**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;

//Superclass Person

**class** Person {

// Attributes

String name;

**int** age;

// Constructor

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

**this**.name = name;

**this**.age = age;

}

// Method to display information

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

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

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

}

}

//Subclass Student inheriting from Person

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

// Additional attribute

**int** studentID;

// Constructor

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

**super**(name, age);

**this**.studentID = studentID;

}

// Method to display student information

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

// Utilizing superclass display method

**super**.display();

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

}

}

//Main class

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

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

// Creating a Student object

Student student1 = **new** Student("John Doe", 20, 123456);

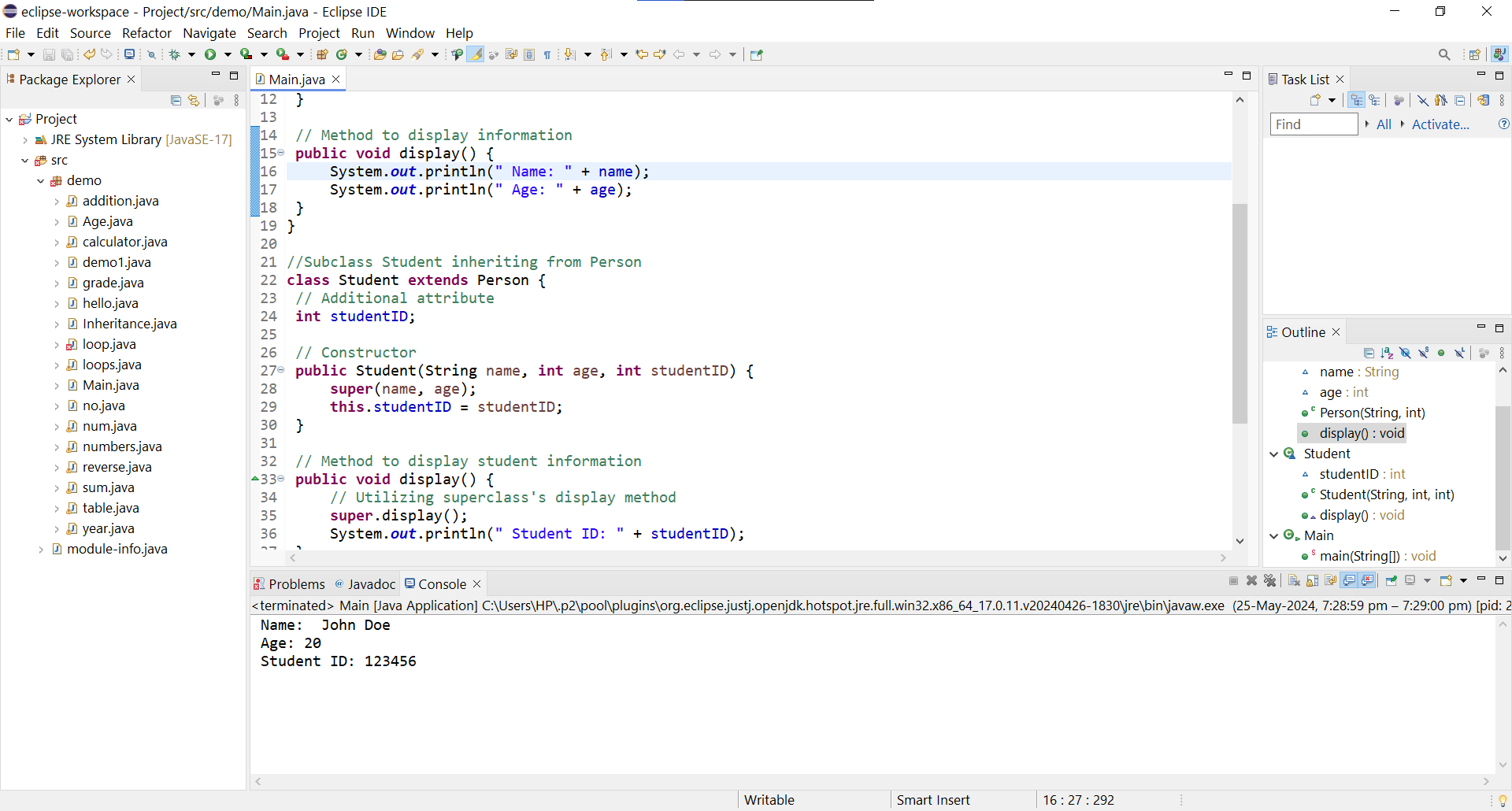
// Displaying all attributes of the student

student1.display();

}

}

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** demo;

//Superclass Calculator

**class** Calculator

{

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

{

**return** a+b;

}

}

//Subclass AdvCalculator

**class** AdvCalculator **extends** Calculator

{

//overloading the add method for adding three integers

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

{

**return** a+b+c;

}

}

//Main class

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

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

Calculator c=**new** Calculator();

AdvCalculator d=**new** AdvCalculator();

//using the add method

**int** s1=c.add(10, 20);

System.***out***.println(" Addition of two numbers: "+ s1);

//using the overloaded add method

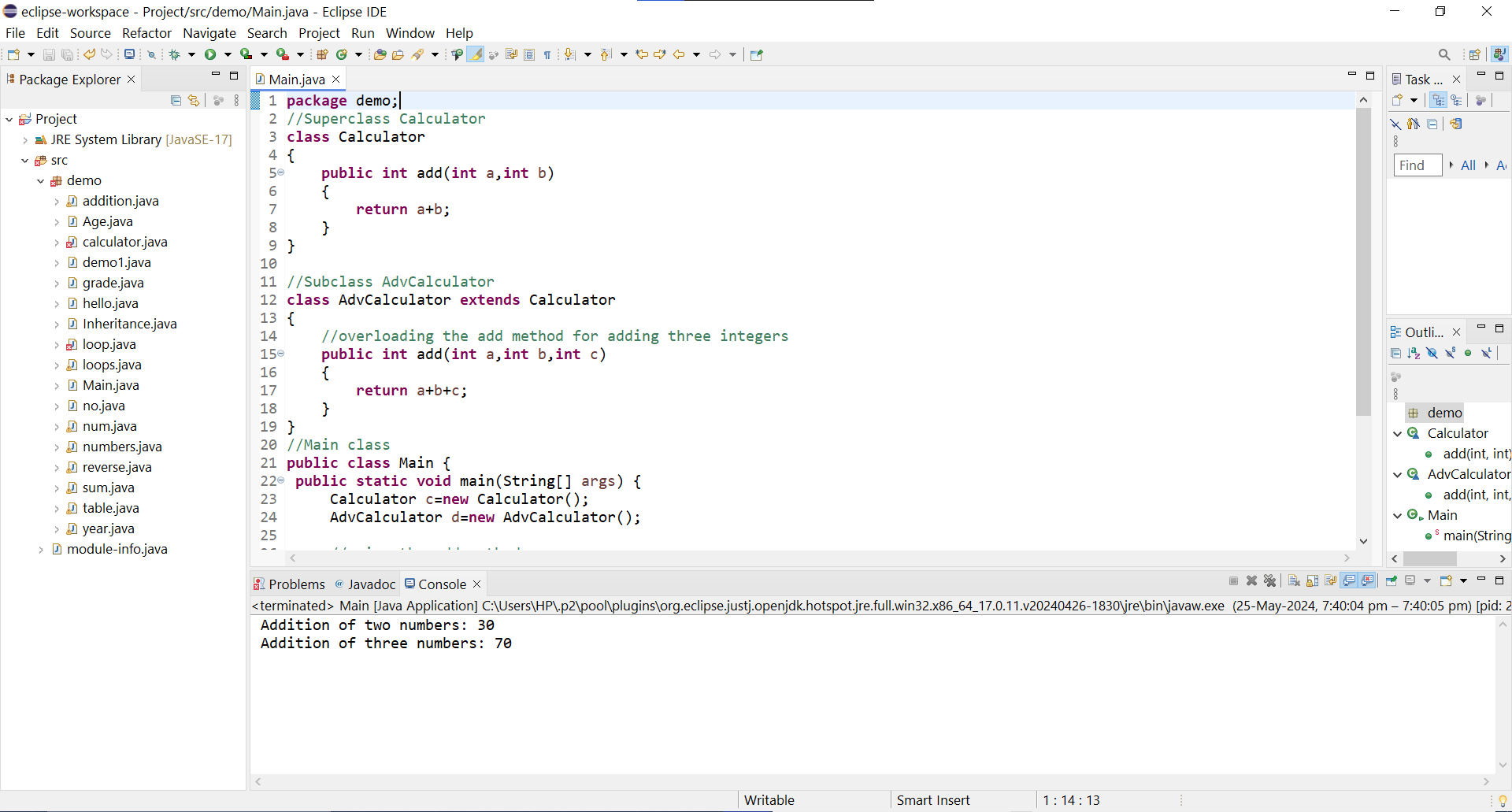
**int** s2=d.add(10, 20, 40);

System.***out***.println(" Addition of three numbers: "+ s2);

}

}

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** demo;

//Superclass Vehicle

**class** Vehicle {

// Method to move

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

{

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

}

}

//Subclass Car inheriting from Vehicle

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

{

// Additional methods and attributes specific to Car can be added here

}

//Subclass Bike inheriting from Vehicle

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

{

// Additional methods and attributes specific to Bike can be added here

}

//Main class

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

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

// Creating objects of Car and Bike

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

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

// Calling the move() method on each object

car.move();

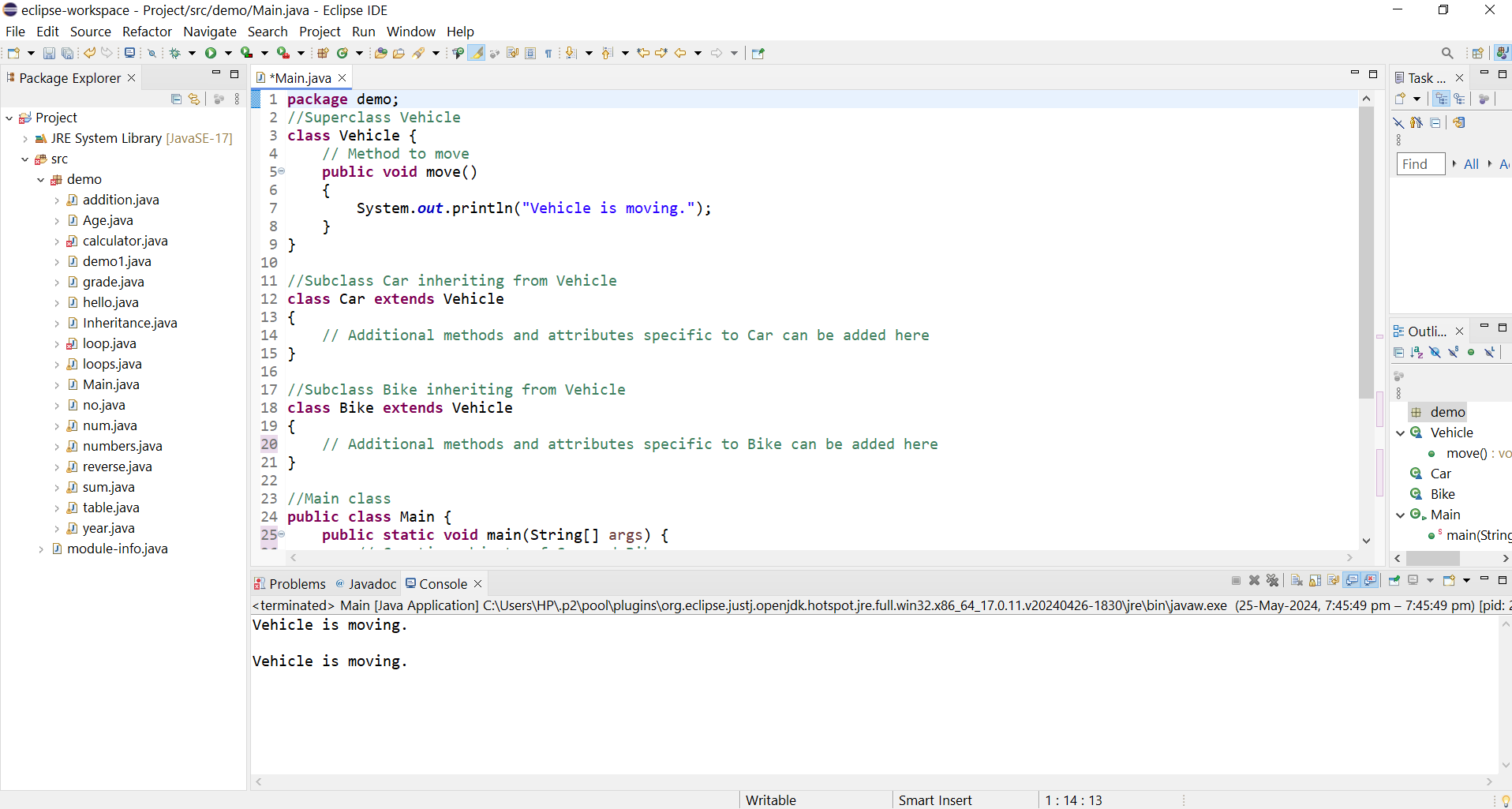
System.***out***.println(" ");

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 superclass Employee

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

// Abstract method calculatePay

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

}

//Subclass SalaryEmployee inheriting from Employee

**class** SalaryEmployee **extends** Employee {

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

// Constructor

**public** SalaryEmployee(**double** salary) {

**this**.salary = salary;

}

// Implementing calculatePay method for SalaryEmployee

@Override

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

// For simplicity, assume monthly salary

**return** salary;

}

}

//Subclass HourlyEmployee inheriting from Employee

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

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

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

// Constructor

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

**this**.hourlyRate = hourlyRate;

**this**.hoursWorked = hoursWorked;

}

// Implementing calculatePay method for HourlyEmployee

@Override

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

// For simplicity, assume no overtime calculation

**return** hourlyRate \* hoursWorked;

}

}

//Main class

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

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

// Creating objects of both subclasses

SalaryEmployee salaryEmp = **new** SalaryEmployee(10000);

HourlyEmployee hourlyEmp = **new** HourlyEmployee(10, 70);

// Calling calculatePay method on each object

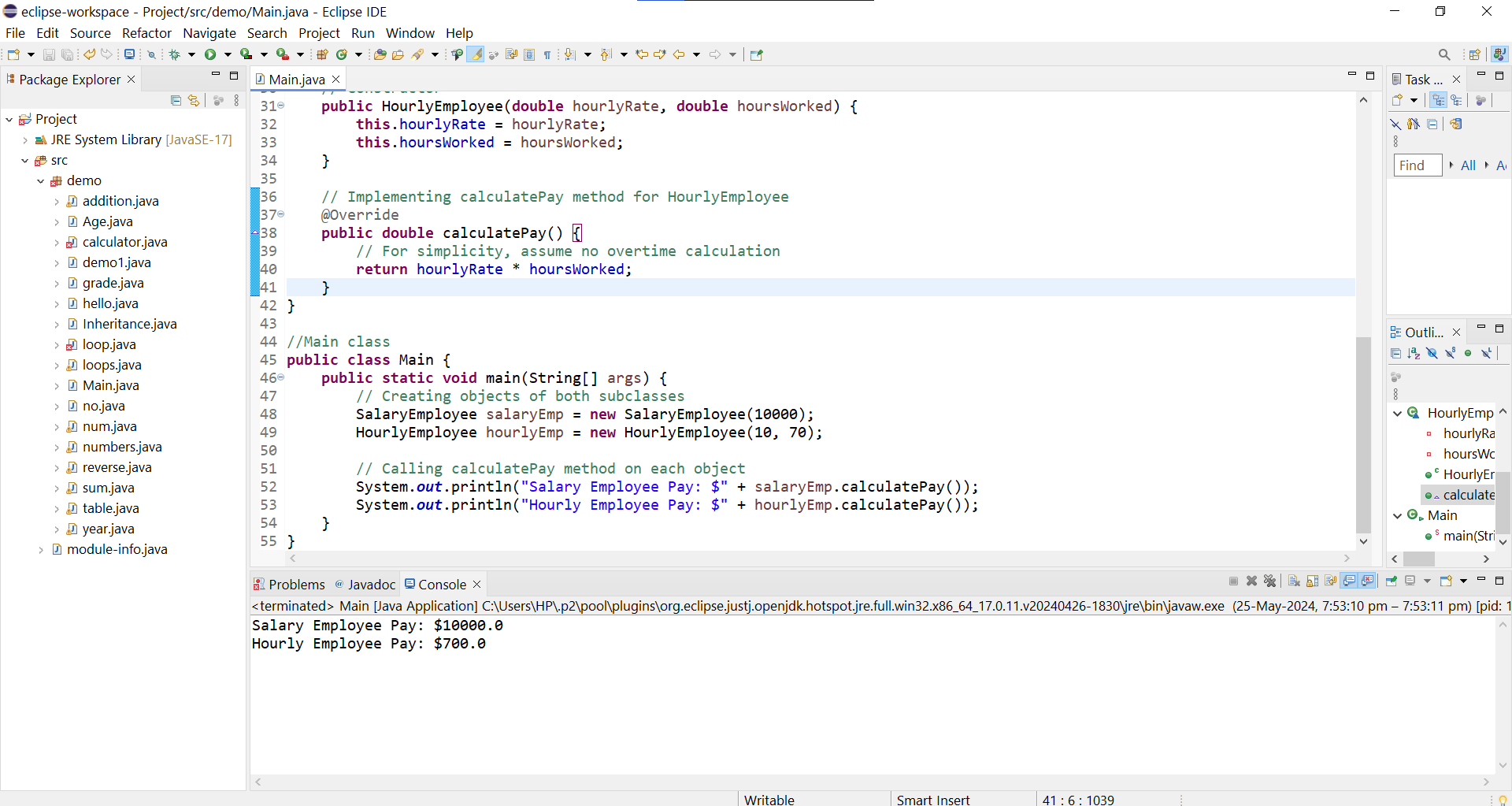
System.***out***.println("Salary Employee Pay: $" + salaryEmp.calculatePay());

System.***out***.println("Hourly Employee Pay: $" + hourlyEmp.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;

//Superclass Document

**class** Document {

// Method to open document

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

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

}

}

//Subclass WordDocument inheriting from Document

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

// Implementing open method for WordDocument

@Override

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

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

}

}

//Subclass PDFDocument inheriting from Document

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

// Implementing open method for PDFDocument

@Override

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

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

}

}

//Subclass SpreadsheetDocument inheriting from Document

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

// Implementing open method for SpreadsheetDocument

@Override

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

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

}

}

//Main class

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

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

// Demonstrating compile-time polymorphism

Document doc1 = **new** WordDocument();

Document doc2 = **new** PDFDocument();

Document doc3 = **new** SpreadsheetDocument();

// Opening different types of documents

doc1.open();

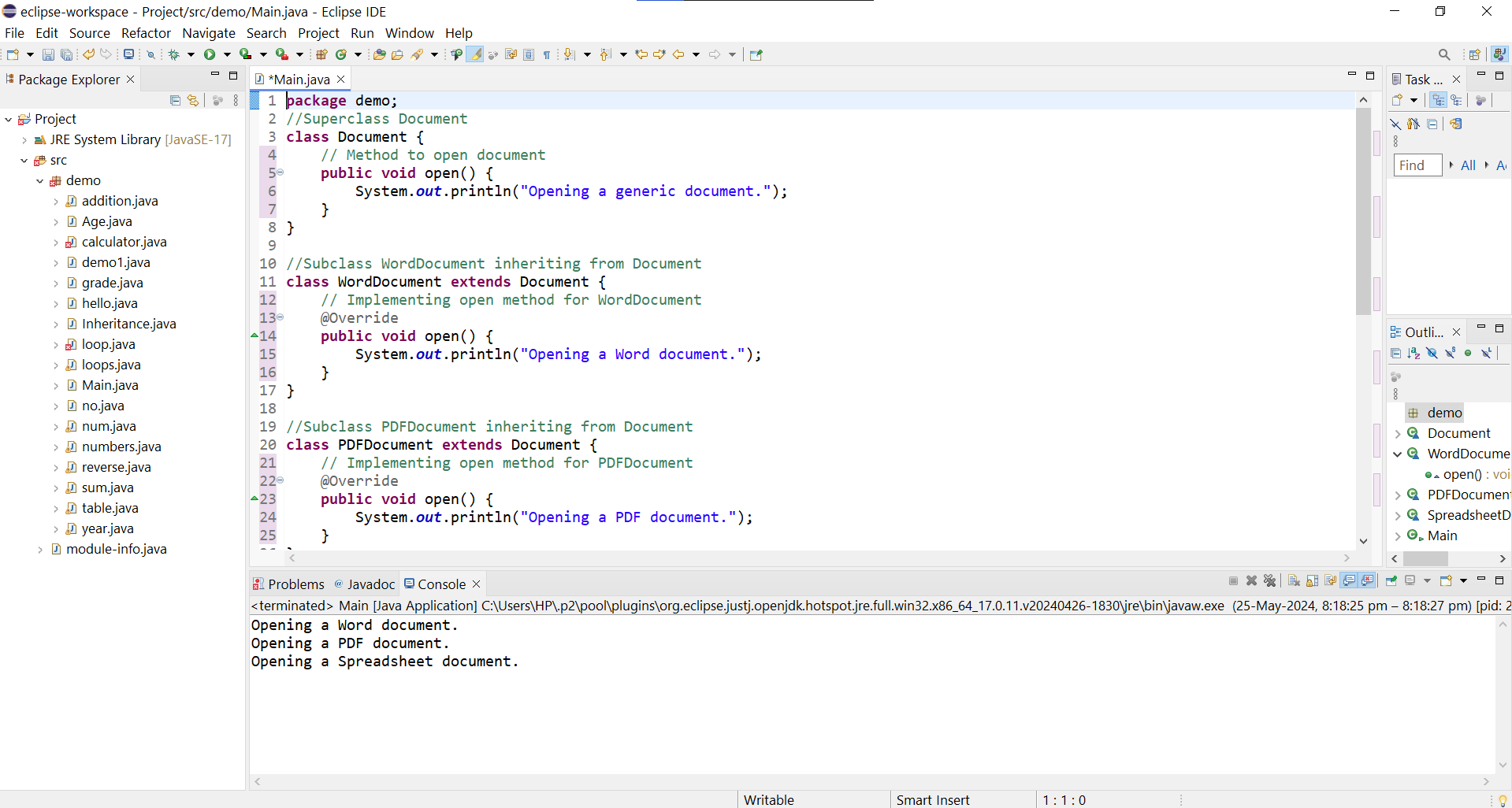
doc2.open();

doc3.open();

}

}

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;

**import** java.util.\*;

**class** Calculator {

// 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** **class** Main {

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

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter three integers: ");

**int** a = sc.nextInt();

**int** b = sc.nextInt();

**int** e = sc.nextInt();

System.***out***.println("Enter two decimal integers: ");

**double** c = sc.nextDouble();

**double** d = sc.nextDouble();

Calculator calculator = **new** Calculator();

// Using the overloaded methods

**int** sum1 = calculator.add(a, b);

**double** sum2 = calculator.add(c, d);

**int** sum3 = calculator.add(a, b, e);

// Displaying the results

System.***out***.println("Sum of integers : " + sum1);

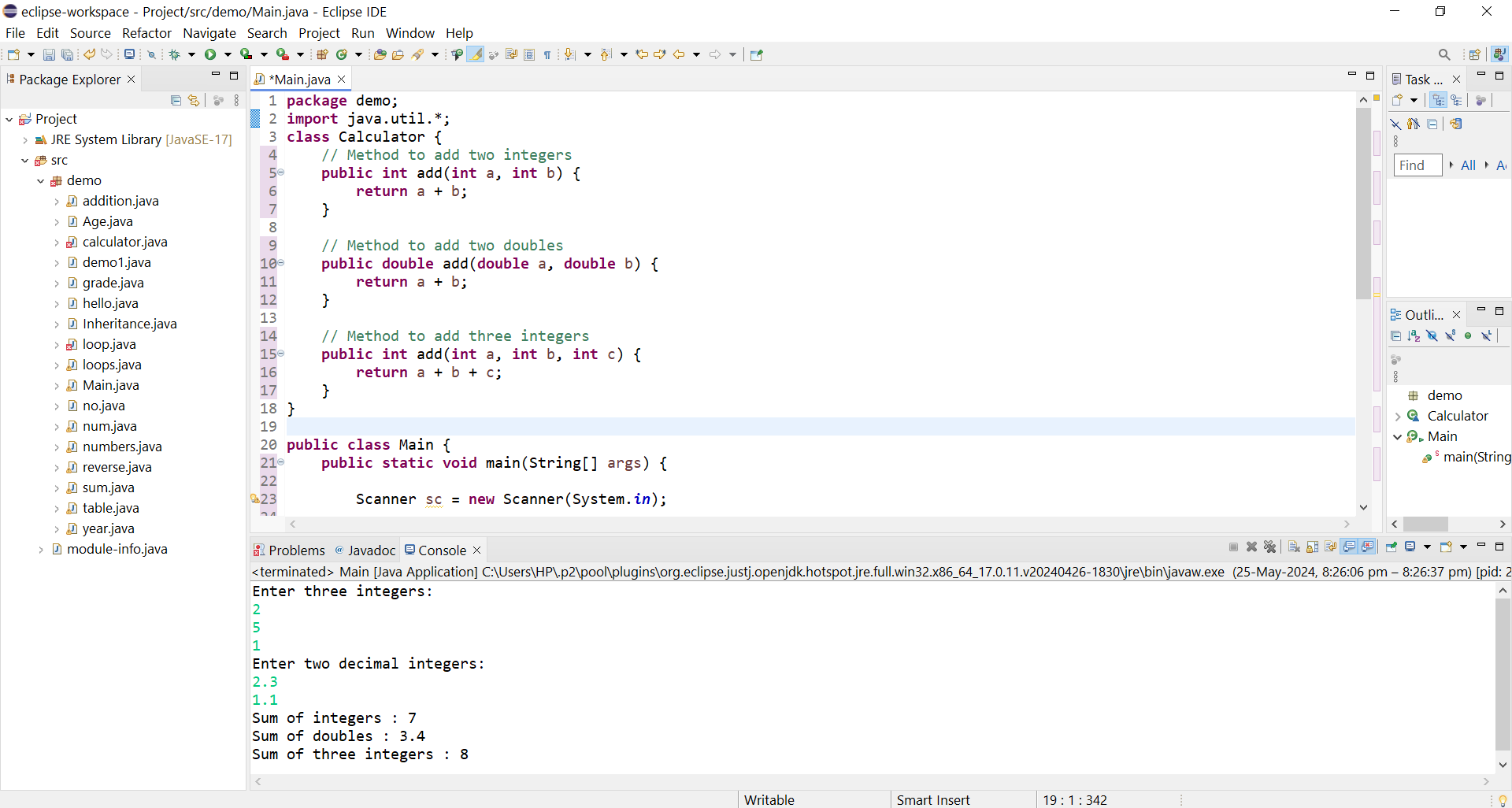
System.***out***.println("Sum of doubles : " + sum2);

System.***out***.println("Sum of three integers : " + sum3);

}

}

Output-



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.

Program-

**package** demo;

//JavaBean class Person

**public** **class** Person {

// Properties

**private** String firstName;

**private** String lastName;

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

**private** String email;

// No-argument constructor

**public** Person() {

// Default constructor

}

// 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;

}

}

//Main class to demonstrate usage

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

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

// Creating an instance of Person

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

// Setting properties

person.setFirstName("John");

person.setLastName("Doe");

person.setAge(30);

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

// Printing out properties

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

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

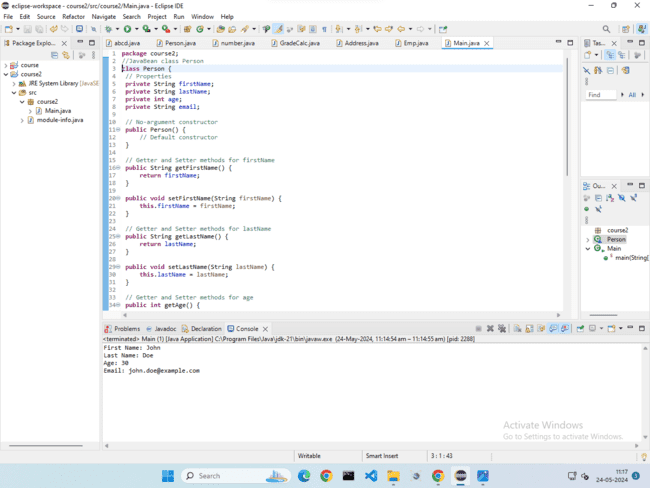
System.***out***.println("Age: " + person.getAge());

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

}

}

Output-



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** demo;

//JavaBean class Car

**public** **class** Car {

// Properties

**private** String make;

**private** String model;

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

**private** String color;

// No-argument constructor

**public** Car() {

// Default constructor

}

// 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;

}

}

//Main class to demonstrate usage

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

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

// Creating an instance of Car

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

// Setting properties

car.setMake("Toyota");

car.setModel("Camry");

car.setYear(2020);

car.setColor("Silver");

// Printing 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-

