Q-1.)     Method Overloading: Write a class Calculator with overloaded methods add(). Implement add() methods that take:

     - Two integers

     - Two double values

     - Three integers

     - A variable number of integers

**Program:**

**package** Home;

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

// 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;

}

// Method to add a variable number of integers

**public** **int** add(**int**... numbers) {

**int** sum = 0;

**for** (**int** num : numbers) {

sum += num;

}

**return** sum;

}

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

Cal calc = **new** Cal();

// Testing add method with two integers

**int** sum1 = calc.add(2, 3);

System.***out***.println("Sum of two integers (2,3):"+sum1); // Output: 5

// Testing add method with two double values

**double** sum2 = calc.add(2.5, 3.5);

System.***out***.println("Sum of two doubles(2.5, 3.5): " + sum2);// Output: 6.0

// Testing add method with three integers

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

System.***out***.println("Sum of three integers(1, 2, 3): " + sum3);// Output: 6

// Testing add method with a variable number of integers

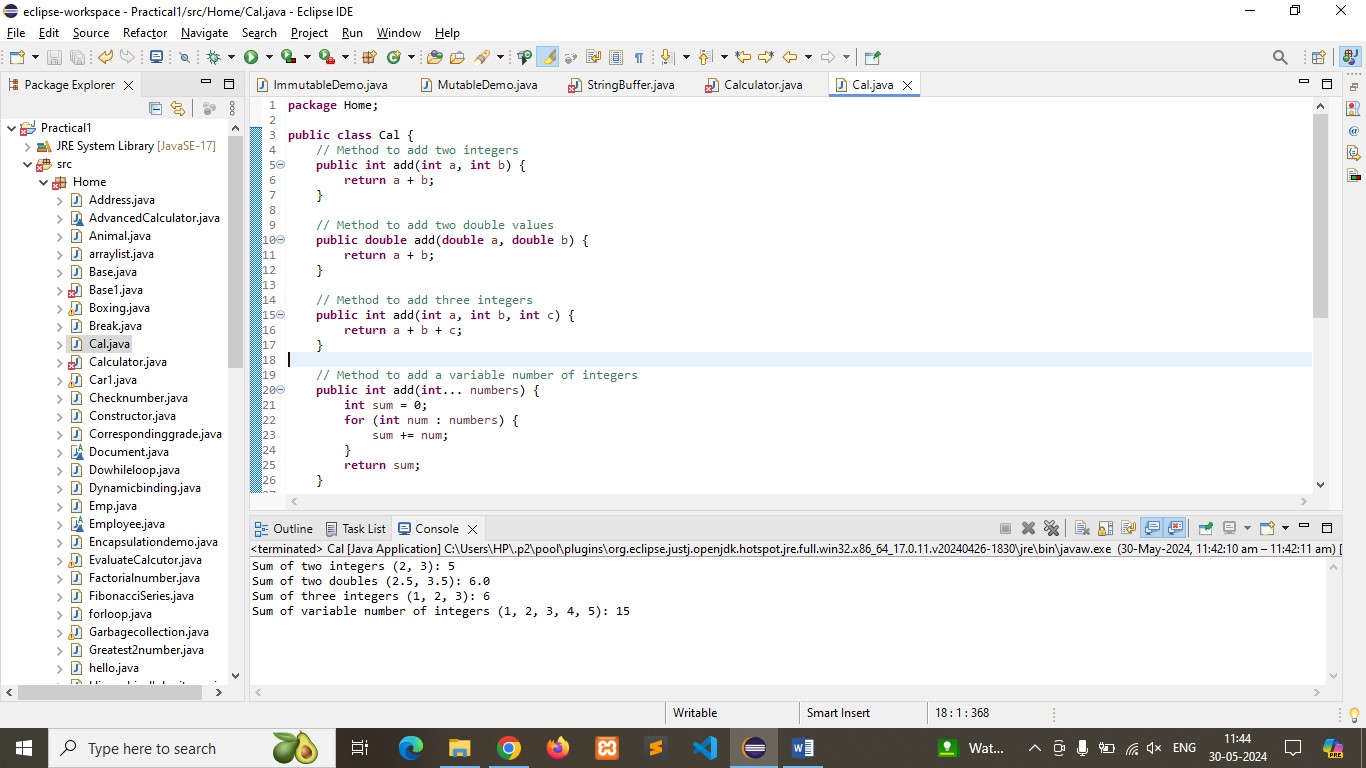
**int** sum4 = calc.add(1, 2, 3, 4, 5);

System.***out***.println("Sum of variable number of integers (1, 2, 3, 4, 5): " + sum4); // Output: 15

}

}

**OutPut:**



Q-2). Super Keyword: Create a class Person with a constructor that accepts and sets name and age.

   - Create a subclass Student that adds a grade property and initializes name and age using the super keyword in its constructor.

   - Demonstrate the creation of Student objects and the usage of super to call the parent class constructor.

Pogram:

**public** **class** person {

**protected** String name;

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

// Constructor for Person class

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

**this**.name = name;

**this**.age = age;

}

// Method to display person details

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

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

}

}

**public** **class** Student **extends** person {

**private** String grade;

// Constructor for Student class

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

// Call the constructor of the superclass (Person)

**super**(name, age);

**this**.grade = grade;

}

// Method to display student details

@Override

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

// Call the display method of the superclass (Person)

**super**.display();

System.***out***.println("Grade: " + grade);

}

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

// Create Student objects

Student student1 = **new** Student("Rinki", 21, "A");

Student student2 = **new** Student("Montu", 22, "B");

// Display the details of the students

System.***out***.println("Student 1 Details:");

student1.display();// Output: Name: Rinki, Age: 21, Grade: A

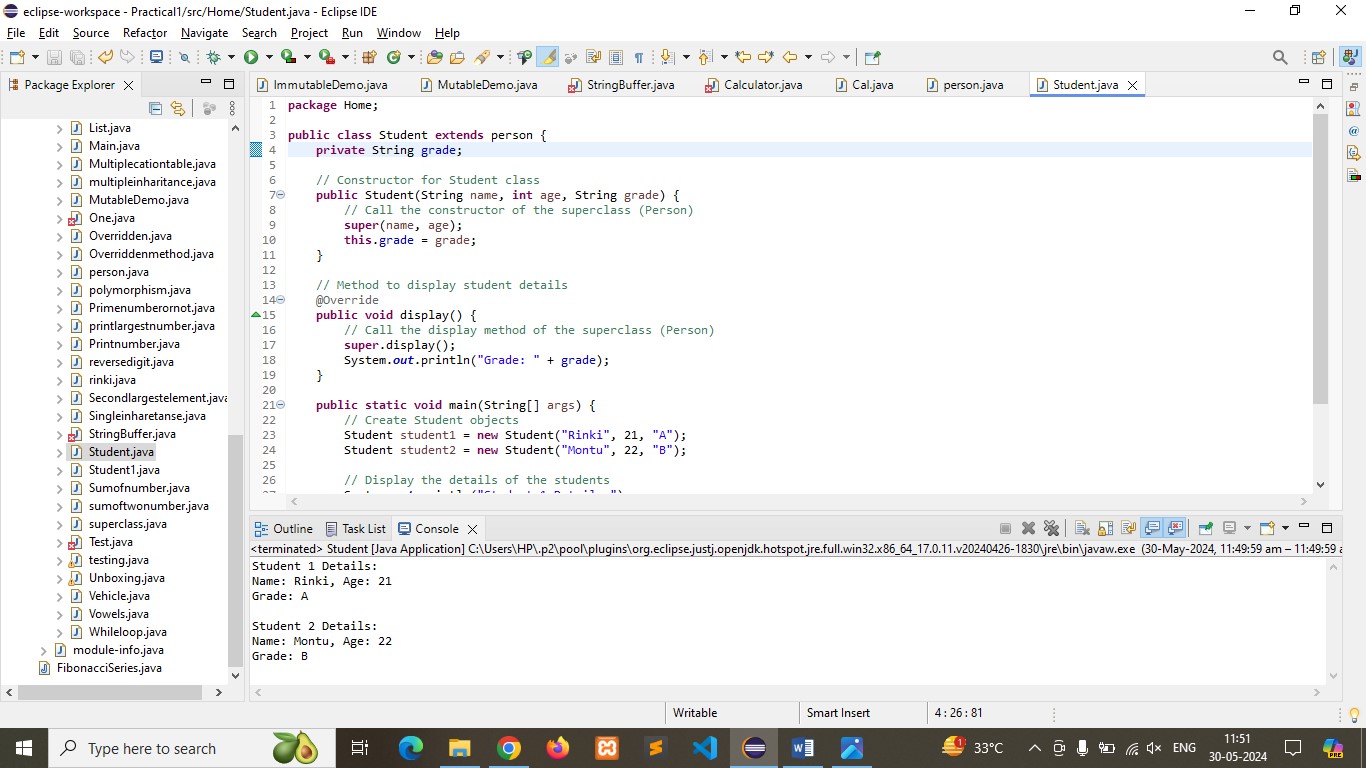
System.***out***.println("\nStudent 2 Details:");

student2.display(); // Output: Name:Montu,Age: 22, Grade: B

}

}

Output:



Q-3). Super Keyword: Create a base class Shape with a method draw() that prints "Drawing Shape".

   - Create a subclass Circle that overrides draw() to print "Drawing Circle".

   - Inside the draw() method of Circle, call the draw() method of the Shape class using super.draw().

   - Write a main method to demonstrate calling draw() on a Circle object.

Program:

**public** **class** Shape {

// Method to draw a shape

**public** **void** draw() {

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

}

}

**public** **class** Circle **extends** Shape {

// Method to draw a circle, overrides the draw method in Shape

@Override

**public** **void** draw() {

// Call the draw method of the superclass (Shape)

**super**.draw();

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

}

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

// Create a Circle object

Circle circle = **new** Circle();

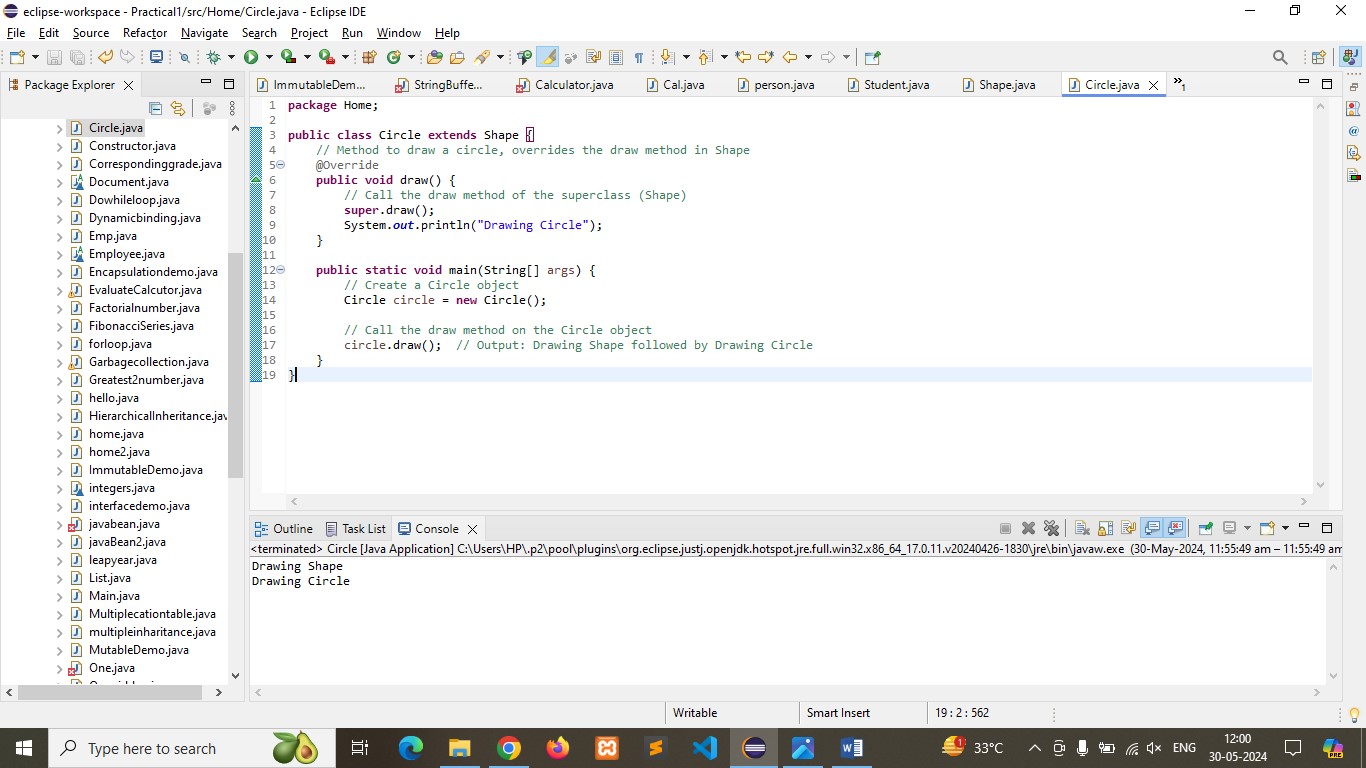
// Call the draw method on the Circle object

circle.draw(); // Output: Drawing Shape followed by Drawing Circle

}

}

Ouput:



Q-4)Create a base class BankAccount with a method deposit(amount) and a constructor that sets the initial balance.

   - Create a subclass SavingsAccount that overrides deposit(amount) to add interest before depositing. Use the super keyword to call the deposit method of the base class.

   - Write a main method to demonstrate creating a SavingsAccount and depositing an amount to see the effect of interest.

Program:

**public** **class** BankAccount {

**protected** **double** balance;

// Constructor to set the initial balance

**public** BankAccount(**double** initialBalance) {

**this**.balance = initialBalance;

}

// Method to deposit an amount into the account

**public** **void** deposit(**double** amount) {

balance += amount;

System.***out***.println("Deposited " + amount + ", New Balance: " + balance);

}

// Method to display the current balance

**public** **void** displayBalance() {

System.***out***.println("Current Balance: " + balance);

}

}

**public** **class** SavingsAccount **extends** BankAccount {

**private** **double** interestRate;

// Constructor to set the initial balance and interest rate

**public** SavingsAccount(**double** initialBalance, **double** interestRate) {

**super**(initialBalance); // Call the constructor of the superclass (BankAccount)

**this**.interestRate = interestRate;

}

// Method to deposit an amount with interest into the account

@Override

**public** **void** deposit(**double** amount) {

// Calculate interest

**double** interest = amount \* interestRate / 100;

**double** totalAmount = amount + interest;

// Call the deposit method of the superclass (BankAccount)

**super**.deposit(totalAmount);

System.***out***.println("Interest added: " + interest);

}

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

// Create a SavingsAccount object

SavingsAccount savingsAccount = **new** SavingsAccount(1000, 5);

// Display the initial balance

savingsAccount.displayBalance(); //Output: Current Balance: 1000.0

// Deposit an amount and see the effect of interest

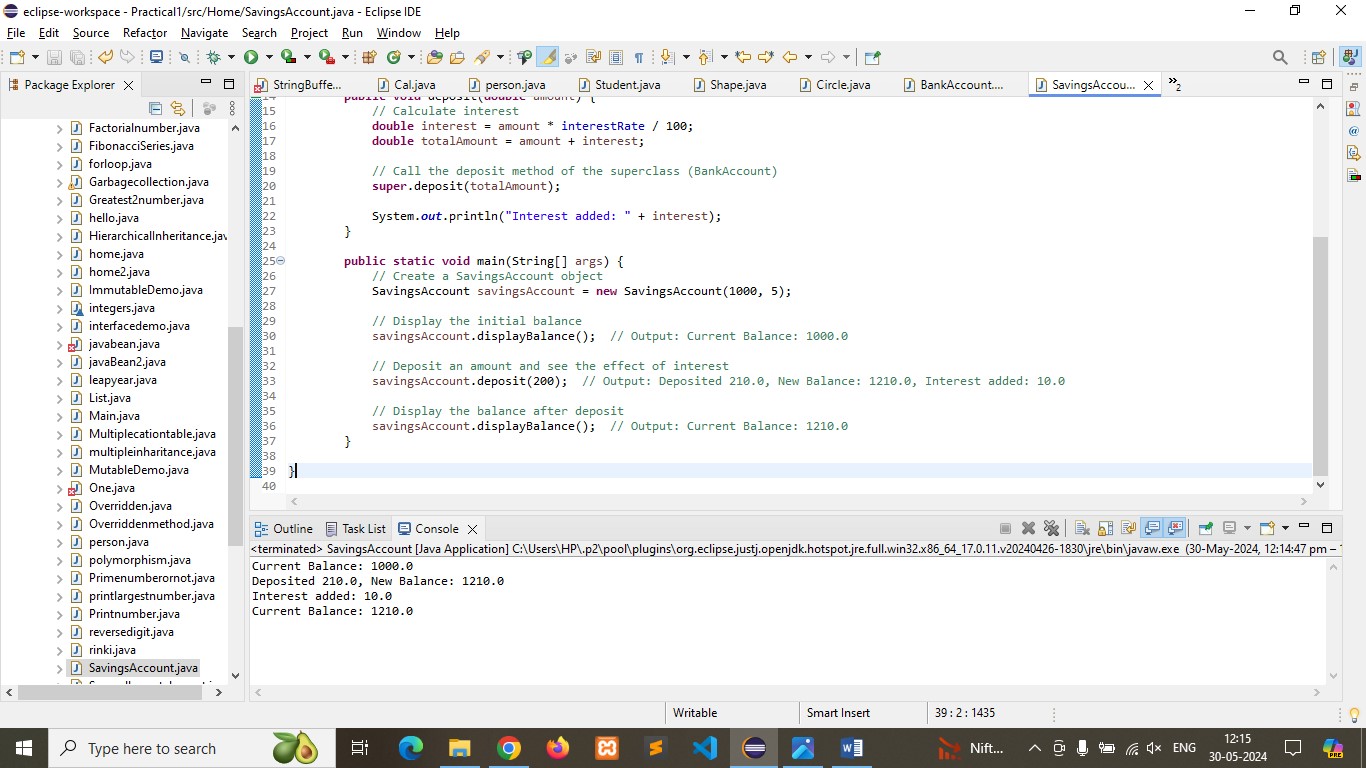
savingsAccount.deposit(200); // Output: Deposited 210.0, New Balance: 1210.0, Interest added: 10.0

// Display the balance after deposit

savingsAccount.displayBalance();// Output: Current Balance: 1210.0

}

}

Output:

Q-5) Define a class Employee with properties name and salary and a method displayDetails().

   - Create a subclass Manager that adds a property department and overrides displayDetails() to include department details. Use the super keyword to call the displayDetails() method of Employee within Manager.

   - In the main method, create objects of Employee and Manager and call displayDetails() to show the details.

Program:

//Define the Employee class

**class** Employee {

// Properties

**protected** String name;

**protected** **double** salary;

// Constructor

**public** Employee(String name, **double** salary) {

**this**.name = name;

**this**.salary = salary;

}

// Method to display details

**public** **void** displayDetails() {

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

System.***out***.println("Salary: " + salary);

}

}

//Define the Manager class as a subclass of Employee

**class** Manager **extends** Employee {

// Additional property

**private** String department;

// Constructor

**public** Manager(String name, **double** salary, String department) {

// Call the constructor of the superclass (Employee)

**super**(name, salary);

**this**.department = department;

}

// Override displayDetails method to include department details

@Override

**public** **void** displayDetails() {

// Call displayDetails method of the superclass (Employee)

**super**.displayDetails();

System.***out***.println("Department: " + department);

}

}

//Main class to demonstrate Employee and Manager classes

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

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

// Create an Employee object

Employee employee = **new** Employee("Rinki", 50000);

// Display details of the employee

System.***out***.println("Employee Details:");

employee.displayDetails();

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

// Create a Manager object

Manager manager = **new** Manager("Riya", 80000, "Engineering");

// Display details of the manager

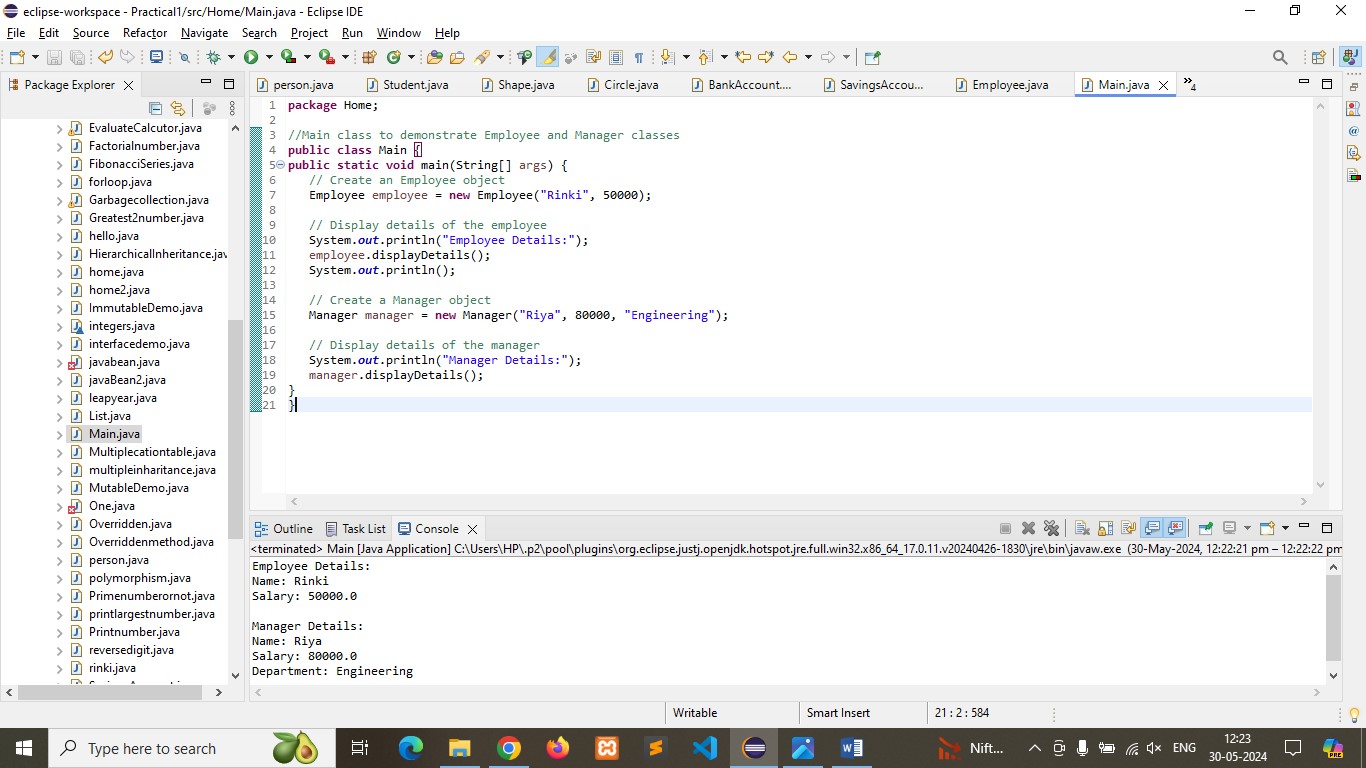
System.***out***.println("Manager Details:");

manager.displayDetails();

}

}

Output:



Q-6) Write the same programme for the class ImmutableExample, to achieve object value ‘Hi’.

Program:

// Define the ImmutableExample class

**public** **class** ImmutableExample {

// Private final instance variable to hold the value "Hi"

**private** **final** String value;

// Constructor to initialize the value to "Hi"

**public** ImmutableExample() {

**this**.value = "Hi";

}

// Getter method to access the value

**public** String getValue() {

**return** value;

}

}

//Main class to demonstrate ImmutableExample

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

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

// Create an instance of ImmutableExample

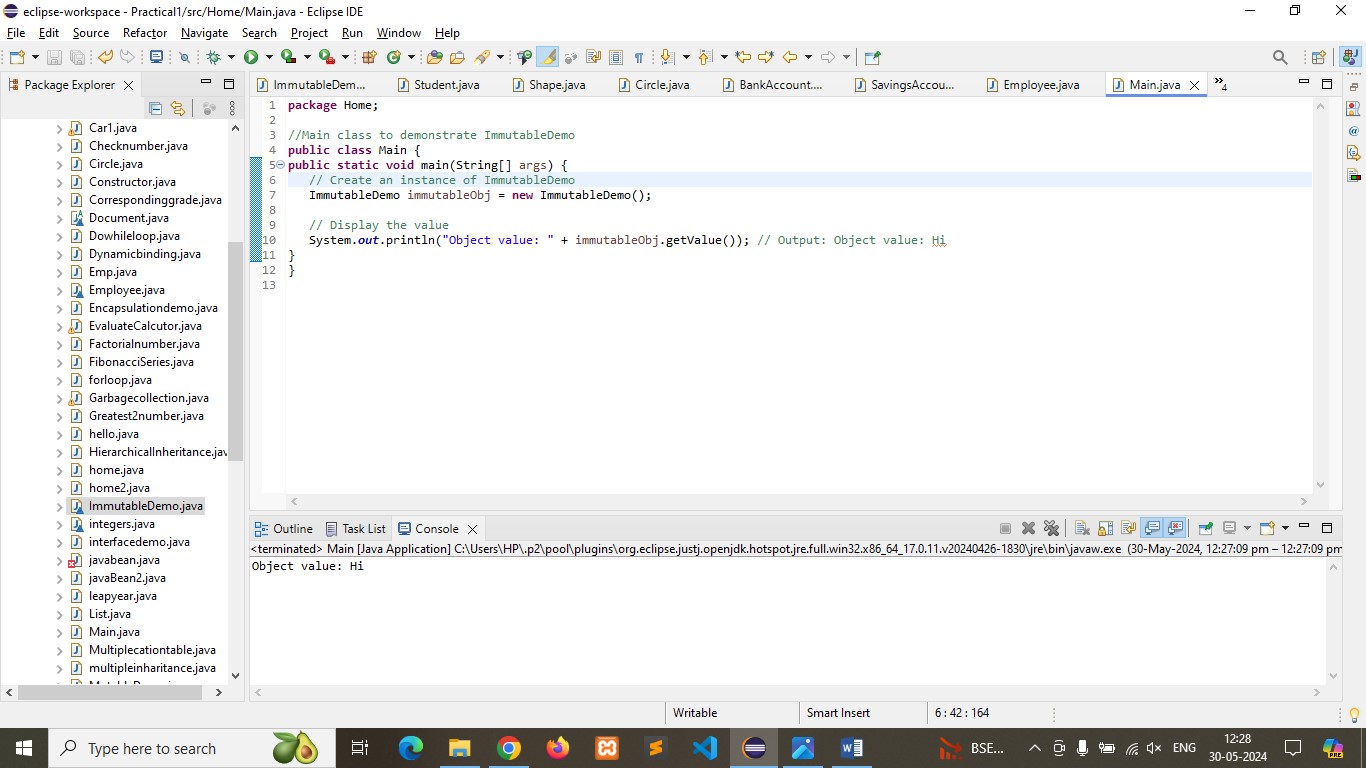
ImmutableExample immutableObj = **new** ImmutableExample();

// Display the value

System.***out***.println("Object value: " + immutableObj.getValue()); // Output: Object value: Hi

}

}

Output:

Q-7) Write the same programme for the class MutableExample, to output the object values ‘hello 2’ and ‘hello3’.

Program:

// Define the MutableExample class

**public** **class** MutableExample {

// Private instance variable to hold the value

**private** String value;

// Constructor to initialize the value

**public** MutableExample(String value) {

**this**.value = value;

}

// Getter method to access the value

**public** String getValue() {

**return** value;

}

// Setter method to update the value

**public** **void** setValue(String value) {

**this**.value = value;

}

}

//Main class to demonstrate MutableExample

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

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

// Create an instance of MutableExample with value "hello 2"

MutableExample mutableObj = **new** MutableExample("hello 2");

// Display the value

System.***out***.println("Object value: " + mutableObj.getValue()); // Output: Object value: hello 2

// Update the value to "hello3"

mutableObj.setValue("hello3");

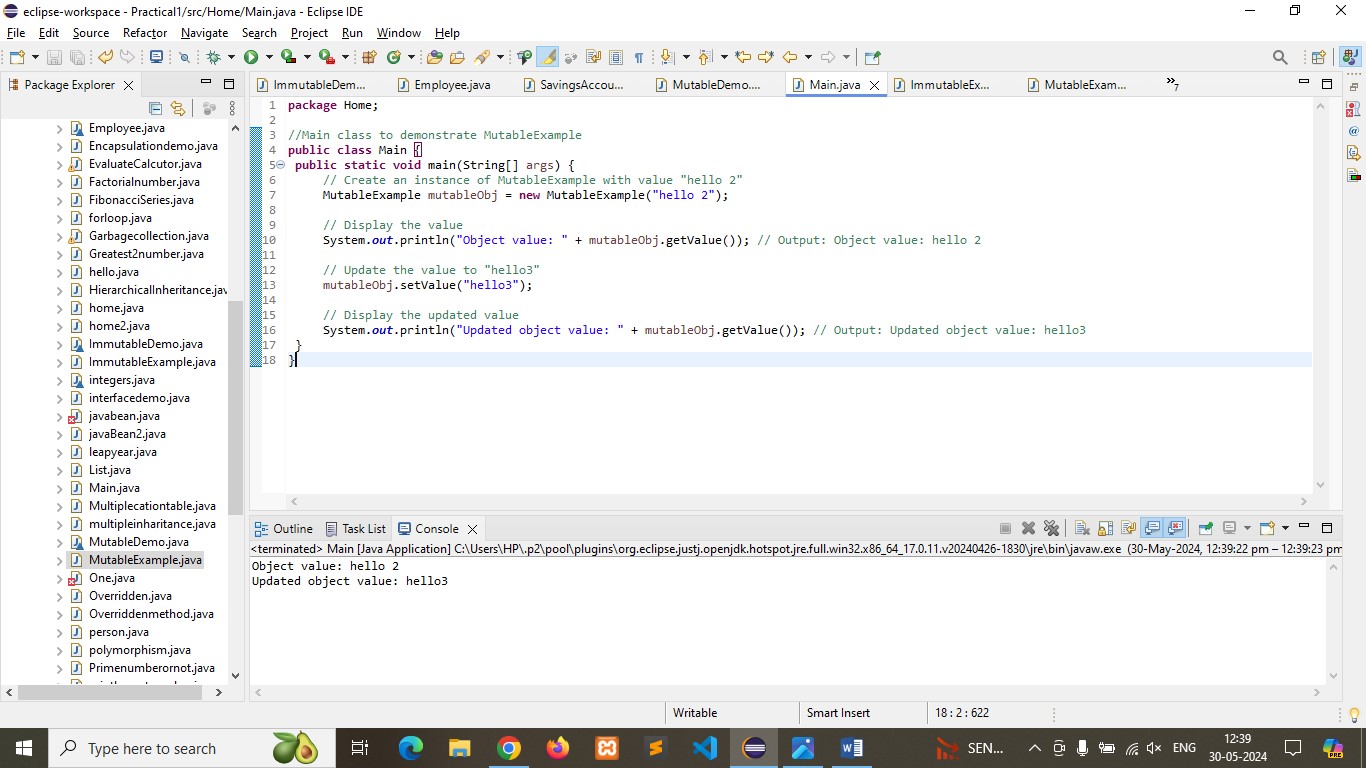
// Display the updated value

System.***out***.println("Updated object value: " + mutableObj.getValue()); // Output: Updated object value: hello3

}

}

Output:



Q-8) Write a java class to implement any 10 string methods:

● replace ● contains ● replaceAll ● indexOf ● substring ● Equals ● lastIndexOf ● startsWith

● endsWith ● EqualsIgnoreCase ● toLowerCase ● toUpperCase ● isEmpty ● Length ● split

Program:

**public** **class** StringMethodsExample {

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

// Example string

String str = "Hello, World!";

// Method 1: replace

String replaced = str.replace('o', 'a');

System.***out***.println("1. Replace: " + replaced); // Output: Hella, Warld!

// Method 2: contains

**boolean** contains = str.contains("World");

System.***out***.println("2. Contains 'World': " + contains); // Output: true

// Method 3: replaceAll

String replacedAll = str.replaceAll("[aeiouAEIOU]", "\*");

System.***out***.println("3. Replace All Vowels: " + replacedAll); // Output: H\*ll\*, W\*rld!

// Method 4: indexOf

**int** indexOfComma = str.indexOf(',');

System.***out***.println("4. Index of ',': " + indexOfComma); // Output: 5

// Method 5: substring

String substring = str.substring(7);

System.***out***.println("5. Substring from index 7: " + substring); // Output: World!

// Method 6: equals

String str2 = "Hello, World!";

**boolean** isEqual = str.equals(str2);

System.***out***.println("6. Equals: " + isEqual); // Output: true

// Method 7: lastIndexOf

**int** lastIndexOfL = str.lastIndexOf('l');

System.***out***.println("7. Last Index of 'l': " + lastIndexOfL); // Output: 10

// Method 8: startsWith

**boolean** startsWithHello = str.startsWith("Hello");

System.***out***.println("8. Starts with 'Hello': " + startsWithHello); // Output: true

// Method 9: endsWith

**boolean** endsWithExclamation = str.endsWith("!");

System.***out***.println("9. Ends with '!': " + endsWithExclamation); // Output: true

// Method 10: equalsIgnoreCase

String str3 = "hello, world!";

**boolean** isEqualIgnoreCase = str.equalsIgnoreCase(str3);

System.***out***.println("10. Equals Ignore Case: " + isEqualIgnoreCase); // Output: true

// Additional methods

// Method 11: toLowerCase

String lowerCase = str.toLowerCase();

System.***out***.println("11. To Lower Case: " + lowerCase);

// Output: hello, world!

// Method 12: toUpperCase

String upperCase = str.toUpperCase();

System.***out***.println("12. To Upper Case: " + upperCase);

// Output: HELLO, WORLD!

// Method 13: isEmpty

**boolean** isEmpty = str.isEmpty();

System.***out***.println("13. Is Empty: " + isEmpty);

// Output: false

// Method 14: length

**int** length = str.length();

System.***out***.println("14. Length: " + length);// Output: 13

// Method 15: split

String[] splitStr = str.split(",");

System.***out***.println("15. Split: ");

**for** (String s : splitStr) {

System.***out***.println(s.trim()); // Output: Hello\nWorld!

}

}

}

Output:

