1. Write a Java program to create a class called Employee with methods called work() and getSalary(). Create a subclass called HRManager that overrides the work() method and adds a new method called addEmployee().

**POGRAM**

//Q:1

// AUTHOR: Argha Digar

// TITLE: Employee and HRManager Program

// DESCRIPTION: This Java program defines an Employee class with work() and getSalary() methods, and a HRManager subclass that overrides the work() method and adds a new method called addEmployee().

class Employee {

    private String name;

    private double salary;

    public Employee(String name, double salary) {

        this.name = name;

        this.salary = salary;

    }

    public void work() {

        System.out.println(name + " is working as an employee.");

    }

    public double getSalary() {

        return salary;

    }

       public String getName() {

        return name;

    }

}

class HRManager extends Employee {

    public HRManager(String name, double salary) {

        super(name, salary);

    }

    public void work() {

        System.out.println(getName() + " is working as an HR Manager.");

    }

    public void addEmployee(Employee employee) {

        System.out.println(getName() + " is adding a new employee: " + employee.getName());

    }

}

public class Main {

    public static void main(String[] args) {

        Employee employee1 = new Employee("ARGHA", 50000.0);

        HRManager hrManager = new HRManager("JEET", 60000.0);

        employee1.work();

        System.out.println("Employee Salary: $" + employee1.getSalary());

        hrManager.work();

        System.out.println("HR Manager Salary: $" + hrManager.getSalary());

        Employee employee2 = new Employee("ARNAB", 55000.0);

        hrManager.addEmployee(employee2);

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUSETION1>javac Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUSETION1>java Main**

**ARGHA is working as an employee.**

**Employee Salary: $50000.0**

**JEET is working as an HR Manager.**

**HR Manager Salary: $60000.0**

**JEET is adding a new employee: ARNAB**

1. Write a Java program to create a class known as Person with methods called getFirstName() and getLastName(). Create a subclass called Employee that adds a new method named getEmployeeId() and overrides the getLastName() method to include the employee's job title.

**POGRAM :**

//Q:2

// AUTHOR: Argha Digar

// TITLE: Person and Employee Information Program

// DESCRIPTION: This Java program defines a Person class with getFirstName() and getLastName() methods, and a subclass called Employee that adds a new method getEmployeeId() and overrides the getLastName() method to include the employee's job title.

class Person {

    private String firstName;

    private String lastName;

    public Person(String firstName, String lastName) {

        this.firstName = firstName;

        this.lastName = lastName;

    }

    public String getFirstName() {

        return firstName;

    }

    public String getLastName() {

        return lastName;

    }

}

class Employee extends Person {

    private String employeeId;

    private String jobTitle;

    public Employee(String firstName, String lastName, String employeeId, String jobTitle) {

        super(firstName, lastName);

        this.employeeId = employeeId;

        this.jobTitle = jobTitle;

    }

    public String getEmployeeId() {

        return employeeId;

    }

    public String getLastName() {

        return super.getLastName() + ", " + jobTitle;

    }

}

public class Main {

    public static void main(String[] args) {

        Employee employee = new Employee("Argha", "Digar", "65432", "Software Engineer");

        System.out.println("First Name: " + employee.getFirstName());

        System.out.println("Last Name with Job Title: " + employee.getLastName());

        System.out.println("Employee ID: " + employee.getEmployeeId());

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-2>JAVAC Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-2>JAVA Main**

**First Name: Argha**

**Last Name with Job Title: Digar, Software Engineer**

**Employee ID: 65432**

1. Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed.

**POGRAM :**

//Q:3

// AUTHOR: Argha Digar

// TITLE: Vehicle Class Hierarchy Program

// DESCRIPTION: This Java program defines a vehicle class hierarchy with a base class Vehicle and three subclasses (Truck, Car, and Motorcycle). Each class includes properties for make, model, year, and fuel type, as well

class Vehicle {

    private String make;

    private String model;

    private int year;

    private String fuelType;

    public Vehicle(String make, String model, int year, String fuelType) {

        this.make = make;

        this.model = model;

        this.year = year;

        this.fuelType = fuelType;

    }

    public double calculateFuelEfficiency(double milesDriven, double gallonsConsumed) {

        if (gallonsConsumed <= 0) {

            return 0.0; // Avoid division by zero

        }

        return milesDriven / gallonsConsumed;

    }

    public double calculateDistance(double speed, double time) {

        return speed \* time;

    }

    public double calculateMaxSpeed(double enginePower, double weight) {

        return enginePower / weight \* 100;

    }

    public String getMake() {

        return make;

    }

    public String getModel() {

        return model;

    }

    public int getYear() {

        return year;

    }

    public String getFuelType() {

        return fuelType;

    }

}

class Truck extends Vehicle {

    public Truck(String make, String model, int year, String fuelType) {

        super(make, model, year, fuelType);

    }

}

class Car extends Vehicle {

    public Car(String make, String model, int year, String fuelType) {

        super(make, model, year, fuelType);

    }

}

class Motorcycle extends Vehicle {

    public Motorcycle(String make, String model, int year, String fuelType) {

        super(make, model, year, fuelType);

    }

}

public class Main {

    public static void main(String[] args) {

        Truck truck = new Truck("Tata", "240", 2023, "Gasoline");

        Car car = new Car("Toyota", "Camy", 2023, "Hybrid");

        Motorcycle motorcycle = new Motorcycle("Honda", "5ABC", 2023, "Gas");

        double fuelEfficiency = truck.calculateFuelEfficiency(250.0, 10.0);

        double distance = truck.calculateDistance(60.0, 4.0);

        double maxSpeed = truck.calculateMaxSpeed(300.0, 4000.0);

        System.out.println("Truck Make: " + truck.getMake());

        System.out.println("Truck Model: " + truck.getModel());

        System.out.println("Truck Year: " + truck.getYear());

        System.out.println("Truck Fuel Type: " + truck.getFuelType());

        System.out.println("Truck Fuel Efficiency: " + fuelEfficiency + " miles per L");

        System.out.println("Truck Distance Traveled: " + distance + " miles");

        System.out.println("Truck Maximum Speed: " + maxSpeed + " mph");

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-3>javac Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-3>java Main**

**Truck Make: Tata**

**Truck Model: 240**

**Truck Year: 2023**

**Truck Fuel Type: Gas**

**Truck Fuel Efficiency: 25.0 miles per L**

**Truck Distance Traveled: 240.0 miles**

**Truck Maximum Speed: 7.5 mph**

4. Write a Java program to create an abstract class Employee with abstract methods calculateSalary() and displayInfo(). Create subclasses Manager and Programmer that extend the Employee class and implement the respective methods to calculate salary and display information for each role.

**POGRAM :**

//Q:4

// AUTHOR: Argha Digar

// TITLE: Employee Management System

// DESCRIPTION: This Java program defines an abstract class Employee with abstract methods calculateSalary() and displayInfo(). It also includes two subclasses Manager and Programmer that extend the Employee class and implement the respective methods to calculate salary and display information for each role.

abstract class Employee {

    private String name;

    private int employeeId;

    public Employee(String name, int employeeId) {

        this.name = name;

        this.employeeId = employeeId;

    }

    // Abstract method to calculate salary

    public abstract double calculateSalary();

    // Abstract method to display employee information

    public abstract void displayInfo();

    public String getName() {

        return name;

    }

    public int getEmployeeId() {

        return employeeId;

    }

}

class Manager extends Employee {

    private double baseSalary;

    private double bonus;

    public Manager(String name, int employeeId, double baseSalary, double bonus) {

        super(name, employeeId);

        this.baseSalary = baseSalary;

        this.bonus = bonus;

    }

    public double calculateSalary() {

        return baseSalary + bonus;

    }

    public void displayInfo() {

        System.out.println("Manager Information:");

        System.out.println("Name: " + getName());

        System.out.println("Employee ID: " + getEmployeeId());

        System.out.println("Base Salary: $" + baseSalary);

        System.out.println("Bonus: $" + bonus);

        System.out.println("Total Salary: $" + calculateSalary());

    }

}

class Programmer extends Employee {

    private double hourlyRate;

    private int hoursWorked;

    public Programmer(String name, int employeeId, double hourlyRate, int hoursWorked) {

        super(name, employeeId);

        this.hourlyRate = hourlyRate;

        this.hoursWorked = hoursWorked;

    }

    public double calculateSalary() {

        return hourlyRate \* hoursWorked;

    }

    public void displayInfo() {

        System.out.println("Programmer Information:");

        System.out.println("Name: " + getName());

        System.out.println("Employee ID: " + getEmployeeId());

        System.out.println("Hourly Rate: $" + hourlyRate);

        System.out.println("Hours Worked: " + hoursWorked + " hours");

        System.out.println("Total Salary: $" + calculateSalary());

    }

}

public class Main {

    public static void main(String[] args) {

        Manager manager = new Manager("Arijit", 1001, 60000.0, 10000.0);

        Programmer programmer = new Programmer("Argha", 1002, 25.0, 160);

        manager.displayInfo();

        System.out.println();

        programmer.displayInfo();

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3>CD QUESTION-4**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-4>JAVAC Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-4>JAVA Main**

**Manager Information:**

**Name: Arijit**

**Employee ID: 1001**

**Base Salary: $60000.0**

**Bonus: $10000.0**

**Total Salary: $70000.0**

**Programmer Information:**

**Name: Argha**

**Employee ID: 1002**

**Hourly Rate: $25.0**

**Hours Worked: 160 hours**

**Total Salary: $4000.0**

5. Write a Java program to create an abstract class GeometricShape with abstract methods area() and perimeter(). Create subclasses Triangle and Square that extend the GeometricShape class and implement the respective methods to calculate the area and perimeter of each shape.

**POGRAM :**

// Q : 5

// AUTHOR: Argha Digar

// TITLE: Geometric Shapes Program

// DESCRIPTION: This Java program defines an abstract class GeometricShape with abstract methods area() and perimeter(). It also includes two subclasses, Triangle and Square, that extend the GeometricShape class and implement the respective methods to calculate the area and perimeter of each shape.

abstract class GeometricShape {

    // Abstract method to calculate the area

    public abstract double area();

    // Abstract method to calculate the perimeter

    public abstract double perimeter();

}

class Triangle extends GeometricShape {

    private double sideA;

    private double sideB;

    private double sideC;

    public Triangle(double sideA, double sideB, double sideC) {

        this.sideA = sideA;

        this.sideB = sideB;

        this.sideC = sideC;

    }

    public double area() {

        // Using Heron's formula to calculate the area form google

        double s = (sideA + sideB + sideC) / 2.0;

        return Math.sqrt(s \* (s - sideA) \* (s - sideB) \* (s - sideC));

    }

    public double perimeter() {

        return sideA + sideB + sideC;

    }

}

class Square extends GeometricShape {

    private double side;

    public Square(double side) {

        this.side = side;

    }

    public double area() {

        return side \* side;

    }

    public double perimeter() {

        return 4 \* side;

    }

}

public class Main {

    public static void main(String[] args) {

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

        Square square = new Square(4.0);

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

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

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

        System.out.println("Square Perimeter: " + square.perimeter());

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-5>javac Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-5>java Main**

**Triangle Area: 6.0**

**Triangle Perimeter: 12.0**

**Square Area: 16.0**

**Square Perimeter: 16.0**

6. Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports

**POGRAM :**

//Q : 6

// AUTHOR: Argha Digar

// TITLE: Sports Simulation Program

// DESCRIPTION: This Java program defines an interface Playable with a method play() that is implemented by three classes (Football, Volleyball, and Basketball). Each class overrides the play() method to simulate playing the respective sport.

interface Playable {

    void play();

}

class Football implements Playable {

    public void play() {

        System.out.println("Playing Football");

    }

}

class Volleyball implements Playable {

    public void play() {

        System.out.println("Playing Volleyball");

    }

}

class Basketball implements Playable {

    public void play() {

        System.out.println("Playing Basketball");

    }

}

public class Main {

    public static void main(String[] args) {

        Playable football = new Football();

        Playable volleyball = new Volleyball();

        Playable basketball = new Basketball();

        // Play the respective sports

        football.play();

        volleyball.play();

        basketball.play();

    }

}

**OUTPUT:**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-6>JAVAC Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-6>JAVA Main**

**Playing Football**

**Playing Volleyball**

**Playing Basketball**

7. Write a java program to implement multiple inheritance through interface.

**POGRAM :**

//Q : 7

// AUTHOR: Argha Digar

// TITLE: Multiple Inheritance Through Interfaces

// DESCRIPTION: This Java program demonstrates multiple inheritance through interfaces. It defines two interfaces (Interface1 and Interface2) with methods and implements both interfaces in a class called MyClass, which inherits from both interfaces and provides its own methods.

// Define first interface

interface Interface1 {

    void method1();

}

// Define the second interface

interface Interface2 {

    void method2();

}

// Implement both interfaces in a class

class MyClass implements Interface1, Interface2 {

    public void method1() {

        System.out.println("Method 1 from Interface1");

    }

    public void method2() {

        System.out.println("Method 2 from Interface2");

    }

    // Additional methods specific to MyClass

    void myMethod() {

        System.out.println("MyClass-specific method");

    }

}

public class Main {

    public static void main(String[] args) {

        MyClass obj = new MyClass();

        obj.method1(); // Call method from Interface1

        obj.method2(); // Call method from Interface2

        obj.myMethod(); // Call method specific to MyClass

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-7>JAVAC Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-7>JAVA Main**

**Method 1 from Interface1**

**Method 2 from Interface2**

**MyClass-specific method**

8. Write a Java program to create a class called Car with private instance variables company\_name, model\_name, year, and mileage. Provide public getter and setter methods to access and modify the company\_name, model\_name, and year variables. However, only provide a getter method for the mileage variable.

**POGRAM :**

// Q : 8

// AUTHOR: Argha Digar

// TITLE: Car Information Program

// DESCRIPTION: This Java program defines a class called Car with private instance variables for company\_name, model\_name, year, and mileage. It provides public getter and setter methods to access and modify the company\_name, model\_name, and year variables. However, only a getter method is provided for the mileage variable. The program demonstrates how to create a Car instance, access its properties, and modify some of its properties.

class Car {

    private String company\_name;

    private String model\_name;

    private int year;

    private double mileage;

    public String getCompany\_name() {

        return company\_name;

    }

    public void setCompany\_name(String company\_name) {

        this.company\_name = company\_name;

    }

    public String getModel\_name() {

        return model\_name;

    }

    public void setModel\_name(String model\_name) {

        this.model\_name = model\_name;

    }

    public int getYear() {

        return year;

    }

    public void setYear(int year) {

        this.year = year;

    }

    public double getMileage() {

        return mileage;

    }

    public void setMileage(double mileage) {

        this.mileage = mileage;

    }

    // Constructor

    public Car(String company\_name, String model\_name, int year, double mileage) {

        this.company\_name = company\_name;

        this.model\_name = model\_name;

        this.year = year;

        this.mileage = mileage;

    }

}

public class Main {

    public static void main(String[] args) {

        Car myCar = new Car("Thar", "ABC", 2022, 30.5);

        // Use getter methods to access the values

        System.out.println("Company: " + myCar.getCompany\_name());

        System.out.println("Model: " + myCar.getModel\_name());

        System.out.println("Year: " + myCar.getYear());

        System.out.println("Mileage: " + myCar.getMileage());

        // Use setter methods to modify values

        myCar.setYear(2023);

        myCar.setMileage(32.0);

        // Check the updated values

        System.out.println("Updated Year: " + myCar.getYear());

        System.out.println("Updated Mileage: " + myCar.getMileage());

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUSTION-8>javac Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUSTION-8>java Main**

**Company: Thar**

**Model: ABC**

**Year: 2022**

**Mileage: 30.5**

**Updated Year: 2023**

**Updated Mileage: 32.0**

9. Write a Java program to create a class called Student with private instance variables student\_id, student\_name, and grades. Provide public getter and setter methods to access and modify the student\_id and student\_name variables. However, provide a method called addGrade() that allows adding a grade to the grades variable while performing additional validation.

**POGRAM :**

// Q : 9

// AUTHOR: Argha Digar

// TITLE: Student Grade Management System

// DESCRIPTION: This Java program defines a class called Student with private instance variables student\_id, student\_name, and grades. It provides public getter and setter methods to access and modify the student\_id and student\_name variables. Additionally, it offers a method called addGrade() that allows adding a grade to the grades array while performing validation to ensure the grade is between 0 and 100. The program demonstrates how to create a Student instance, access its properties, and manage student grades using arrays.

class Student {

    private int student\_id;

    private String student\_name;

    public int[] grades;

    private int gradeCount;

    public Student(int student\_id, String student\_name, int maxGradeCount) {

        this.student\_id = student\_id;

        this.student\_name = student\_name;

        this.grades = new int[maxGradeCount];

        this.gradeCount = 0;

    }

    public int getStudent\_id() {

        return student\_id;

    }

    public void setStudent\_id(int student\_id) {

        this.student\_id = student\_id;

    }

    public String getStudent\_name() {

        return student\_name;

    }

    public void setStudent\_name(String student\_name) {

        this.student\_name = student\_name;

    }

    // Method to add a grade with validation

    public void addGrade(int grade) {

        if (grade >= 0 && grade <= 100 && gradeCount < grades.length) {

            grades[gradeCount] = grade;

            gradeCount++;

            System.out.println("Grade added: " + grade);

        } else if (gradeCount >= grades.length) {

            System.out.println("Cannot add more grades. Maximum grade count reached.");

        } else {

            System.out.println("Invalid grade. Grade must be between 0 and 100.");

        }

    }

  }

}

public class Main {

    public static void main(String[] args) {

        Student student = new Student(101, "Argha Digar", 5); // Maximum 5 grades

        // Use getter methods to access the values

        System.out.println("Student ID: " + student.getStudent\_id());

        System.out.println("Student Name: " + student.getStudent\_name());

        // Add grades

        student.addGrade(85);

        student.addGrade(92);

        student.addGrade(105); // Invalid grade

        student.addGrade(75);

        student.addGrade(98);

        student.addGrade(88); // Maximum grade count reached

        // Access grades

        System.out.print("All Grades: ");

        for (int i=0;i<5;i++){

        System.out.print(student.grades[i]+"  ");

        }

    }}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-9>javac Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-9>java Main**

**Student ID: 101**

**Student Name: Argha Digar**

**Grade added: 85**

**Grade added: 92**

**Invalid grade. Grade must be between 0 and 100.**

**Grade added: 75**

**Grade added: 98**

**Grade added: 88**

**All Grades: 85 92 75 98 88**

10. Write a Java program to create a base class Shape with methods draw() and calculateArea(). Create three subclasses: Circle, Square, and Triangle. Override the draw() method in each subclass to draw the respective shape, and override the calculateArea() method to calculate and return the area of each shape.

**POGRAM :**

// Q : 10

// AUTHOR: Argha Digar

// TITLE: Shape Drawing and Area Calculation

// DESCRIPTION: This Java program defines a base class Shape with methods draw() and calculateArea(). It also includes three subclasses: Circle, Square, and Triangle. The draw() method is overridden in each subclass to draw the respective shape, and the calculateArea() method is overridden to calculate and return the area of each shape. The program demonstrates how to create instances of these shapes and utilize their draw and area calculation functionalities.

class Shape {

    public void draw() {

        System.out.println("Drawing a shape");

    }

    public double calculateArea() {

        return 0.0;

    }

}

// Subclass Circle

class Circle extends Shape {

    private double radius;

    public Circle(double radius) {

        this.radius = radius;

    }

    public void draw() {

        System.out.println("Drawing a circle");

    }

    public double calculateArea() {

        return Math.PI \* radius \* radius;

    }

}

// Subclass Square

class Square extends Shape {

    private double side;

    public Square(double side) {

        this.side = side;

    }

    public void draw() {

        System.out.println("Drawing a square");

    }

    public double calculateArea() {

        return side \* side;

    }

}

// Subclass Triangle

class Triangle extends Shape {

    private double base;

    private double height;

    public Triangle(double base, double height) {

        this.base = base;

        this.height = height;

    }

    public void draw() {

        System.out.println("Drawing a triangle");

    }

    public double calculateArea() {

        return 0.5 \* base \* height;

    }

}

public class Main {

    public static void main(String[] args) {

        Shape circle = new Circle(5.0);

        Shape square = new Square(4.0);

        Shape triangle = new Triangle(3.0, 6.0);

        circle.draw();

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

        System.out.println();

        square.draw();

        System.out.println("Square Area: " + square.calculateArea());

        System.out.println();

        triangle.draw();

        System.out.println("Triangle Area: " + triangle.calculateArea());

    }

}

**OUTPUT :**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-10>JAVAC Main.java**

**C:\Users\Argha Digar\Desktop\JAVA\E2L\ASSIGEMENT\ASS3\QUESTION-10>JAVA Main**

**Drawing a circle**

**Circle Area: 78.53981633974483**

**Drawing a square**

**Square Area: 16.0**

**Drawing a triangle**

**Triangle Area: 9.0**