LAB 3 ANP-C7781

Solve following questions:

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.

Program :-

package Day07;

public class PersonStudentClass {

// Person class created

public static class Person {

String name;

int age;

public void display() {

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

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

}

}

// Student class created and extends Person class

public static class Student extends Person {

int studID;

public void displayStud() {

display();

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

}

}

// In Main method student object is created to access all attributes/Variables From Person and Student Class

public static void main(String[] args) {

Student student = new Student();

student.name = "Pranay";

student.age = 21;

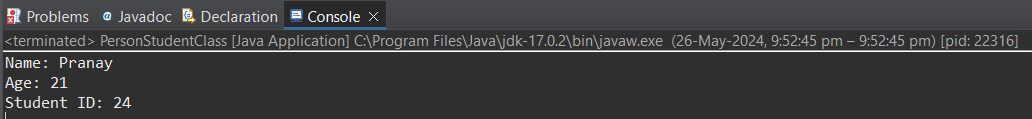
student.studID = 24;

student.displayStud();

}

}

Output :-



1. 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.

Program :-

package Day07;

public class CalculatorProgramWithClass {

// Calculator class created

public static class Calculator {

// Two parameter add function created

public int add(int num1, int num2) {

int result = num1 + num2;

System.***out***.println(result);

return result;

}

}

// AdvancedCalculator class created and extends Calculator class

public static class AdvancedCalculator extends Calculator {

// Three parameter add function created

public int add(int num1, int num2, int num3) {

int result = num1 + num2 + num3;

System.***out***.println(result);

return result;

}

}

public static void main(String[] args) {

// calculator object created

AdvancedCalculator calc = new AdvancedCalculator();

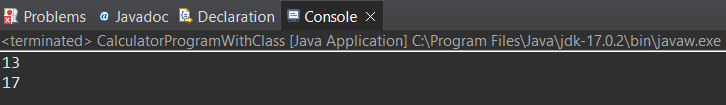
calc.add(5, 8);

calc.add(5, 8, 4);

}

}

Output :-



1. 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.

Program:-

package Day07;

public class VehicleClassesProgram {

// Vehicle class and move method created

public static class Vehicle {

public void move() {

System.***out***.println("Vehicle is moving.");

}

}

// Car class created and extends Vehicle and move method created

public static class Car extends Vehicle {

public void move() {

System.***out***.println("Car is moving.");

}

}

// Bike class created and extends Vehicle and move method created

public static class Bike extends Vehicle {

public void move() {

System.***out***.println("Bike is moving.");

}

}

public static void main(String[] args) {

// Car and Bike objects and move method invoked

Car car = new Car();

Bike bike = new Bike();

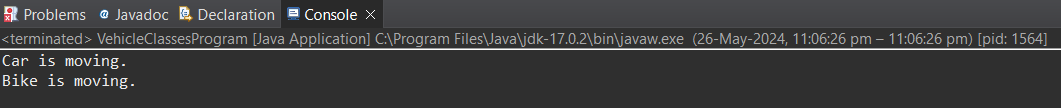
car.move();

bike.move();

}

}

Output :-



1. 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.

Program :-

package Day07;

public class EmployeeAbstractClassProgram {

// Abstract employee class created

public static abstract class Employee {

String name;

double salary;

double hoursWorked;

double hourlyRate;

// Abstract method to calculate pay

public abstract double calculatePay();

}

// SalariedEmployee class created

static class SalariedEmployee extends Employee {

public SalariedEmployee(String name, double salary) {

this.name = name;

this.salary = salary;

}

*@Override*

public double calculatePay() {

return salary;

}

}

// HourlyEmployee class created

static class HourlyEmployee extends Employee {

public HourlyEmployee(String name, double hourlyRate, double hoursWorked) {

this.name = name;

this.hourlyRate = hourlyRate;

this.hoursWorked = hoursWorked;

}

*@Override*

public double calculatePay() {

return hourlyRate \* hoursWorked;

}

}

public static void main(String[] args) {

// Created a SalariedEmployee object and calculate pay

SalariedEmployee salariedEmployee = new SalariedEmployee("Pranay", 45000);

double salariedPay = salariedEmployee.calculatePay();

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

// Created an HourlyEmployee object and calculate pay

HourlyEmployee hourlyEmployee = new HourlyEmployee("Pranay", 2000, 9);

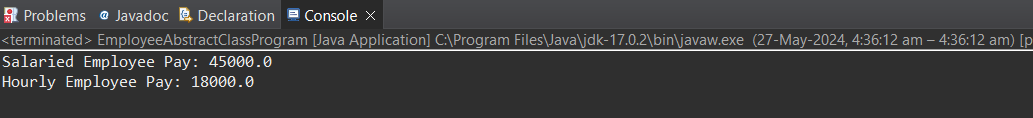
double hourlyPay = hourlyEmployee.calculatePay();

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

}

}

Output :-



1. 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).

Program :-

package Day07;

public class DocumentClassProgram {

// Method to open a Document using doc class

public static void openDocument(Document doc) {

doc.open();

}

// Document class created

public static class Document {

public void open() {

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

}

}

// WordDocument class created

public static class WordDocument extends Document {

*@Override*

public void open() {

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

}

}

// PDFDocument class created

public static class PDFDocument extends Document {

*@Override*

public void open() {

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

}

}

// SpreadsheetDocument class created

public static class SpreadsheetDocument extends Document {

*@Override*

public void open() {

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

}

}

public static void main(String[] args) {

// Created Document objects of different types

Document doc1 = new WordDocument();

Document doc2 = new PDFDocument();

Document doc3 = new SpreadsheetDocument();

// Calling the openDocument method with different Document objects

*openDocument*(doc1);

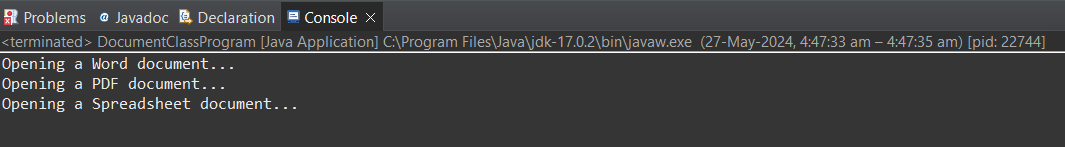
*openDocument*(doc2);

*openDocument*(doc3);

}

}

Output :-



1. 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.

Program :-

package Day07;

public class OverloadMethodsCalculator {

// Calculator class created with Overloaded add functions

public static class Calculator {

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

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

return a + b + c;

}

public static void main(String[] args) {

Calculator calculator = new Calculator();

// Calling and displaying the result of the overloaded add() functions

System.***out***.println(calculator.add(5, 10));

System.***out***.println(calculator.add(3.5, 2.5));

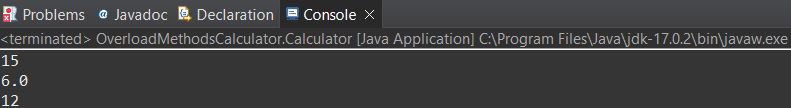
System.***out***.println(calculator.add(2, 4, 6));

}

}

}

Output :-



1. 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.

Program :-

package Day07;

public class JavabeanPersonClassGETSETProgram {

// Person class created

public static class Person {

private String firstName;

private String lastName;

private int age;

private String email;

public Person() {

// No-argument constructor

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

public static void main(String[] args) {

// Create an object of Person

Person person = new Person();

// Set the properties of the Person object

person.setFirstName("Pranay");

person.setLastName("Surve");

person.setAge(21);

person.setEmail("Pranay@gmail.com");

// Print (Get) the properties of the Person object

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

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

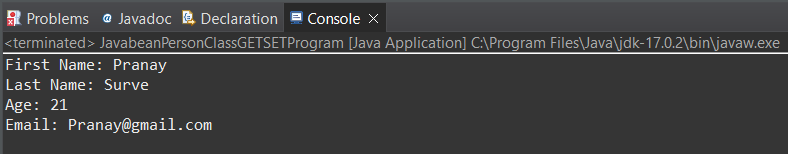
System.***out***.println("Age: " + person.getAge());

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

}

}

Output :-



1. 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.

Program :-

package Day07;

public class javabeanCarClassGETSETProgram {

// Car class created with Getter and Setter methods

public static class Car {

private String make;

private String model;

private int year;

private String color;

public Car() {

// No-argument constructor

}

public String getMake() {

return make;

}

public void setMake(String make) {

this.make = make;

}

public String getModel() {

return model;

}

public void setModel(String model) {

this.model = model;

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

}

public static void main(String[] args) {

// Create an instance of Car

Car car = new Car();

// Set the properties of the Car object

car.setMake("Maruti Suzuki");

car.setModel("Swift");

car.setYear(2024);

car.setColor("Silver");

// Print the details of the Car object

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

}

}

Output :-

