LAB 3 (JAVA)

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:**

**class** Person {

String name;

**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 {

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** tst {

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

Student student1 = **new** Student("SUMI", 20, "S12345");

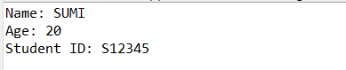
student1.display();

}

}

**OUTPUT :**

**This code defines a Person class with attributes name and age, along with a display() method to print those attributes. The Student class inherits from the Person class and adds an additional attribute studentID. The display() method of the Student class overrides the display() method of the Person class to include the studentID attribute. Finally, a Student object is created and its attributes are displayed using the display() method.**

****

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** bhargavi;

**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 basicCalculator = **new** Calculator();

System.***out***.println("Result of basic addition: " + basicCalculator.add(5, 7));

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

System.***out***.println("Result of advanced addition: " + advancedCalculator.add(3, 6, 9));

}

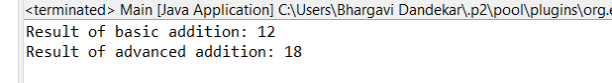
}

**OUTPUT:**

**The Calculator class has a method add(int a, int b) that adds two integers.**

**The AdvancedCalculator class extends Calculator and overloads the add method to handle three integers.**

**In the Main class, instances of both classes are created and the add methods are invoked to demonstrate basic and advanced addition functionalities.**

****

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** bhargavi;

**class** Vehicle {

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

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

}

}

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

// Car-specific methods and attributes can be added here

}

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

// Bike-specific methods and attributes can be added here

}

**public** **class** Main {

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

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

car.move(); // Calls move method of Car

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

bike.move(); // Calls move method of Bike

}

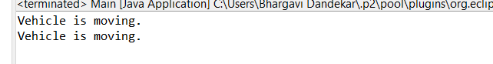
**}**

**OUTPUT :**

**The Vehicle class has a method move() that prints "Vehicle is moving."**

**The Car and Bike classes extend the Vehicle class, inheriting the move() method.**

**In the Main class, objects of Car and Bike are created and their move() methods are called, demonstrating polymorphism where the appropriate move() method of each subclass is invoked.**



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** bhargavi;

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

// Abstract method to calculate pay

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

}

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

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

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

**this**.salary = salary;

}

// Implementing calculatePay() method for SalariedEmployee

@Override

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

**return** salary;

}

}

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

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

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

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

**this**.hourlyRate = hourlyRate;

**this**.hoursWorked = hoursWorked;

}

// Implementing calculatePay() method for HourlyEmployee

@Override

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

**return** hourlyRate \* hoursWorked;

}

}

**public** **class** Main {

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

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

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

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

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

}

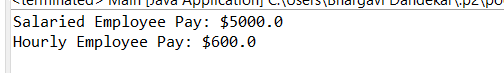
}

**OUTPUT:**

**The Employee class is abstract with an abstract method calculatePay().**

**The SalariedEmployee and HourlyEmployee classes are concrete subclasses of Employee that implement the calculatePay() method according to their specific payment rules.**

**In the Main class, objects of both subclasses are created, and their calculatePay() methods are called to calculate the pay for each type of employee.**

****

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** bhargavi;

**class** Document {

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

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

}

}

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

@Override

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

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

}

}

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

@Override

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

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

}

}

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

@Override

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

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

}

}

**public** **class** Main {

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

Document[] documents = {**new** WordDocument(), **new** PDFDocument(), **new** SpreadsheetDocument()};

**for** (Document doc : documents) {

doc.open();

}

}

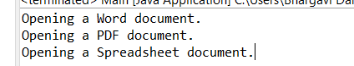
}

**OUTPUT:**

**The Document class has a method open() which is overridden by its subclasses.**

**The subclasses WordDocument, PDFDocument, and SpreadsheetDocument each provide their own implementation of the open() method.**

**In the Main class, an array of Document objects is created with instances of each subclass. By calling the open() method on each object, we demonstrate compile-time polymorphism, where the appropriate open() method for each subclass is invoked based on the reference type.**

****

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** bhargavi;

**public** **class** JAVA {

// Method to add two integers

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

**return** a + b;

}

// Method to add two doubles

**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;

}

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

JAVA calculator = **new** JAVA();

// Demonstrating usage of overloaded add() methods

System.***out***.println("Sum of two integers: " + calculator.add(5, 7));

System.***out***.println("Sum of two doubles: " + calculator.add(3.5, 6.7));

System.***out***.println("Sum of three integers: " + calculator.add(3, 6, 9));

}

}

**OUTPUT:**

**The Calculator class defines three overloaded add() methods:**

**add(int a, int b) for adding two integers.**

**add(double a, double b) for adding two doubles.**

**add(int a, int b, int c) for adding three integers.**

**In the main() method, these overloaded add() methods are called with different parameters to demonstrate their usage.**

****

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**.**

**CODE:**

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

// Create an instance of Person

Person person = **new** Person();

// Set properties

person.setFirstName("John");

person.setLastName("Doe");

person.setAge(30);

person.setEmail("john.doe@example.com");

// Print out 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());

}

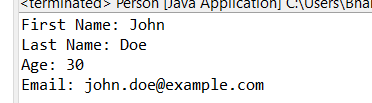
}

**OUTPUT:**

**The Person class is a JavaBean class with properties firstName, lastName, age, and email.**

**It provides a no-argument constructor and getter and setter methods for each property.**

**In the main() method, an instance of Person is created, its properties are set using setter methods, and then the properties are printed out using getter methods.**

****

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:**

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

// Create an instance of Car

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

// Set properties

car.setMake("Toyota");

car.setModel("Camry");

car.setYear(2022);

car.setColor("Red");

// Print out 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());

}

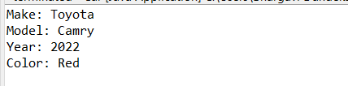
}

**OUTPUT:**

**The Car class is a JavaBean class with properties make, model, year, and color.**

**It provides a no-argument constructor and getter and setter methods for each property.**

**In the main() method, an instance of Car is created, its properties are set using setter methods, and then the car details are printed out using getter methods.**

****