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.

Code:

**package** Lab3;

//Define the superclass Person

**class** Person {

// Attributes of the Person class

**protected** String name;

**protected** **int** age;

// Constructor of the Person class

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

// Method to display the attributes of the Person class

**public** **void** display() {

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

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

}

}

//Define the subclass Student that extends Person

**class** Student **extends** Person {

// Additional attribute for the Student class

**private** String studentID;

// Constructor of the Student class

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

**super**(name, age); // Call the constructor of the superclass Person

**this**.studentID = studentID;

}

// Method to display the attributes of the Student class

@Override

**public** **void** display() {

**super**.display(); // Call the display method of the superclass Person

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

}

}

**public** **class** encapsulation {

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

// Create a Student object

Student student = **new** Student("Abhishek", 20, "S1234567");

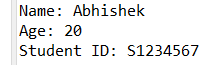
// Display the attributes of the Student object

student.display();

}

}

Output:



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.

Code:

**package** Lab3;

//Define the superclass Calculator

**class** Calc {

// Method to add two integers

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

**return** a + b;

}

}

//Define the subclass AdvancedCalculator that extends Calculator

**class** AdvancedCalculator **extends** Calculator {

// Overloaded method to add three integers

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

**return** a + b + c;

}

}

//Main class to test the functionality

**public** **class** Calculator {

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

// Create an instance of Calculator

Calc calculator = **new** Calc();

// Add two integers using the Calculator class

**int** result1 = calculator.add(10, 20);

System.***out***.println("Calculator result: " + result1);

// Create an instance of AdvancedCalculator

AdvancedCalculator advancedCalculator = **new** AdvancedCalculator();

// Add two integers using the AdvancedCalculator class (inherited method)

**int** result2 = advancedCalculator.add(30, 40, 50);

System.***out***.println("Advance Calculator, inherited: " + result2);

// Add three integers using the AdvancedCalculator class (overloaded method)

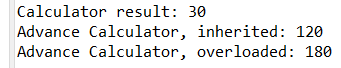
**int** result3 = advancedCalculator.add(50, 60, 70);

System.***out***.println("Advance Calculator, overloaded: " + result3);

}

}

Output:



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.

Code:

**package** Lab3;

//Define the superclass Vehicle

**class** Vehicle {

// Method to be overridden by subclasses

**public** **void** move() {

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

}

}

//Define the subclass Car that extends Vehicle

**class** Car **extends** Vehicle {

// Override the move method for Car

@Override

**public** **void** move() {

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

}

}

//Define the subclass Bike that extends Vehicle

**class** Bike **extends** Vehicle {

// Override the move method for Bike

@Override

**public** **void** move() {

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

}

}

//Main class to test the functionality

**public** **class** vahicle {

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

// Create an instance of Car

Car car = **new** Car();

// Call the move method on the Car object

car.move();

// Create an instance of Bike

Bike bike = **new** Bike();

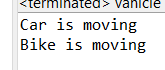
// Call the move method on the Bike object

bike.move();

}

}

Output:



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.

Code:

**package** Lab3;

//Define the abstract class Employee

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

// Abstract method to be implemented by subclasses

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

}

//Define the subclass SalariedEmployee that extends Employee

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

// Attribute for the annual salary

**private** **double** annualSalary;

// Constructor to initialize the annual salary

**public** SalariedEmployee(**double** annualSalary) {

**this**.annualSalary = annualSalary;

}

// Implement the calculatePay method for salaried employees

@Override

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

// For a salaried employee, the monthly pay is the annual salary divided by 12

**return** annualSalary / 12;

}

}

//Define the subclass HourlyEmployee that extends Employee

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

// Attributes for the hourly wage and hours worked

**private** **double** hourlyWage;

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

// Constructor to initialize the hourly wage and hours worked

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

**this**.hourlyWage = hourlyWage;

**this**.hoursWorked = hoursWorked;

}

// Implement the calculatePay method for hourly employees

@Override

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

// For an hourly employee, the pay is the hourly wage multiplied by the hours worked

**return** hourlyWage \* hoursWorked;

}

}

//Main class to test the functionality

**public** **class** emp\_pay {

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

// Create an instance of SalariedEmployee

SalariedEmployee salariedEmployee = **new** SalariedEmployee(60000);

// Call the calculatePay method on the SalariedEmployee object

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

// Create an instance of HourlyEmployee

HourlyEmployee hourlyEmployee = **new** HourlyEmployee(50, 8);

// Call the calculatePay method on the HourlyEmployee object

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

}

}

Output:



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).

Code:

**package** Lab3;

//Define the superclass Document

**class** Document {

// Method to be overridden by subclasses

**public** **void** open() {

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

}

// Overloaded open method with a filename parameter for compile-time polymorphism

**public** **void** open(String filename) {

System.***out***.println("Opening document: " + filename);

}

}

//Define the subclass WordDocument that extends Document

**class** WordDocument **extends** Document {

// Override the open method for WordDocument

@Override

**public** **void** open() {

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

}

// Overloaded open method with a filename parameter for compile-time polymorphism

@Override

**public** **void** open(String filename) {

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

}

}

//Define the subclass PDFDocument that extends Document

**class** PDFDocument **extends** Document {

// Override the open method for PDFDocument

@Override

**public** **void** open() {

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

}

// Overloaded open method with a filename parameter for compile-time polymorphism

@Override

**public** **void** open(String filename) {

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

}

}

//Define the subclass SpreadsheetDocument that extends Document

**class** SpreadsheetDocument **extends** Document {

// Override the open method for SpreadsheetDocument

@Override

**public** **void** open() {

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

}

// Overloaded open method with a filename parameter for compile-time polymorphism

@Override

**public** **void** open(String filename) {

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

}

}

//Main class to test the functionality

**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();

// Call the open method on each document object

wordDoc.open();

pdfDoc.open();

spreadsheetDoc.open();

// Call the overloaded open method with a filename on each document object

wordDoc.open("example.docx");

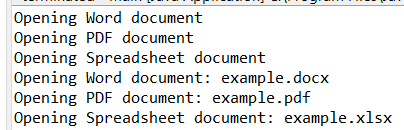
pdfDoc.open("example.pdf");

spreadsheetDoc.open("example.xlsx");

}

}

Output:



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.

Code:

**package** Lab3;

//Define the Calculator class

**class** Calculator1 {

// Method to add two integers

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

**return** a + b;

}

// Method to add two double values

**public** **double** add(**double** a, **double** b) {

**return** a + b;

}

// Method to add three integers

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

**return** a + b + c;

}

}

//Main class to test the functionality

**public** **class** calsi {

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

// Create an instance of Calculator

Calculator1 calculator = **new** Calculator1();

// Call the add method with two integers

**int** sum1 = calculator.add(10, 20);

System.***out***.println("Sum of 10 and 20: " + sum1);

// Call the add method with two double values

**double** sum2 = calculator.add(15.5, 4.5);

System.***out***.println("Sum of 15.5 and 4.5: " + sum2);

// Call the add method with three integers

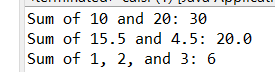
**int** sum3 = calculator.add(1, 2, 3);

System.***out***.println("Sum of 1, 2, and 3: " + sum3);

}

}

Output:



7.     Create a [JavaBean](https://aln.anudip.org/mod/resource/view.php?id=12692) 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.

Code:

**package** Lab3;

**import** java.io.Serializable;

**class** Person11 **implements** Serializable {

**private** String firstName;

**private** String lastName;

**private** **int** age;

**private** String email;

// No-argument constructor

**public** Person11() {

}

// Getter and Setter for firstName

**public** String getfirstName() {

**return** firstName;

}

**public** **void** setfirstName(String firstName) {

**this**.firstName = firstName;

}

// Getter and Setter for lastName

**public** String getlastName() {

**return** lastName;

}

**public** **void** setlastName(String lastName) {

**this**.lastName = lastName;

}

// Getter and Setter for age

**public** **int** getage() {

**return** age;

}

**public** **void** setage(**int** age) {

**this**.age = age;

}

// Getter and Setter for email

**public** String getemail() {

**return** email;

}

**public** **void** setemail(String email) {

**this**.email = email;

}

}

**public** **class** get\_set {

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

// Create an instance of Person

Person11 person = **new** Person11();

// Set properties

person.setfirstName("Abhi");

person.setlastName("Gupta");

person.setage(30);

person.setemail("abhi.g@example.com");

// Print properties

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

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

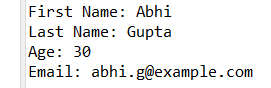
System.***out***.println("Age: " + person.getage());

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

}

}

Output:



8.     Create a [JavaBean](https://aln.anudip.org/mod/resource/view.php?id=12692) 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.

Code:

**package** Lab3;

**import** java.io.Serializable;

**class** Car1 **implements** Serializable {

**private** String make;

**private** String model;

**private** **int** year;

**private** String color;

// No-argument constructor

**public** Car1() {

}

// Getter and Setter for make

**public** String getmake() {

**return** make;

}

**public** **void** setmake(String make) {

**this**.make = make;

}

// Getter and Setter for model

**public** String getmodel() {

**return** model;

}

**public** **void** setmodel(String model) {

**this**.model = model;

}

// Getter and Setter for year

**public** **int** getyear() {

**return** year;

}

**public** **void** setyear(**int** year) {

**this**.year = year;

}

// Getter and Setter for color

**public** String getcolor() {

**return** color;

}

**public** **void** setcolor(String color) {

**this**.color = color;

}

}

**public** **class** getter\_setter {

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

// Create an instance of Car

Car1 c = **new** Car1();

// Set properties

c.setmake("Toyota");

c.setmodel("Corolla");

c.setyear(2021);

c.setcolor("Red");

// Print car details

System.***out***.println("Car Make: " + c.getmake());

System.***out***.println("Car Model: " + c.getmodel());

System.***out***.println("Car Year: " + c.getyear());

System.***out***.println("Car Color: " + c.getcolor());

}

}

Output:

