "speed of bicycle is " + speed);

}

}

class MountainBike extends Bicycle

{

public int seatHeight;

public MountainBike(int gear,int speed,int startHeight)

{

super(gear, speed);

seatHeight = startHeight;

}

public void setHeight(int newValue)

{

seatHeight = newValue;

}

@Override

public String toString()

{

return (super.toString()+

"\nseat height is "+seatHeight);

}

}

public class Test

{

public static void main(String args[])

{

MountainBike mb = new MountainBike(3, 100, 25);

System.out.println(mb.toString());

}

}

**Output:**



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

public class Encapsulate

{

private String Name;

private int Roll;

private int Age;

public int getAge()

{

return Age;

}

public String getName()

{

return Name;

}

public int getRoll()

{

return Roll;

}

public void setAge( int newAge)

{

Age = newAge;

}

public void setName(String newName)

{

Name = newName;

}

public void setRoll( int newRoll)

{

Roll = newRoll;

}

}

public class TestEncapsulation

{

public static void main (String[] args)

{

Encapsulate obj = new Encapsulate();

obj.setName("Harsh");

obj.setAge(19);

obj.setRoll(51);

System.out.println("My name: " + obj.getName());

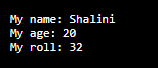
System.out.println("My age: " + obj.getAge());

System.out.println("My roll: " + obj.getRoll());

}

}

**Output:**



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

abstract class Shape

{

String color;

abstract double area();

public abstract String toString();

public Shape(String color)

{

System.out.println("Shape constructor called");

this.color = color;

}

public String getColor()

{

return color;

}

}

class Circle extends Shape

{

double radius;

public Circle(String color,double radius)

{

super(color);

System.out.println("Circle constructor called");

this.radius = radius;

}

@Override

double area()

{

return Math.PI \* Math.pow(radius, 2);

}

@Override

public String toString()

{

return "Circle color is " + super.color + "and area is : " + area();

}

}

class Rectangle extends Shape

{

double length;

double width;

public Rectangle(String color,double length,double width)

{

super(color);

System.out.println("Rectangle constructor called");

this.length = length;

this.width = width;

}

@Override

double area()

{

return length\*width;

}

@Override

public String toString()

{

return "Rectangle color is " + super.color +

"and area is : " + area();

}

}

public class Test

{

public static void main(String[] args)

{

Shape s1 = new Circle("Red", 2.2);

Shape s2 = new Rectangle("Yellow", 2, 4);

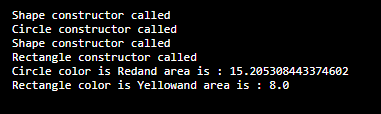
System.out.println(s1.toString());

System.out.println(s2.toString());

}

}

**Output:**



**Step 3.8.7:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**