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.

**Program:-**

package demo;

class Person {

// Attributes

private String name;

private int age;

// Constructor

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Method to display Person attributes

public void display() {

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

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

}

// Getters

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

// Define the subclass Student

class Student extends Person {

// Attribute specific to Student

private String studentID;

// Constructor

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

// Call the constructor of the superclass

super(name, age);

this.studentID = studentID;

}

// Method to display Student attributes

public void display() {

// Call the display method of the superclass

super.display();

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

}

// Getter

public String getStudentID() {

return studentID;

}

}

// Main class to test the functionality

public class Main {

public static void main(String[] args) {

// Create a Student object

Student student = new Student("John Doe", 20, "S123456");

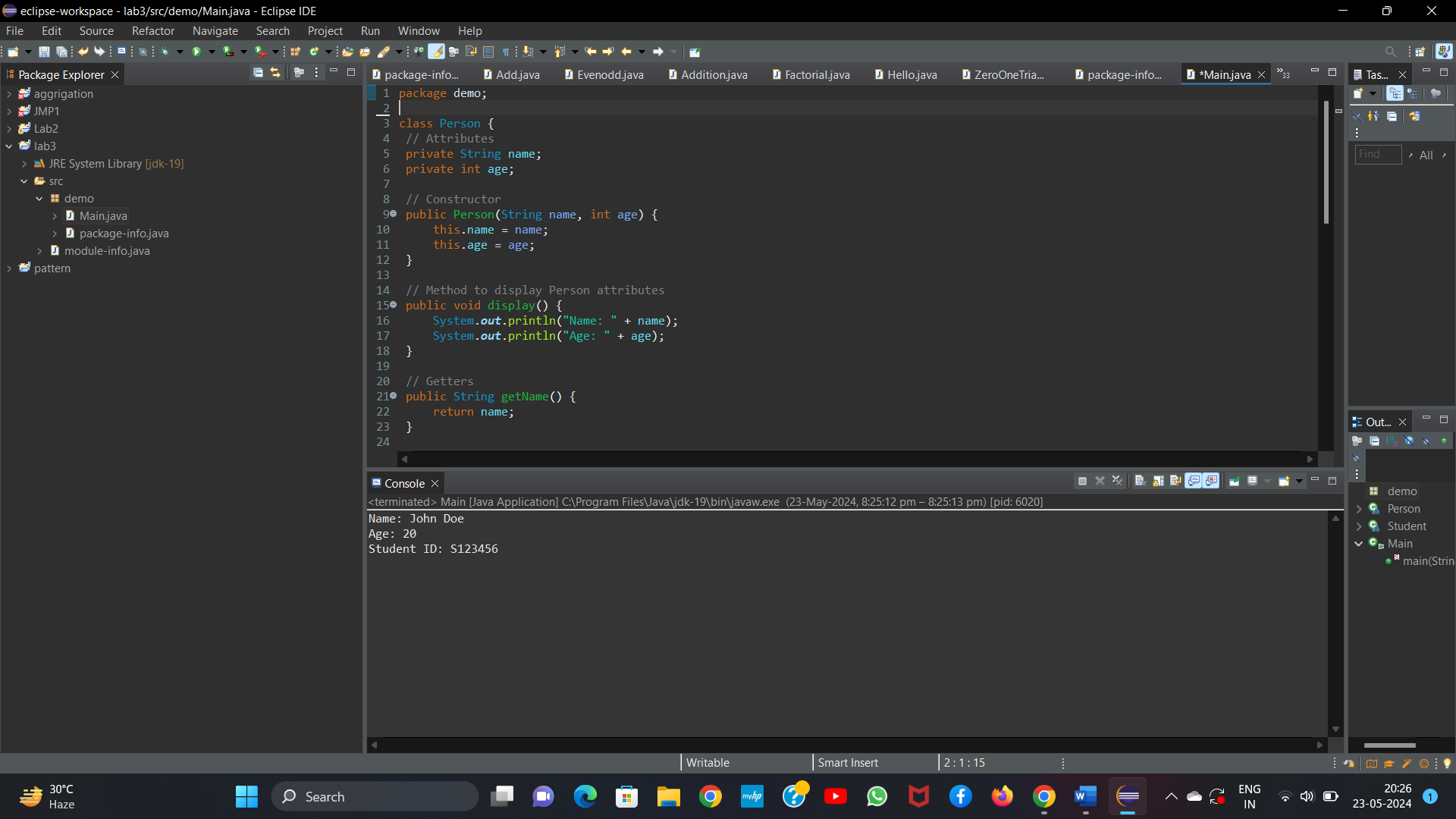
// Display all attributes of the Student

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.

**Program:-**

package demo;

class Calculator {

// Method to add two integers

public int add(int a, int b) {

return a + b;

}

}

// Define the subclass AdvancedCalculator

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 cal {

public static void main(String[] args) {

// Create an instance of Calculator

Calculator calculator = new Calculator();

int sumTwo = calculator.add(5, 10);

System.out.println("Sum of two numbers: " + sumTwo);

// Create an instance of AdvancedCalculator

AdvancedCalculator advancedCalculator = new AdvancedCalculator();

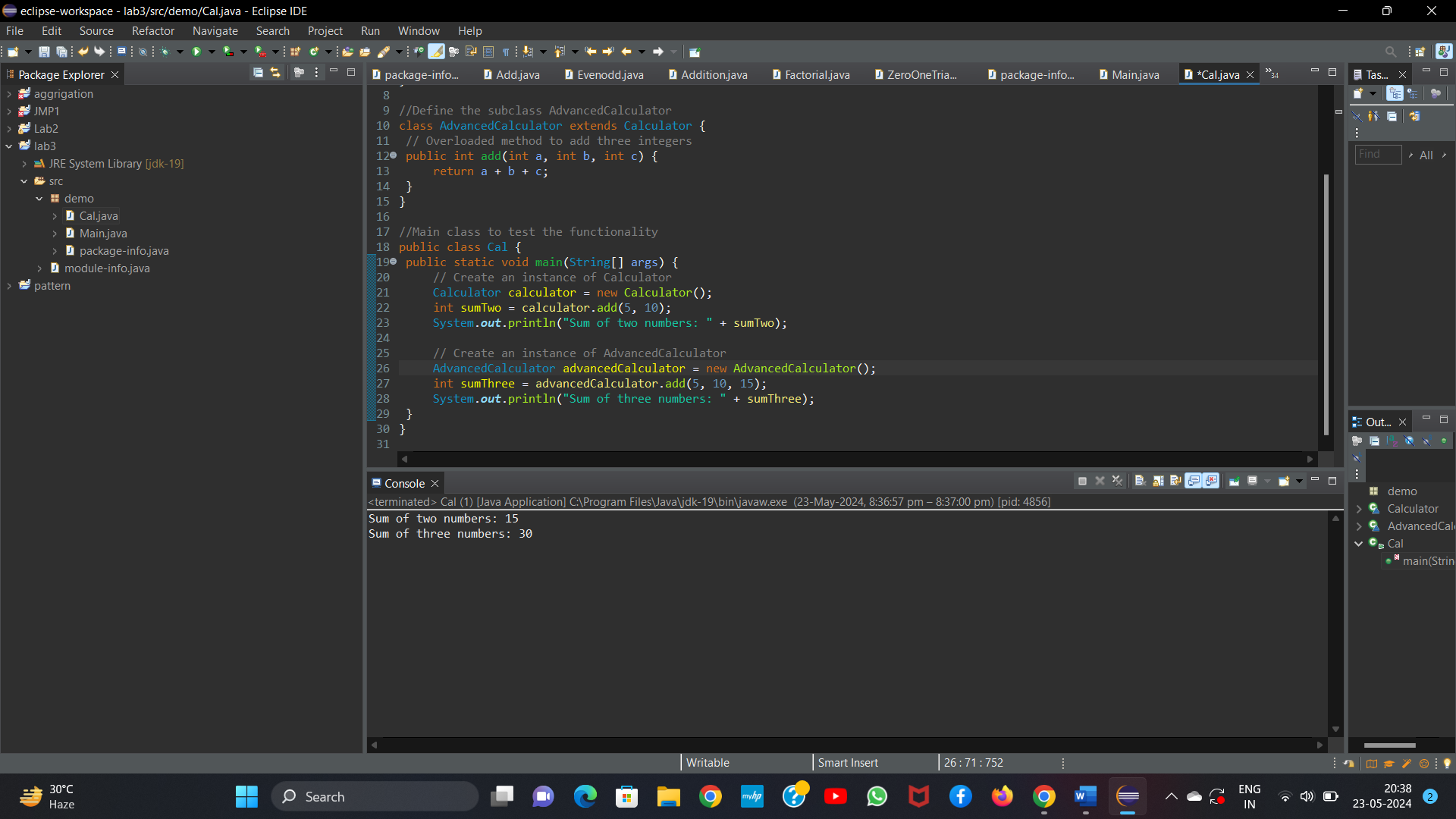
int sumThree = advancedCalculator.add(5, 10, 15);

System.out.println("Sum of three numbers: " + sumThree);

}

}

**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.

**Program:-**

package demo

class Vehicle {

// Method to be overridden by subclasses

public void move() {

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

}

}

// Define the subclass Car

class Car extends Vehicle {

// Override the move method

public void move() {

System.out.println("The car is moving");

}

}

// Define the subclass Bike

class Bike extends Vehicle {

// Override the move method

public void move() {

System.out.println("The bike is moving");

}

}

// Main class to test the functionality

public class Vehicles {

public static void main(String[] args) {

// Create an instance of Car

Car car = new Car();

// Call the move method on Car

car.move();

// Create an instance of Bike

Bike bike = new Bike();

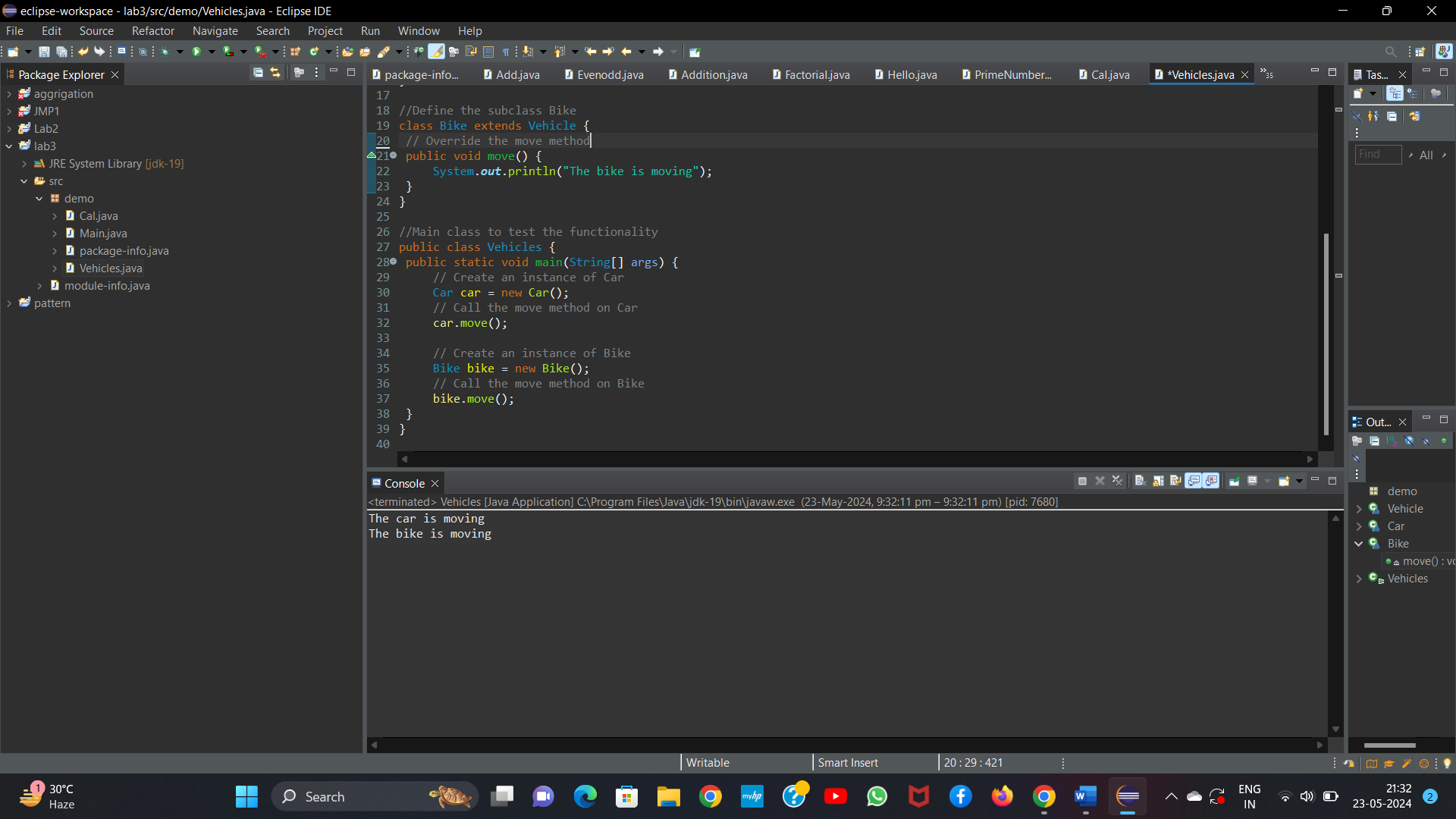
// Call the move method on Bike

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.

**Program:-**

package demo;

abstract class Employee {

// Abstract method to be implemented by subclasses

public abstract double calculatePay();

}

// Define the subclass SalariedEmployee

class SalariedEmployee extends Employee {

// Attributes specific to SalariedEmployee

private double annualSalary;

// Constructor

public SalariedEmployee(double annualSalary) {

this.annualSalary = annualSalary;

}

// Implement the calculatePay method

public double calculatePay() {

// Assuming pay is calculated on a monthly basis

return annualSalary / 12;

}

}

// Define the subclass HourlyEmployee

class HourlyEmployee extends Employee {

// Attributes specific to HourlyEmployee

private double hourlyRate;

private int hoursWorked;

// Constructor

public HourlyEmployee(double hourlyRate, int hoursWorked) {

this.hourlyRate = hourlyRate;

this.hoursWorked = hoursWorked;

}

// Implement the calculatePay method

public double calculatePay() {

return hourlyRate \* hoursWorked;

}

}

// Main class to test the functionality

public class Calculate {

public static void main(String[] args) {

// Create an instance of SalariedEmployee

SalariedEmployee salariedEmployee = new SalariedEmployee(60000);

// Call the calculatePay method on SalariedEmployee

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

// Create an instance of HourlyEmployee

HourlyEmployee hourlyEmployee = new HourlyEmployee(20, 160);

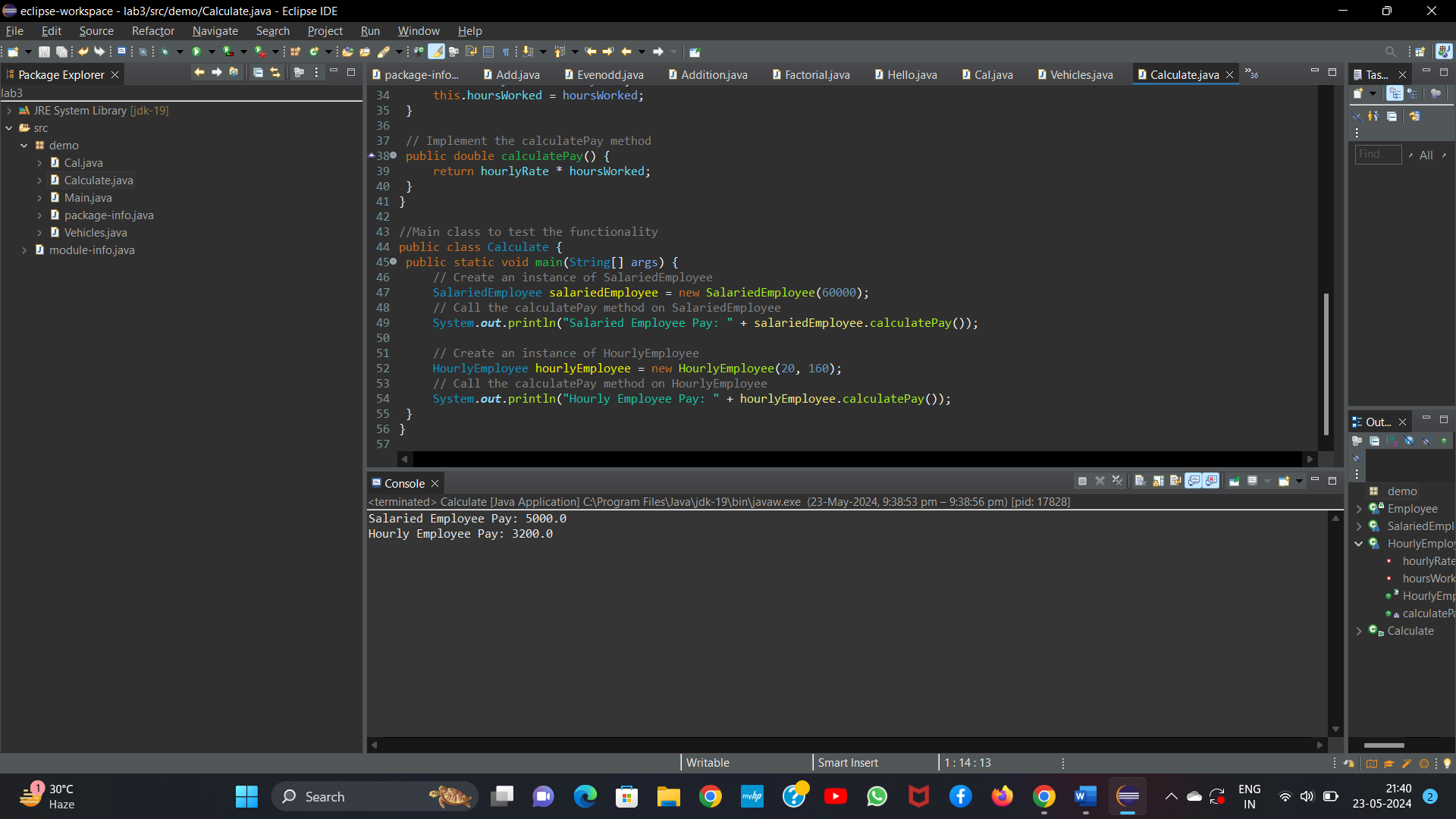
// Call the calculatePay method on HourlyEmployee

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).

**Program:-**

package demo

// Define the superclass Document

class Document {

// Method to be overridden by subclasses

public void open() {

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

}

}

// Define the subclass WordDocument

class WordDocument extends Document {

// Override the open method

public void open() {

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

}

}

// Define the subclass PDFDocument

class PDFDocument extends Document {

// Override the open method

public void open() {

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

}

}

// Define the subclass SpreadsheetDocument

class SpreadsheetDocument extends Document {

// Override the open method

public void open() {

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

}

}

// Main class to test the functionality

public class Main {

// Method to open documents demonstrating compile-time polymorphism

public static void openDocument(Document doc) {

doc.open();

}

public static void main(String[] args) {

// Create instances of each document type

Document wordDoc = new WordDocument();

Document pdfDoc = new PDFDocument();

Document spreadsheetDoc = new SpreadsheetDocument();

// Open each document

openDocument(wordDoc);

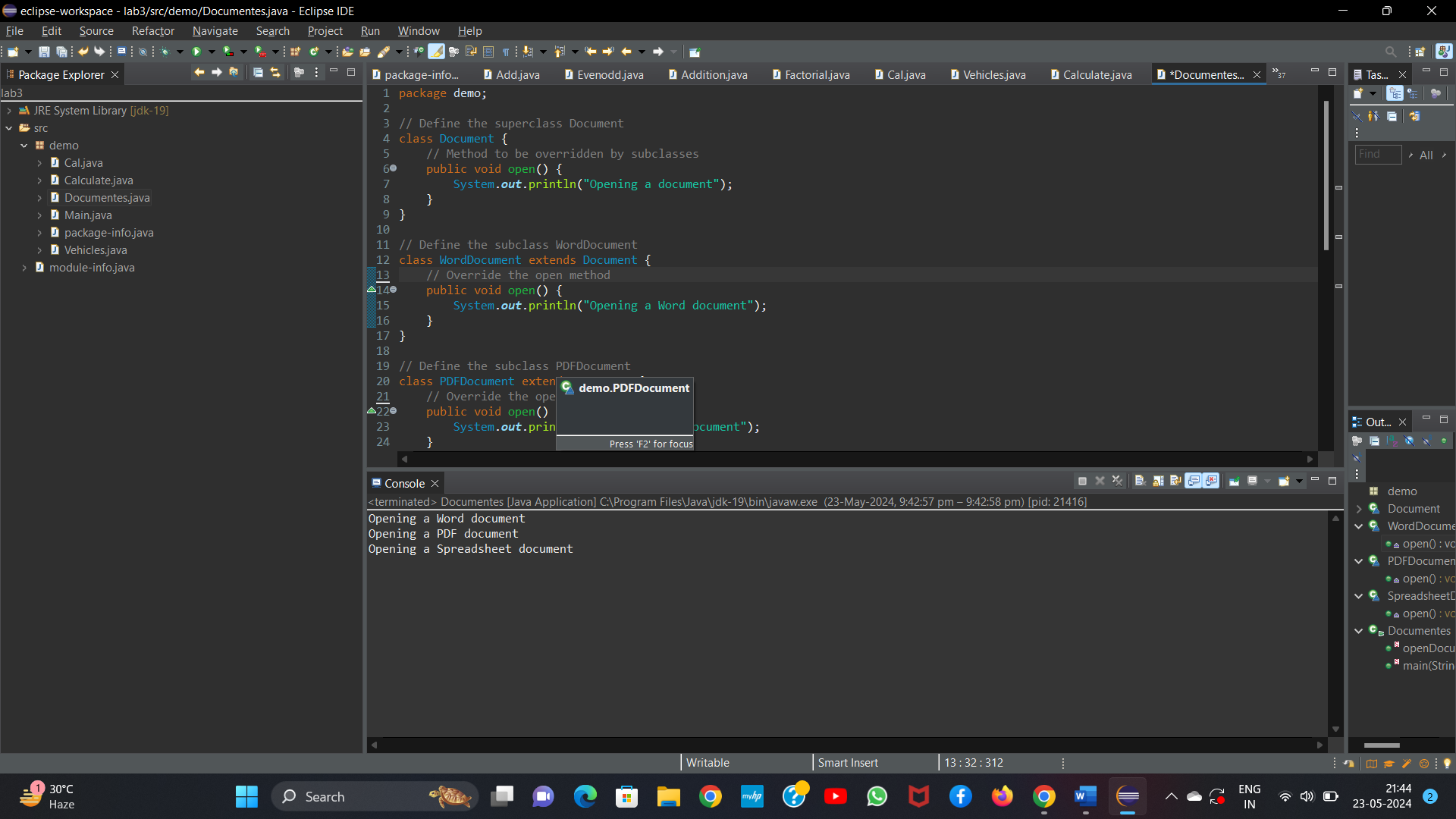
openDocument(pdfDoc);

openDocument(spreadsheetDoc);

}

}

**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.

**Program:-**

package demo;

class Calculator {

// Overloaded method to add two integers

public int add(int a, int b) {

return a + b;

}

// Overloaded method to add two doubles

public double add(double a, double b) {

return a + b;

}

// 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 Main {

public static void main(String[] args) {

// Create an instance of Calculator

Calculator calculator = new Calculator();

// Call the add method with two integers

int sumInt = calculator.add(5, 10);

System.out.println("Sum of two integers: " + sumInt);

// Call the add method with two doubles

double sumDouble = calculator.add(5.5, 10.5);

System.out.println("Sum of two doubles: " + sumDouble);

// Call the add method with three integers

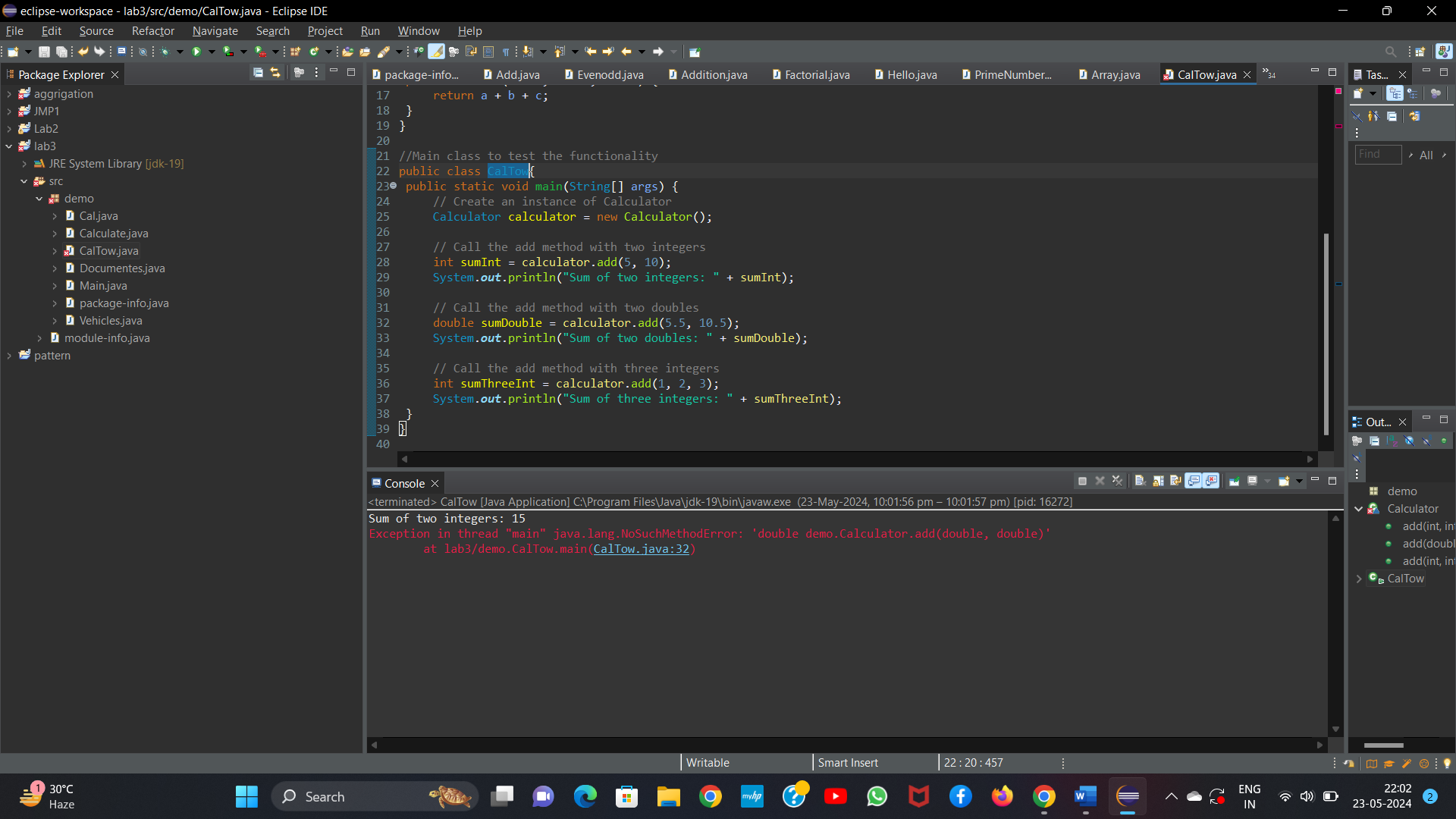
int sumThreeInt = calculator.add(1, 2, 3);

System.out.println("Sum of three integers: " + sumThreeInt);

}

}

**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.

Program:-

**Person.java**

package demo;

import java.io.Serializable;

public class Person implements Serializable {

// Properties

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;

}

}

**PersonMain.java**

package demo;

public class Main {

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");

// Get and print properties

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

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

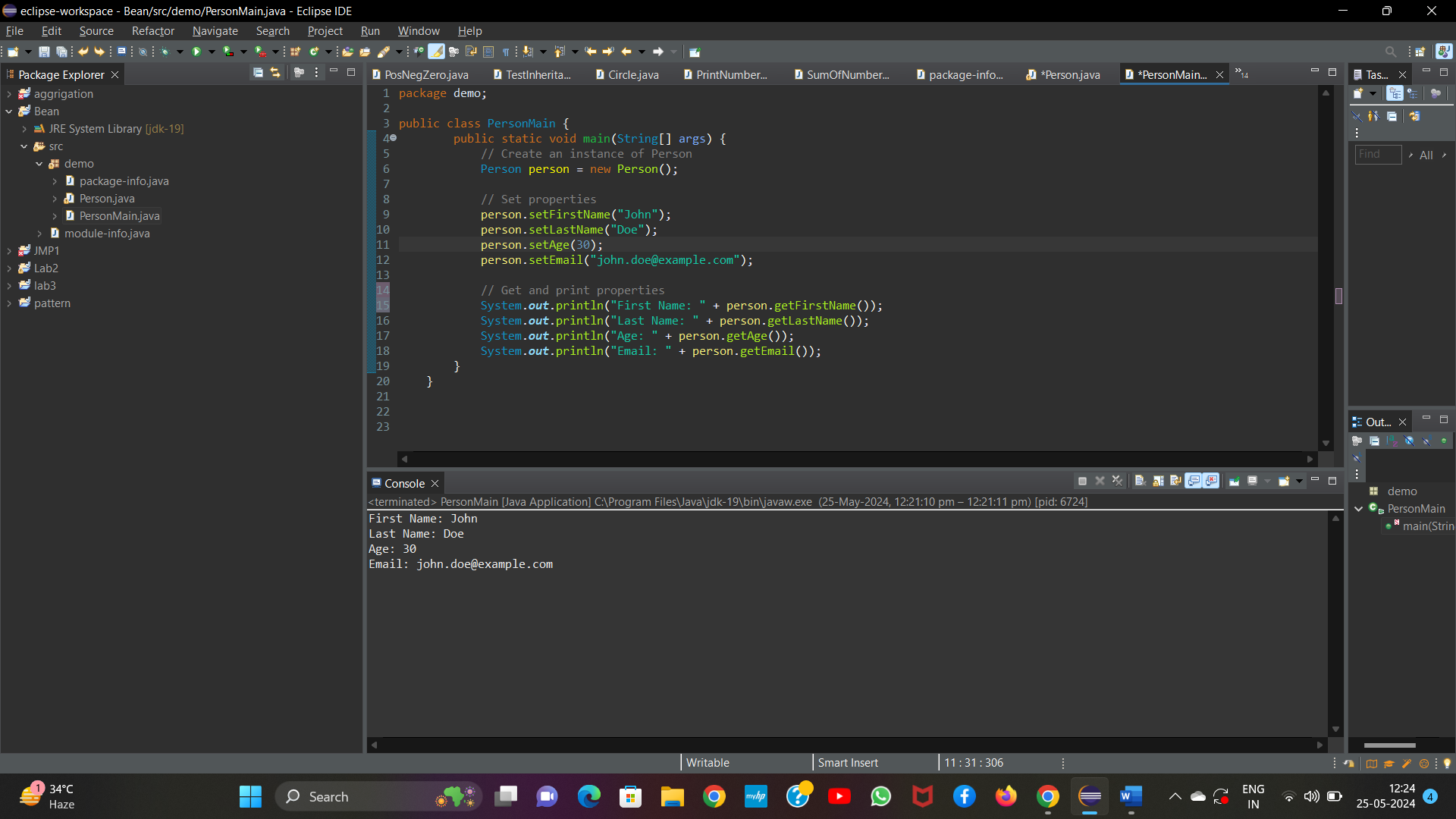
System.out.println("Age: " + person.getAge());

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.

**Program:**

**Car.java**

package demo;

import java.io.Serializable;

public class Car implements Serializable {

private String make;

private String model;

private int year;

private String color;

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;

}

}

**CarMain.java**

public class CarMain {

public static void main(String[] args) {

// Create an instance of Car

Car car = new Car();

// Set car properties

car.setMake("Toyota");

car.setModel("Camry");

car.setYear(2020);

car.setColor("Silver");

// Print car details

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:**

