Q-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:

**public** **class** superclass {

// Define the Person superclass

**static** **class** Person {

**protected** String name;

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

// Constructor to initialize name and age

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

**this**.name = name;

**this**.age = age;

}

// Method to display name and age

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

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

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

}

}

// Define the Student subclass

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

**private** String studentID;

// Constructor to initialize name, age, and studentID

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

**super**(name, age);

**this**.studentID = studentID;

}

// Method to display name, age, and studentID

@Override

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

**super**.display();

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

}

}

// Main class to create and display a Student object

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

// Create a Student object

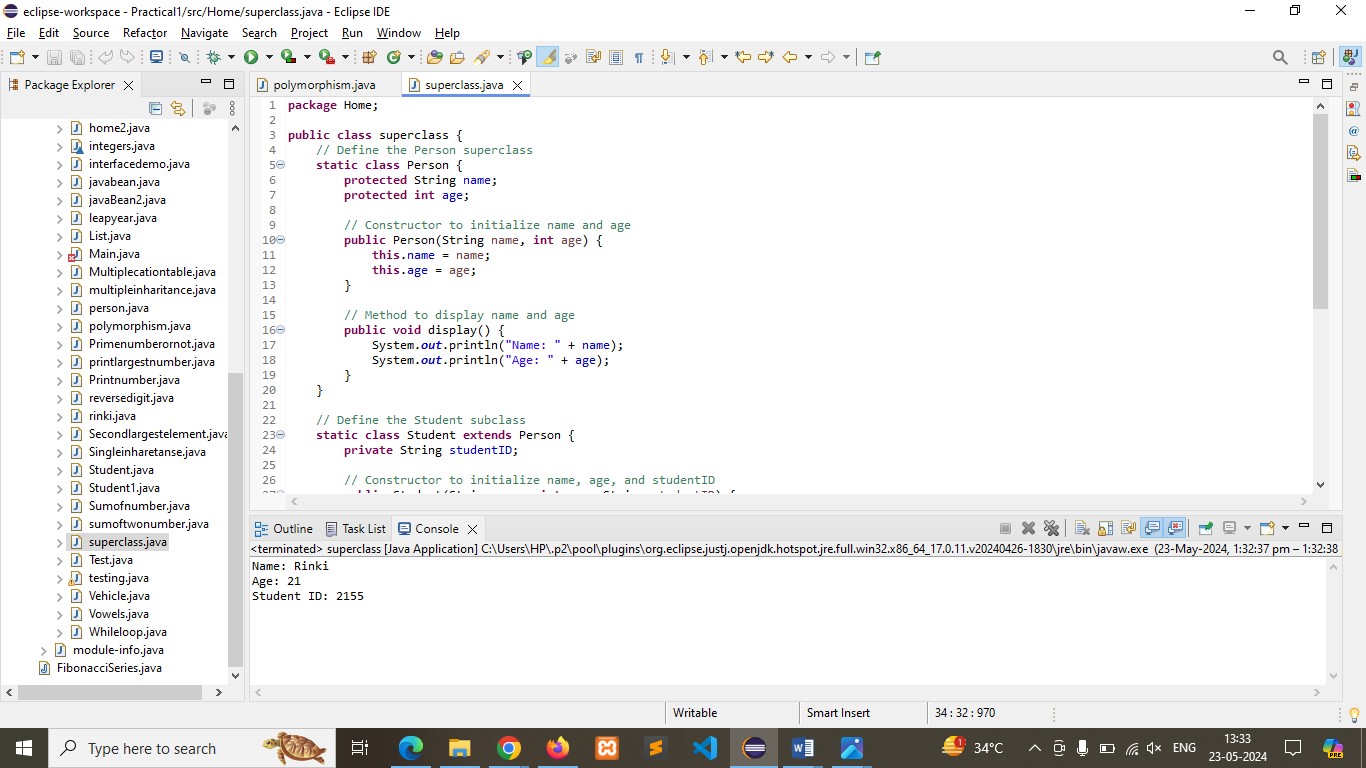
Student student = **new** Student("Rinki", 21, "2155");

// Display the attributes of the Student object

student.display();

}

}

Output

Q-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:

// Define the Calculator superclass

**class** Calculator {

// Method to add two integers

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

**return** a + b;

}

}

// Define the AdvancedCalculator subclass

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

// Overloaded 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) {

// **TODO** Auto-generated method stub

// Create an instance of AdvancedCalculator

AdvancedCalculator calc = **new** AdvancedCalculator();

// Use the add method to add two integers

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

System.***out***.println("Result of adding two integers: " + result1);

// Use the overloaded add method to add three integers

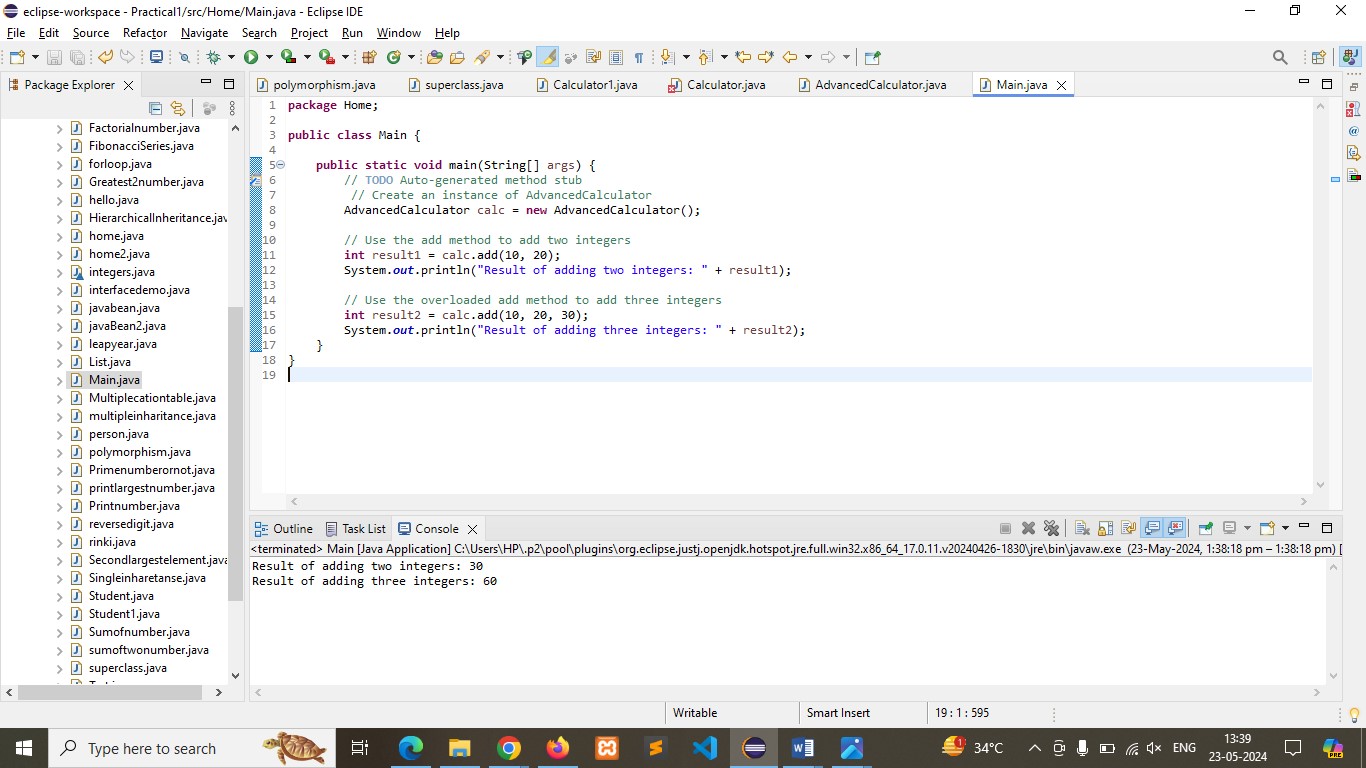
**int** result2 = calc.add(10, 20, 30);

System.***out***.println("Result of adding three integers: " + result2);

}

}

Output:



Q-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:

**public** **class** Vehicle {

// Method to move

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

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

}

}

// Define subclass Car inheriting from Vehicle

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

// No additional methods or attributes

}

// Define subclass Bike inheriting from Vehicle

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

// No additional methods or attributes

}

// Main class to demonstrate the program

**public** **class** Main

{

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

// Create objects of Car and Bike

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

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

// Call the move() method on each object

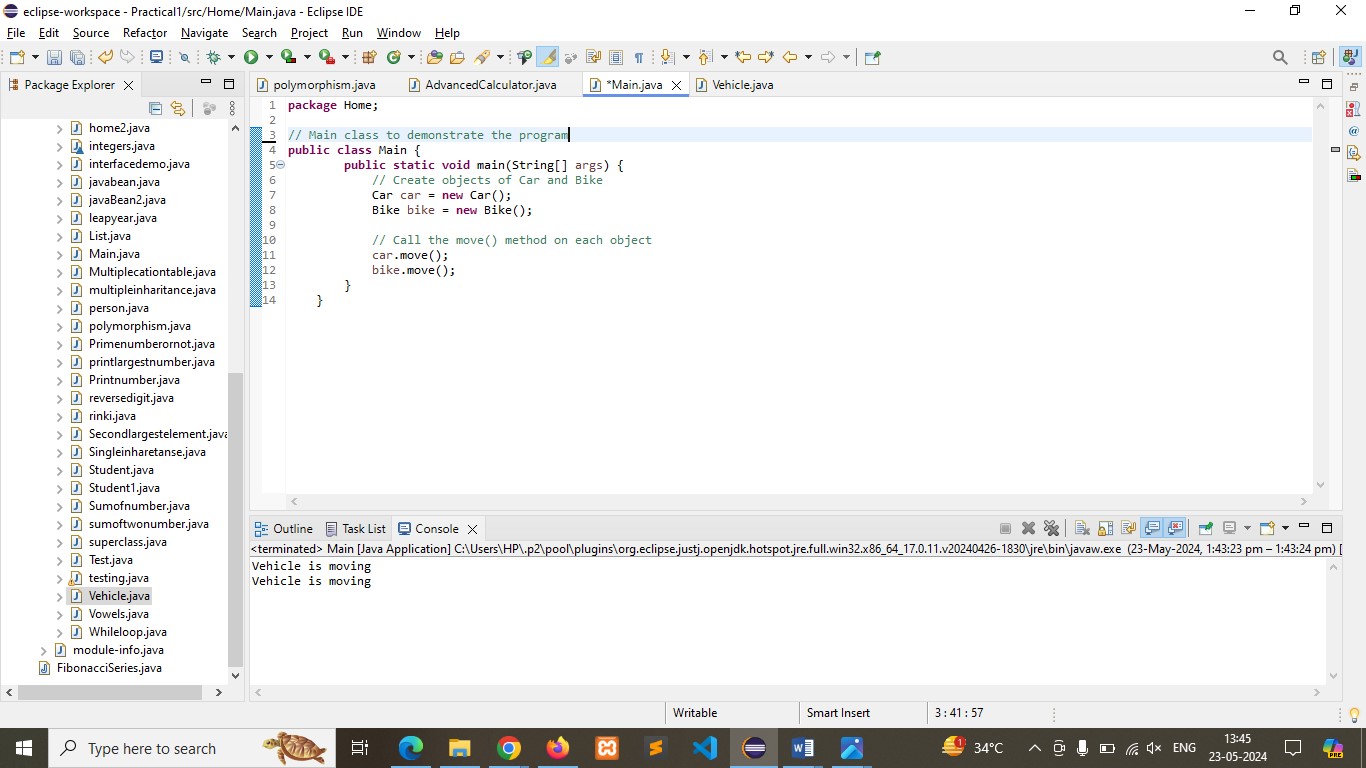
car.move();

bike.move();

}

}

Output:



Q-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:

//Define abstract class Employee

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

String name;

// Constructor

**public** Employee(String name) {

**this**.name = name;

}

// Abstract method to calculate pay

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

}

// Define subclass SalariedEmployee inheriting from Employee

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

**double** annualSalary;

// Constructor

**public** SalariedEmployee(String name, **double** annualSalary) {

**super**(name);

**this**.annualSalary = annualSalary;

}

// Implement the calculatePay method

@Override

**double** calculatePay() {

**return** annualSalary / 12; // Assuming monthly pay

}

}

// Define subclass HourlyEmployee inheriting from Employee

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

**double** hourlyRate;

**int** hoursWorked;

// Constructor

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

**super**(name);

**this**.hourlyRate = hourlyRate;

**this**.hoursWorked = hoursWorked;

}

// Implement the calculatePay method

@Override

**double** calculatePay() {

**return** hourlyRate \* hoursWorked; // Pay based on hours worked

}

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

// **TODO** Auto-generated method stub

}

}

// Main class to demonstrate the program

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

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

// Create an object of SalariedEmployee

SalariedEmployee salariedEmployee = **new** SalariedEmployee("rinki", 60000);

// Create an object of HourlyEmployee

HourlyEmployee hourlyEmployee = **new** HourlyEmployee("Montu", 20, 160);

// Call the calculatePay() method on each object and display the result

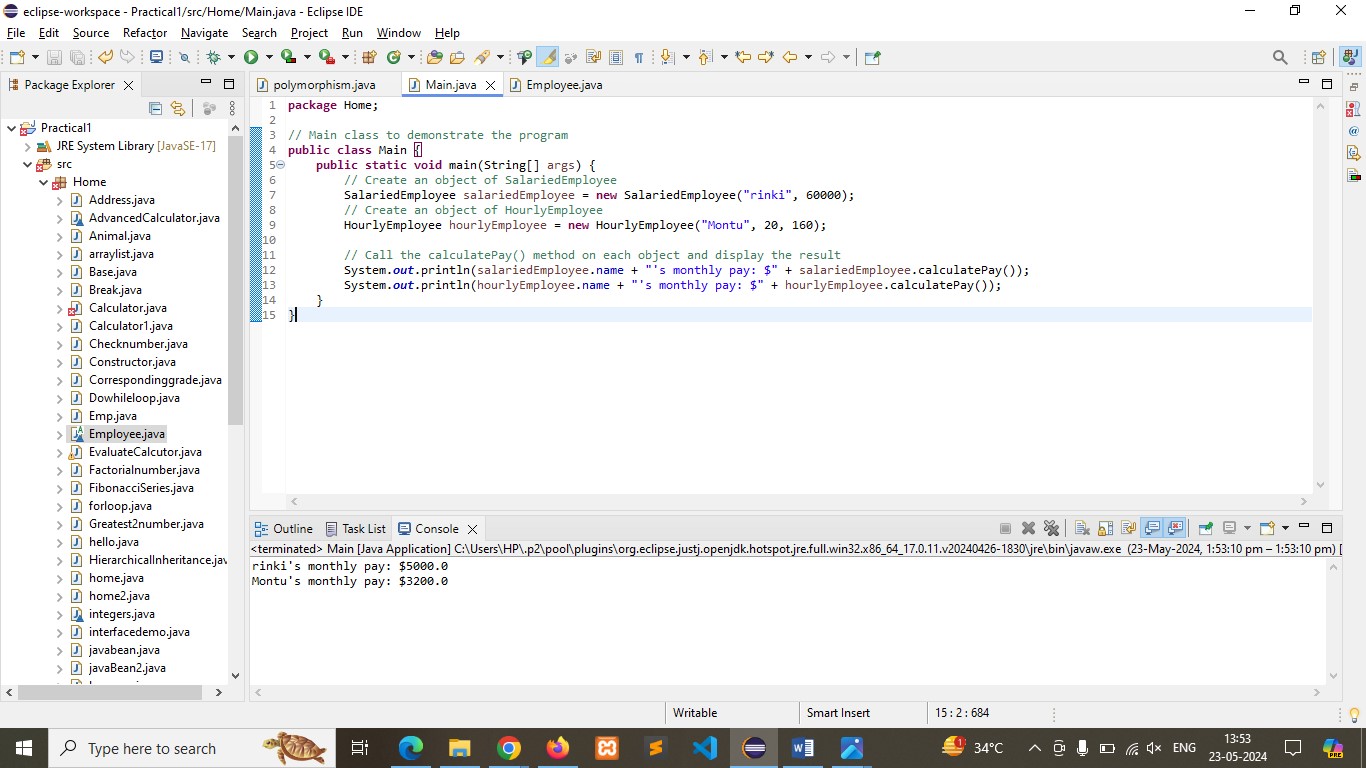
System.***out***.println(salariedEmployee.name + "'s monthly pay: $" + salariedEmployee.calculatePay());

System.***out***.println(hourlyEmployee.name + "'s monthly pay: $" + hourlyEmployee.calculatePay());

}

}

Output:



Q-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:

//Define abstract class Document

**abstract** **class** Document {

// Abstract method to open the document

**abstract** **void** open();

}

//Define subclass WordDocument inheriting from Document

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

// Implement the open method

@Override

**void** open() {

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

}

}

//Define subclass PDFDocument inheriting from Document

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

// Implement the open method

@Override

**void** open() {

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

}

}

//Define subclass SpreadsheetDocument inheriting from Document

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

// Implement the open method

@Override

**void** open() {

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

}

}

// Main class to demonstrate the program

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

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

// Create an array of Document type to store different document objects

Document[] documents = **new** Document[3];

// Instantiate each document type and store in the array

documents[0] = **new** WordDocument();

documents[1] = **new** PDFDocument();

documents[2] = **new** SpreadsheetDocument();

// Use a loop to open each document

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

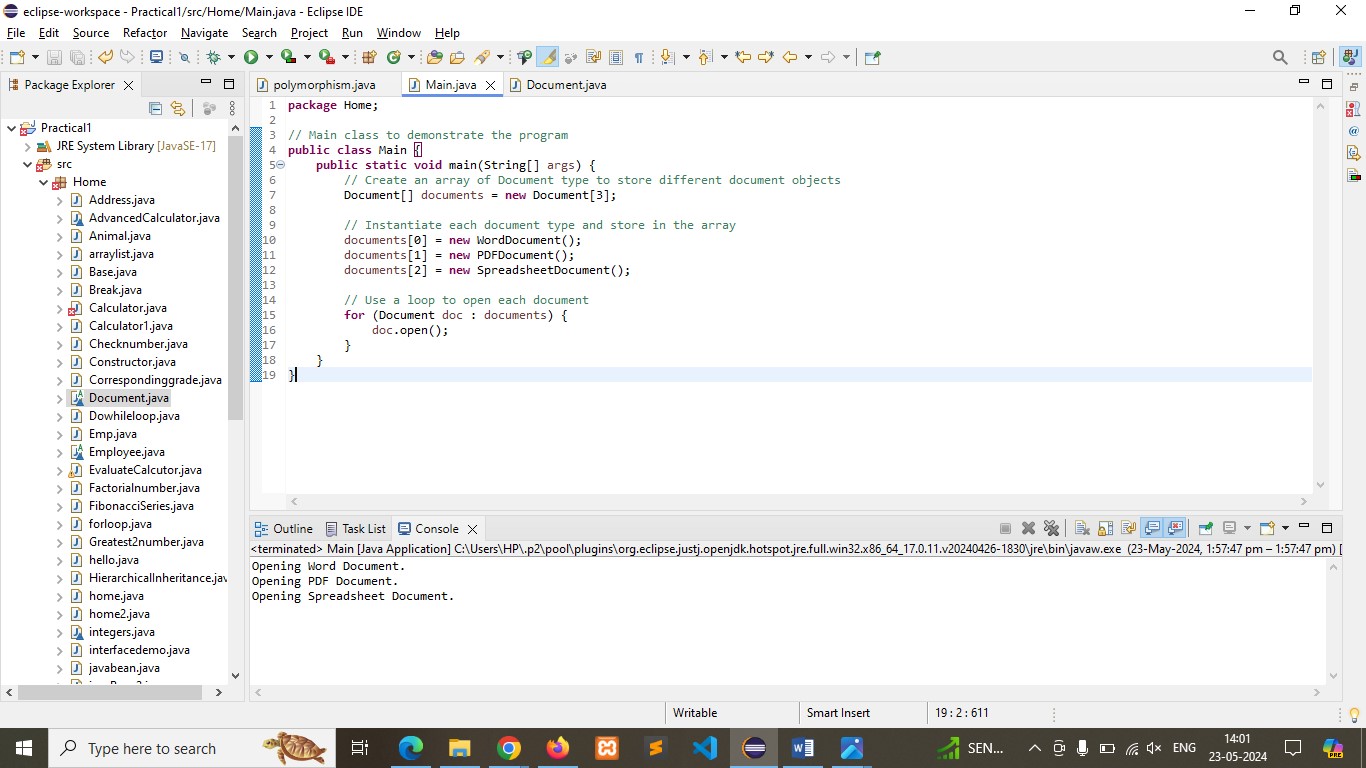
doc.open();

}

}

}

Output:



Q-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:

**public** **class** Cal {

// 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) {

// **TODO** Auto-generated method stub

}

}

//Main class to demonstrate the usage of Cal methods

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

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

// Create an object of Cal

Cal cal = **new** Cal();

// Demonstrate add method with two integers

**int** sum1 = cal.add(5, 10);

System.***out***.println("Sum of 5 and 10: " + sum1);

// Demonstrate add method with two doubles

**double** sum2 = cal.add(5.5, 10.5);

System.***out***.println("Sum of 5.5 and 10.5: " + sum2);

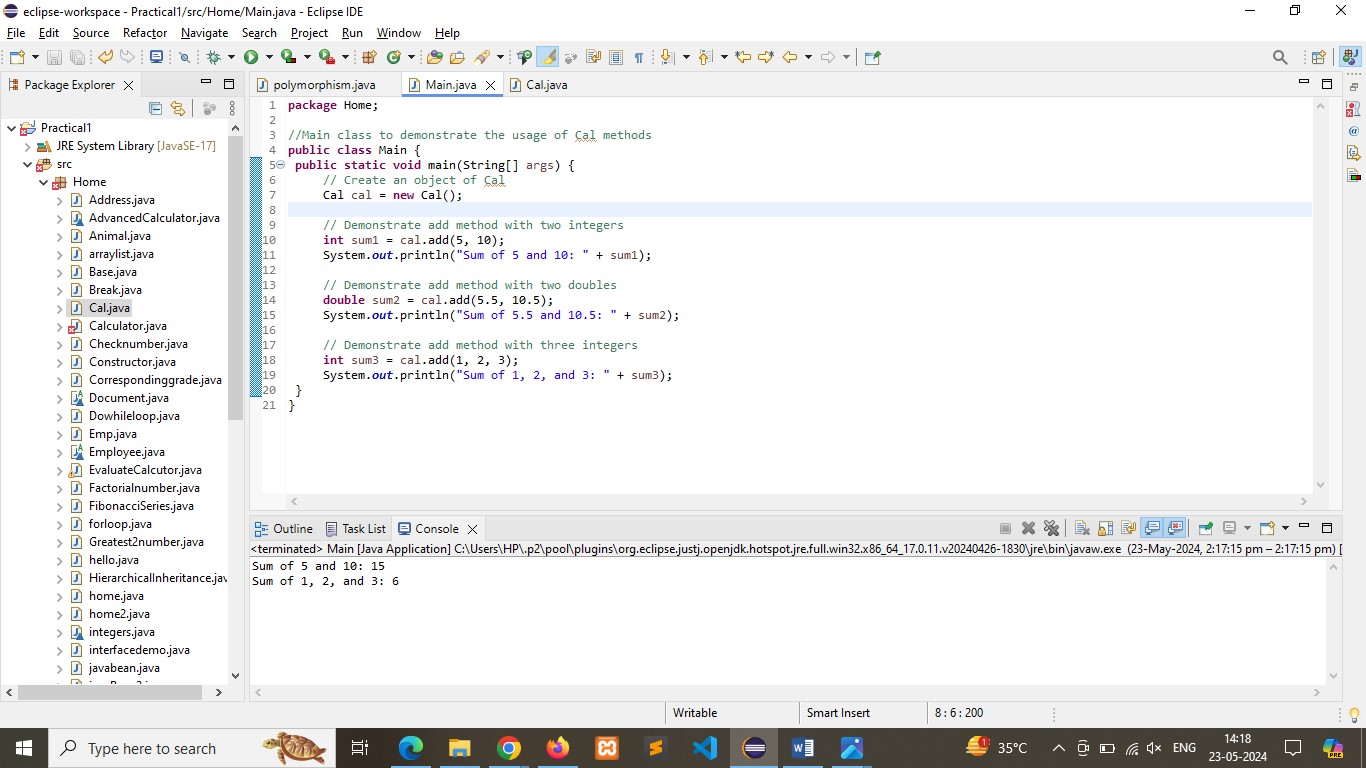
// Demonstrate add method with three integers

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

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

}

}

Output: 

Q-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:

**import** java.io.Serializable;

// Define JavaBean class Person

**public** **class** person **implements** Serializable {

**private** String firstName;

**private** String lastName;

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

**private** String email;

// No-argument constructor

**public** person() {}

// Getter method for firstName

**public** String getFirstName() {

**return** firstName;

}

// Setter method for firstName

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

**this**.firstName = firstName;

}

// Getter method for lastName

**public** String getLastName() {

**return** lastName;

}

// Setter method for lastName

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

**this**.lastName = lastName;

}

// Getter method for age

**public** **int** getAge() {

**return** age;

}

// Setter method for age

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

**this**.age = age;

}

// Getter method for email

**public** String getEmail() {

**return** email;

}

// Setter method for email

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

**this**.email = email;

}

}

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

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

// Create an instance of Person

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

// Set properties using setter methods

person.setFirstName("Rinki");

person.setLastName("Jaiswar");

person.setAge(23);

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

// Get and print properties using getter methods

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

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

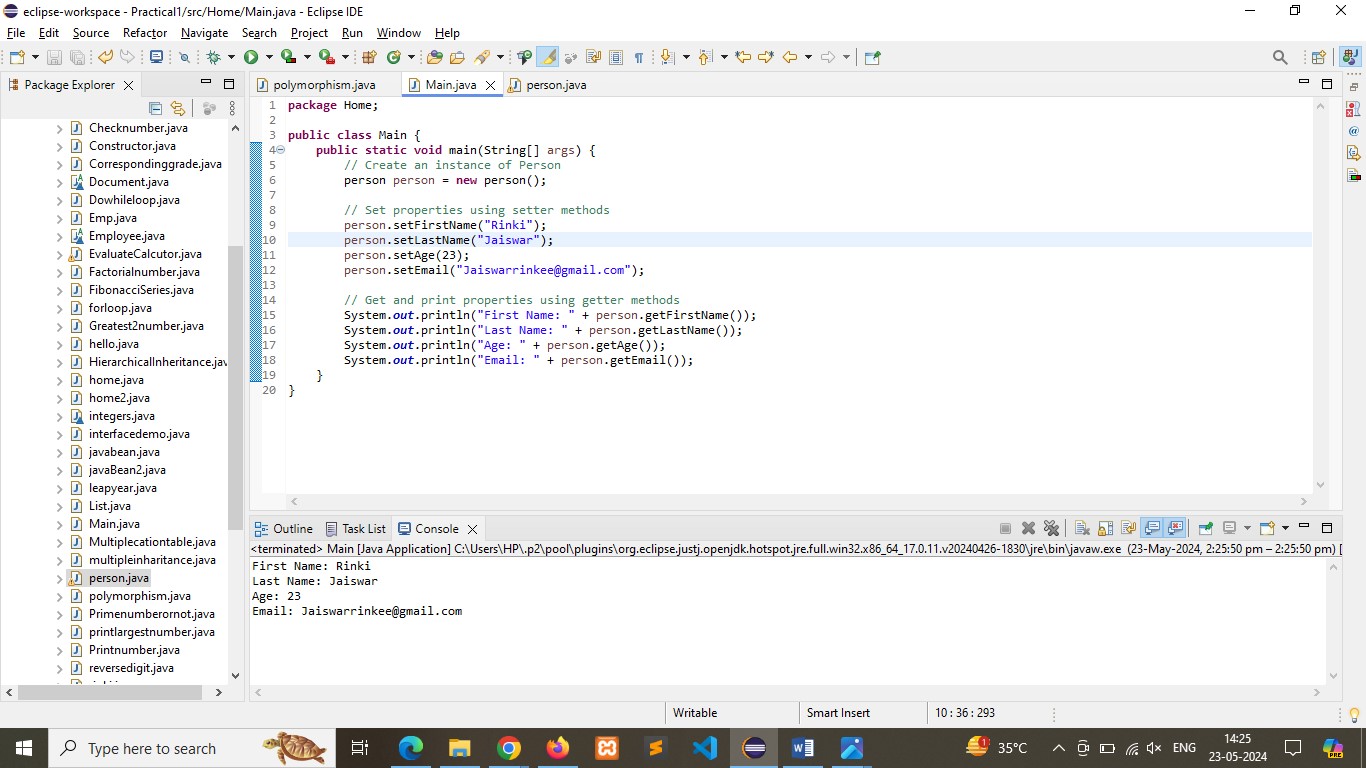
System.***out***.println("Age: " + person.getAge());

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

}

}

Output:



Q-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:

**import** java.io.Serializable;

//Define JavaBean class Car

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

**private** String make;

**private** String model;

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

**private** String color;

// No-argument constructor

**public** Car1() {}

// Getter method for make

**public** String getMake() {

**return** make;

}

// Setter method for make

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

**this**.make = make;

}

// Getter method for model

**public** String getModel() {

**return** model;

}

// Setter method for model

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

**this**.model = model;

}

// Getter method for year

**public** **int** getYear() {

**return** year;

}

// Setter method for year

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

**this**.year = year;

}

// Getter method for color

**public** String getColor() {

**return** color;

}

// Setter method for color

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

**this**.color = color;

}

}

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

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

// Create an instance of Car

Car1 car = **new** Car1();

// Set properties using setter methods

car.setMake("Toyota");

car.setModel("Camry");

car.setYear(2020);

car.setColor("Blue");

// Get and print properties using getter methods

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:

