|  |
| --- |
| **LAB FILE (CAF 612)**  *A report submitted in partial fulfillment of the requirement for the course*    **Advance Java Programming**  Part of the degree of  Master’s  **In**  **Computer Applications**    **SUBMITTED TO:**  Ms. Vishakha Arya  Assistant Professor  School of Computing    **SUBMITTED BY:**  Mohit Kumar  1000021153    **SCHOOL OF COMPUTING**    **DITUNIVERSITY,DEHRADUN**  (StatePrivateUniversitythroughStateLegislatureActNo. 10of2013ofUttarakhand and approvedbyUGC)  **MussoorieDiversion Road,Dehradun,Uttarakhand-248009,India.**  **2024-25**  1 |

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.** | **Name of Experiment** | **Date** | **Signature** |
| 1 | Write a java program for following:   1. Abstraction 2. Interface 3. Inheritance 4. Encapsulation 5. Polymorphism 6. Method overloading 7. Method overriding |  |  |
| 2 | Write a Java Program that creates a Hierarchy for Employees of a Company. The Base Class should be employee with sub classes Manager, Developer and Programmer.  Each sub class should have properties such as name, address, salary and job title. Implements methods for calculating bonuses, generating performance report & managing projects (Using Inheritance Only) |  |  |
| 3 | Write a Java Program to Create an abstract class Person with abstract method eat and exercise.  Create sub classes Athlete and Lazy Person that extends the Person class & implement the respective method to describe how each person eat and exercise. |  |  |
| 4 | Write a Java Program to create a class shape with methods getArea(), getParameter().  Create 3 sub classes Circle, Rectangle & Triangle. Override the getArea() and getParameter() in each sub classes to calculate & return the Area and Parameter of respective Classes |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| **7** |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |

**EXPERIMENT- 1**

Write a java program for following:

1. Abstraction
2. Interface
3. Inheritance
4. Encapsulation
5. Polymorphism
6. Method overloading
7. Method overriding

**Source Code:**

1. Abstraction

abstract class Animal {

        public abstract void animalSound();

        }

      class Cat extends Animal {

        public void animalSound() {

          System.out.println("The cat sounds: Meow");

        }

      }

     public class abstraction {

        public static void main(String[] args) {

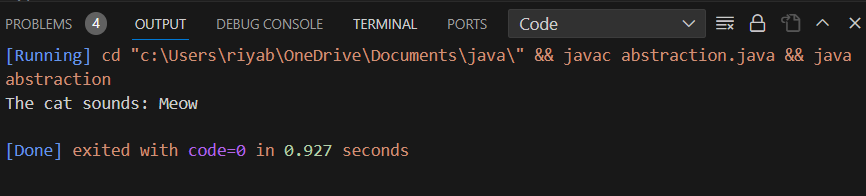
          Cat c = new Cat();

          c.animalSound();

        }

      }

**Output:**

****

1. Interface

class Animal{

        void eat()

        {

            System.out.println("eating...");}

        }

        class Dog extends Animal{

        void bark()

        {

            System.out.println("barking...");}

        }

        class BabyDog extends Dog{

        void weep()

        {

            System.out.println("weeping...");}

        }

       public class inheritance{

        public static void main(String args[]){

        BabyDog d=new BabyDog();

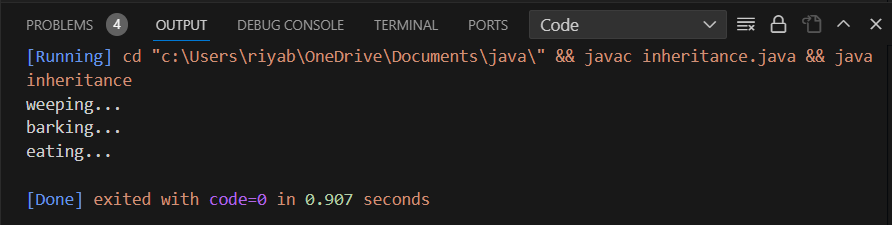
        d.weep();

        d.bark();

        d.eat();

        }}

**Output:**

****

1. Inheritance

interface Dog {

public void sound();

  }

interface Panda{

    public void eat();

  }

class Animal implements Dog, Panda {

    public void sound(){

      System.out.println("Barking");

    }

    public void eat() {

      System.out.println("Bamboo");

    }

}

  public class riya {

    public static void main(String[] args) {

      Animal dp = new Animal();

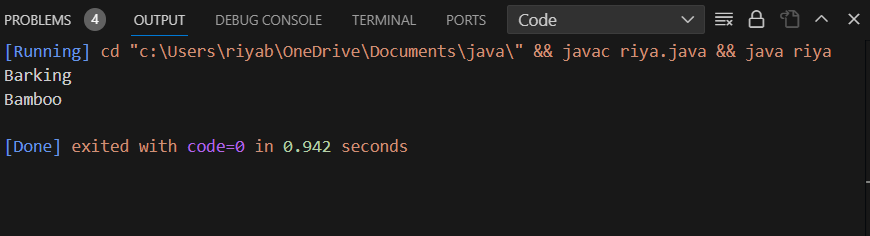
      dp.sound();

      dp.eat();

    }

  }

Output:



1. Encapsulation

class Area {

    int length;

    int breadth;

    Area(int length, int breadth) {

      this.length = length;

      this.breadth = breadth;

    }

    public void getArea() {

      int area = length \* breadth;

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

    }

  }

  public class encapsulation {

    public static void main(String[] args) {

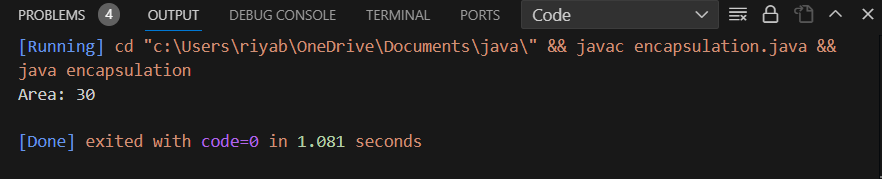
      Area rectangle = new Area(5, 6);

      rectangle.getArea();

    }

  }

Output:



1. 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...");}

    }

    public class polymorphism{

    public static void main(String args[]){

    Shape s;

    s=new Rectangle();

    s.draw();

    s=new Circle();

    s.draw();

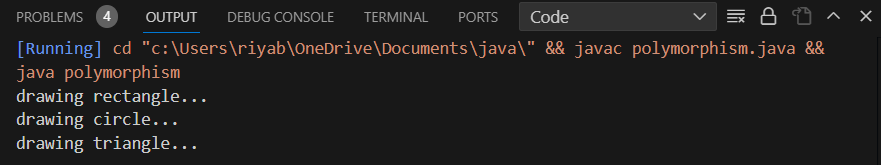
    s=new Triangle();

    s.draw();

    }

    }

Output:



1. Method overloading

public class methodoverloading {

    private static void display(int a){

        System.out.println("Arguments: " + a);

    }

    private static void display(int a, int b){

        System.out.println("Arguments: " + a + " and " + b);

    }

    public static void main(String[] args) {

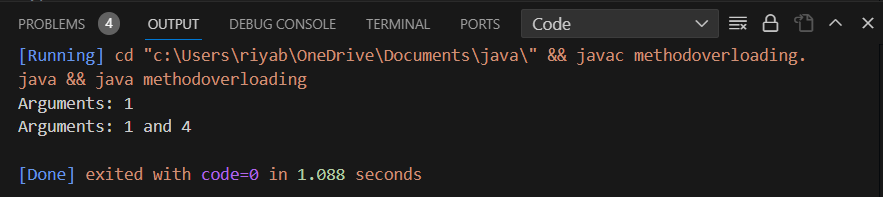
        display(1);

        display(1, 4);

    }

}

Output:’



1. Method overriding

class Animal {

    void sound() {

        System.out.println("Animal makes a sound");

    }

}

class Dog extends Animal{

    void sound() {

        System.out.println("Dog barks");

    }

}

class Cat extends Animal {

    void sound() {

        System.out.println("Cat meows");

    }

}

public class methodoverriding {

    public static void main(String[] args) {

        Animal animal1 = new Dog();

        Animal animal2 = new Cat();

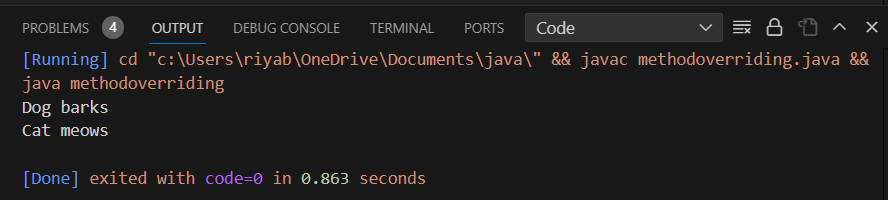
        animal1.sound();

        animal2.sound();

    }

}

Output:



# EXPERIMENT- 2

2. Write a Java Program that creates a Hierarchy for Employees of a Company. The Base Class should be employee with sub classes Manager, Developer and Programmer.Each sub class should have properties such as name, address, salary and job title. Implements methods for calculating bonuses, generating performance report & managing projects (Using Inheritance Only)

**Source Code:**

class Employee {

    String name;

    String address;

    double salary;

    String jobTitle;

    public Employee(String name, String address, double salary, String jobTitle) {

        this.name = name;

        this.address = address;

        this.salary = salary;

        this.jobTitle = jobTitle;

    }

    public void calculateBonus() {

        System.out.println("Bonus calculation for generic employee");

    }

    public void generatePerformanceReport() {

        System.out.println("Performance report for generic employee");

    }

}

class Manager extends Employee {

    int teamSize;

    public Manager(String name, String address, double salary, String jobTitle, int teamSize) {

        super(name, address, salary, jobTitle);

        this.teamSize = teamSize;

    }

    @Override

    public void calculateBonus() {

        System.out.println("Bonus calculation for Manager");

    }

    public void manageProjects() {

        System.out.println("Managing projects as a Manager");

    }

}

class Developer extends Employee {

    String programmingLanguage;

    public Developer(String name, String address, double salary, String jobTitle, String programmingLanguage) {

        super(name, address, salary, jobTitle);

        this.programmingLanguage = programmingLanguage;

    }

    @Override

    public void calculateBonus() {

        System.out.println("Bonus calculation for Developer");

    }

    public void writeCode() {

        System.out.println("Writing code as a Developer");

    }

}

class Programmer extends Developer {

    public Programmer(String name, String address, double salary, String programmingLanguage) {

        super(name, address, salary, "Programmer", programmingLanguage);

    }

    @Override

    public void calculateBonus() {

        System.out.println("Bonus calculation for Programmer");

    }

}

public class CompanyHierarchy {

    public static void main(String[] args) {

        Manager manager = new Manager("John Manager", "123 Main St", 80000, "Manager", 10);

        Developer developer = new Developer("Alice Developer", "456 Oak St", 60000, "Developer", "Java");

        Programmer programmer = new Programmer("Bob Programmer", "789 Pine St", 70000, "C++");

        manager.calculateBonus();

        manager.generatePerformanceReport();

        manager.manageProjects();

        developer.calculateBonus();

        developer.generatePerformanceReport();

        developer.writeCode();

        programmer.calculateBonus();

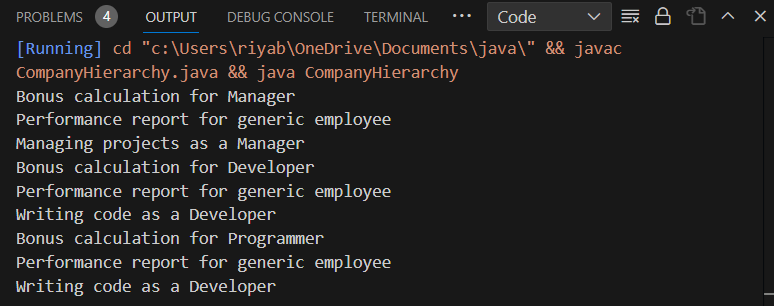
        programmer.generatePerformanceReport();

        programmer.writeCode();

    }

}

Output:



# EXPERIMENT- 3

Write a Java Program to Create an abstract class Person with abstract method eat and exercise.Create sub classes Athlete and Lazy Person that extends the Person class & implement the respective method to describe how each person eat and exercise**.**

**Source Code:**

abstract class Person {

    String name;

    public Person(String name) {

        this.name = name;

    }

    // Abstract methods

    abstract void eat();

    abstract void exercise();

}

class Athlete extends Person {

    public Athlete(String name) {

        super(name);

    }

    @Override

    void eat() {

        System.out.println(name + " (Athlete) eats a balanced and nutritious diet.");

    }

    @Override

    void exercise() {

        System.out.println(name + " (Athlete) follows a rigorous exercise routine.");

    }

}

class LazyPerson extends Person {

    public LazyPerson(String name) {

        super(name);

    }

    @Override

    void eat() {

        System.out.println(name + " (Lazy Person) enjoys snacks and fast food.");

    }

    @Override

    void exercise() {

        System.out.println(name + " (Lazy Person) prefers a sedentary lifestyle, avoiding exercise.");

    }

}

public class PersonHierarchy {

    public static void main(String[] args) {

        Athlete athlete = new Athlete("John");

        LazyPerson lazyPerson = new LazyPerson("Alice");

        athlete.eat();

        athlete.exercise();

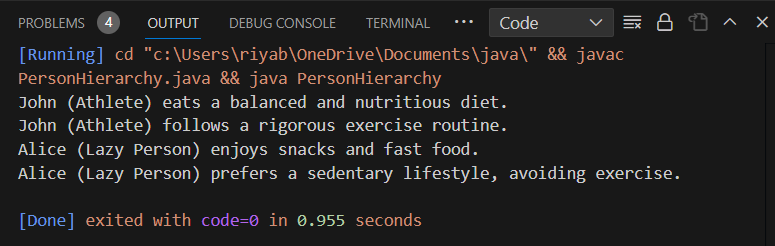
        lazyPerson.eat();

        lazyPerson.exercise();

    }

}

Output:



# EXPERIMENT- 4

Write a Java Program to create a class shape with methods getArea(), getParameter().Create 3 sub classes Circle, Rectangle & Triangle. Override the getArea() and getParameter() in each sub classes to calculate & return the Area and Parameter of respective Classes

**Source Code:**

abstract class Shape {

    abstract double getArea();

    abstract double getParameter();

}

class Circle extends Shape {

    double radius;

    public Circle(double radius) {

        this.radius = radius;}

    @Override

    double getArea() {

        return Math.PI \* radius \* radius;

    }

    @Override

    double getParameter() {

        return 2 \* Math.PI \* radius;}}

class Rectangle extends Shape {

    double length;

    double width;

    public Rectangle(double length, double width) {

        this.length = length;

        this.width = width;}

    @Override

    double getArea() {

        return length \* width;}

    @Override

    double getParameter() {

        return 2 \* (length + width);}}

class Triangle extends Shape {

    double side1;

    double side2;

    double side3;

    public Triangle(double side1, double side2, double side3) {

        this.side1 = side1;

        this.side2 = side2;

        this.side3 = side3;}

    @Override

    double getArea() {

        double s = getParameter() / 2.0;

        return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));}

    @Override

    double getParameter() {

        return side1 + side2 + side3;}}

public class ShapeDemo {

    public static void main(String[] args) {

        Circle circle = new Circle(5.0);

        Rectangle rectangle = new Rectangle(4.0, 6.0);

        Triangle triangle = new Triangle(3.0, 4.0, 5.0);

        System.out.println("Circle Area: " + circle.getArea());

        System.out.println("Circle Perimeter: " + circle.getParameter());

        System.out.println("\nRectangle Area: " + rectangle.getArea());

        System.out.println("Rectangle Perimeter: " + rectangle.getParameter());

        System.out.println("\nTriangle Area: " + triangle.getArea());

        System.out.println("Triangle Perimeter: " + triangle.getParameter());

    }

}

Output:

