**Aishwarya Nandkishore Sonawane LAB 3**

1. Create a superclass Person with attributes name and age, and a method display(). Create a subclass Student that adds an attribute studentID. Write a program to create a Student object and display all its attributes.

Package demo;

class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public void display() {

System.***out***.println("Name: " + name);

System.***out***.println("Age: " + age);

}

}

class Student extends Person {

private String studentID;

public Student(String name, int age, String studentID) {

super(name, age);

this.studentID = studentID;

}

@Override

public void display() {

super.display() ;

System.***out***.println("Student ID: " + studentID);

}

}

public class Main {

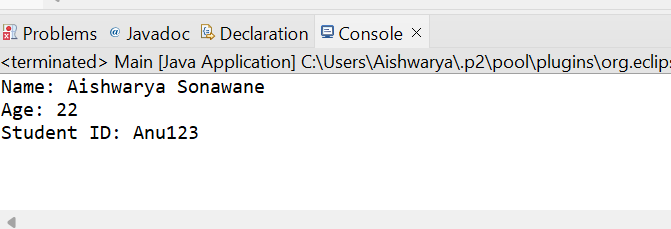
public **static** void main(String[] args) {

Student student = new Student("Aishwarya Sonawane", 22, "Anu123");

student.display();

}

}



2. Create a superclass Calculator with a method add(int a, int b). Create a subclass AdvancedCalculator that overloads the add method to handle three integers.

class Calculator {

public int add(int a, int b) {

return a + b;

}

}

class AdvancedCalculator extends Calculator {

public int add(int a, int b, int c) {

return a + b + c;

}

}

public class Main {

public **static** void main(String[] args) {

Calculator calc = new Calculator();

AdvancedCalculator advCalc = new AdvancedCalculator();

// Using Calculator's add method

int sum1 = calc.add(5, 10);

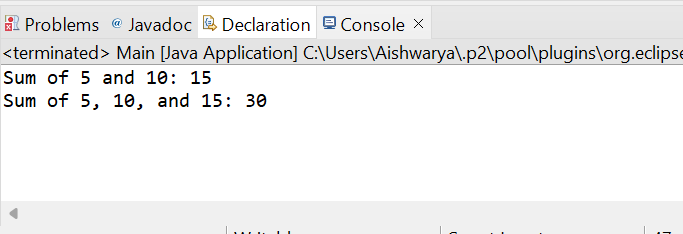
System.out.println("Sum of 5 and 10: " + sum1);

// Using AdvancedCalculator's overloaded add method

int sum2 = advCalc.add(5, 10, 15);

System.out.println("Sum of 5, 10, and 15: " + sum2);

}



3. Create a superclass Vehicle with a method move(). Create subclasses Car and Bike that inherit from Vehicle. Write a program to create objects of Car and Bike and call the move() method on each.// Superclass Vehicle

Package demo;

// Superclass Vehicle

class Vehicle {

public void move() {

System.***out***.println("The vehicle is moving");

}

}

// Subclass Car

class Car extends Vehicle {

@Override

public void move() {

System.***out***.println("The car is driving");

}

}

// Subclass Bike

class Bike extends Vehicle {

@Override

public void move() {

System.***out***.println("The bike is riding");

}

}

// Main class to test the program

public class Main {

public **static** void main(String[] args)

{

// Creating objects of Car and Bike

Vehicle myCar = new Car();

Vehicle myBike = new Bike();

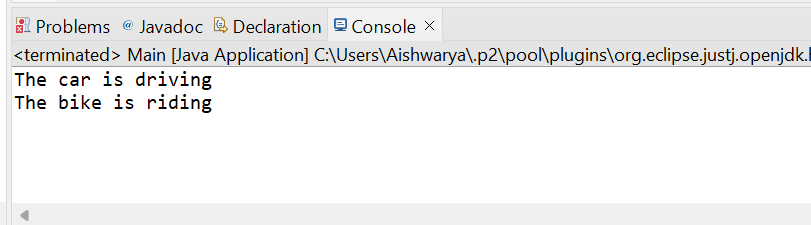
// Calling the move() method on each object

myCar.move();

myBike.move();

}

}



4. Create an class Employee with an abstract method calculatePay(). Create subclasses SalariedEmployee and HourlyEmployee that implement the calculatePay() method. Write a program to create objects of both subclasses and call the calculatePay() method.

Package demo;

// Define the abstract class Employee

**abstract** **class** Employee {

// Abstract method calculatePay

**public** **abstract** **double** calculatePay();

}

// Subclass SalariedEmployee

**class** SalariedEmployee **extends** Employee {

**private** **double** salary;

// Constructor

**public** SalariedEmployee(**double** salary) {

**this**.salary = salary;

}

// Implement calculatePay method

@Override

**public** **double** calculatePay() {

// For simplicity, assume monthly salary

**return** salary;

}

}

// Subclass HourlyEmployee

**class** HourlyEmployee **extends** Employee {

**private** **double** hourlyRate;

**private** **double** hoursWorked;

// Constructor

**public** HourlyEmployee(**double** hourlyRate, **double** hoursWorked) {

**this**.hourlyRate = hourlyRate;

**this**.hoursWorked = hoursWorked;

}

// Implement calculatePay method

@Override

**public** **double** calculatePay() {

// For simplicity, assume no overtime pay

**return** hourlyRate \* hoursWorked;

}

}

// Main class to test the implementation

**public** **class** EmployeePay {

**public** **static** **void** main(String[] args) {

// Create objects of SalariedEmployee and HourlyEmployee

SalariedEmployee salariedEmployee = **new** SalariedEmployee(5000);

HourlyEmployee hourlyEmployee = **new** HourlyEmployee(15, 40);

// 15 per hour, worked 40 hours

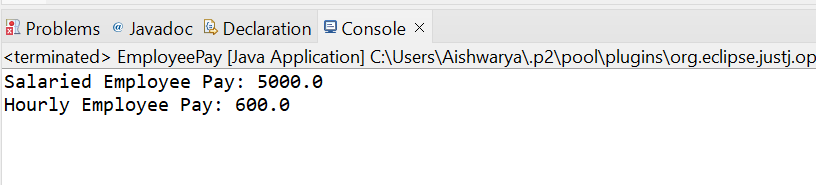
// Call calculatePay method for each employee

System.***out***.println("Salaried Employee Pay: " + salariedEmployee.calculatePay());

System.***out***.println("Hourly Employee Pay: " + hourlyEmployee.calculatePay());

}

}



5. Create an class Document with an method void open(). Implement subclasses WordDocument, PDFDocument, and SpreadsheetDocument that extend Document and provide implementations for open(). Write a main class to demonstrate opening different types of documents.(implement complile time- polymorphism).

// Define the Document class

class Document {

// Method to open the document

public void **open**() {

System.***out***.println("Opening document...");

}

}

// Subclass WordDocument extending Document

class WordDocument extends Document {

// Override the open method

@Override

public void **open**() {

System.***out***.println("Opening Word document...");

}

}

// Subclass PDFDocument extending Document

class PDFDocument extends Document {

// Override the open method

@Override

public void **open**() {

System.***out***.println("Opening PDF document...");

}

}

// Subclass SpreadsheetDocument extending Document

class SpreadsheetDocument extends Document {

// Override the open method

@Override

public void **open**() {

System.***out***.println("Opening Spreadsheet document...");

}

}

// Main class to demonstrate opening different types of documents

public class Main {

public **static** void main(String[] args) {

// Create instances of different document types

Document wordDoc = new WordDocument();

Document pdfDoc = new PDFDocument();

Document spreadsheetDoc = new SpreadsheetDocument();

// Demonstrate polymorphism by calling the open method on each document

*openDocument*(wordDoc);

*openDocument*(pdfDoc);

*openDocument*(spreadsheetDoc);

}

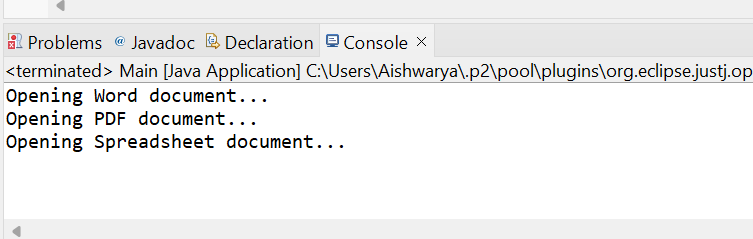
// Method to open a document (compile-time polymorphism)

public **static** void openDocument(Document doc) {

doc.**open**();

}

}



6.Create a class Calculator with overloaded methods add() that take different numbers and types of parameters: int add(int a, int b)

double add(double a, double b) int add(int a, int b, int c) Write a main class to demonstrate the usage of these methods.

class Calculator {

// Method to add two integers

int add(int a, int b) {

return a + b;

}

// Method to add two doubles

double add(double a, double b) {

return a + b;

}

// Method to add three integers

int add(int a, int b, int c) {

return a + b + c;

}

}

public class Integers {

public **static** void main(String[] args) {

Calculator calc = new Calculator();

// Demonstrating different add() methods

int sum1 = calc.add(5, 3);

double sum2 = calc.add(3.5, 4.2);

int sum3 = calc.add(2, 4, 6);

// Displaying the results

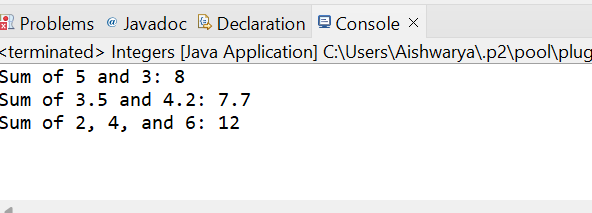
System.***out***.println("Sum of 5 and 3: " + sum1);

System.***out***.println("Sum of 3.5 and 4.2: " + sum2);

System.***out***.println("Sum of 2, 4, and 6: " + sum3);

}

}



7. Create a JavaBean class Person with properties firstName, lastName, age, and email. Implement the required no-argument constructor, getter and setter methods for each property. Write a main class to create an instance of Person, set its properties, and print them out.

Package demo;

class Person {

private String firstName;

private String lastName;

private int age;

private String email;

// No-argument constructor

public Person() {

}

// Getter and setter methods for firstName

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

// Getter and setter methods for lastName

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

// Getter and setter methods for age

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

// Getter and setter methods for email

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

public class javabeanclass {

public **static** void main(String[] args) {

// Creating an instance of Person

Person person = new Person();

// Setting properties

person.setFirstName("Aishwarya");

person.setLastName("Sonawane");

person.setAge(22);

[person.setEmail("aishusonawane2@example.com](mailto:person.setEmail("aishusonawane2@example.com)");

// Printing out the properties

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

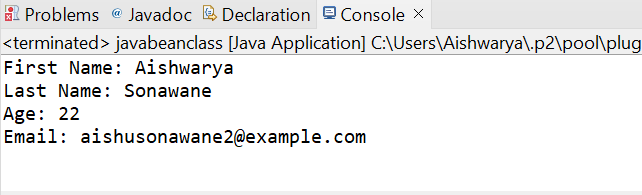
System.***out***.println("Last Name: " + person.getLastName());

System.***out***.println("Age: " + person.getAge());

System.***out***.println("Email: " + person.getEmail());

}

}



8.Create a JavaBean class Car with properties make, model, year, and color. Implement the required no-argument constructor, getter and setter methods for each property. Write a main class to create an instance of Car, set its properties, and print the car details.

Package demo;

class Car {

private String make;

private String model;

private int year;

private String color;

// No-argument constructor

public Car() {

}

// Getter and setter methods for make

public String getMake() {

return make;

}

public void setMake(String make) {

this.make = make;

}

// Getter and setter methods for model

public String getModel() {

return model;

}

public void setModel(String model) {

this.model = model;

}

// Getter and setter methods for year

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

// Getter and setter methods for color

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

}

public class javabeanclass {

public **static** void main(String[] args) {

// Creating an instance of Car

Car car = new Car();

// Setting properties

car.setMake("Renault");

car.setModel("Triber Limited Edition");

car.setYear(2024);

car.setColor("Silver");

// Printing out the car details

System.***out***.println("Make: " + car.getMake());

System.***out***.println("Model: " + car.getModel());

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

System.***out***.println("Color: " + car.getColor());

}

}

