Lab4

Practical1: 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** newone;

**public** **class** calculator {

// 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 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) {

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

// Testing add() methods

System.***out***.println("Sum of two integers: " + calculator.add(5, 10));

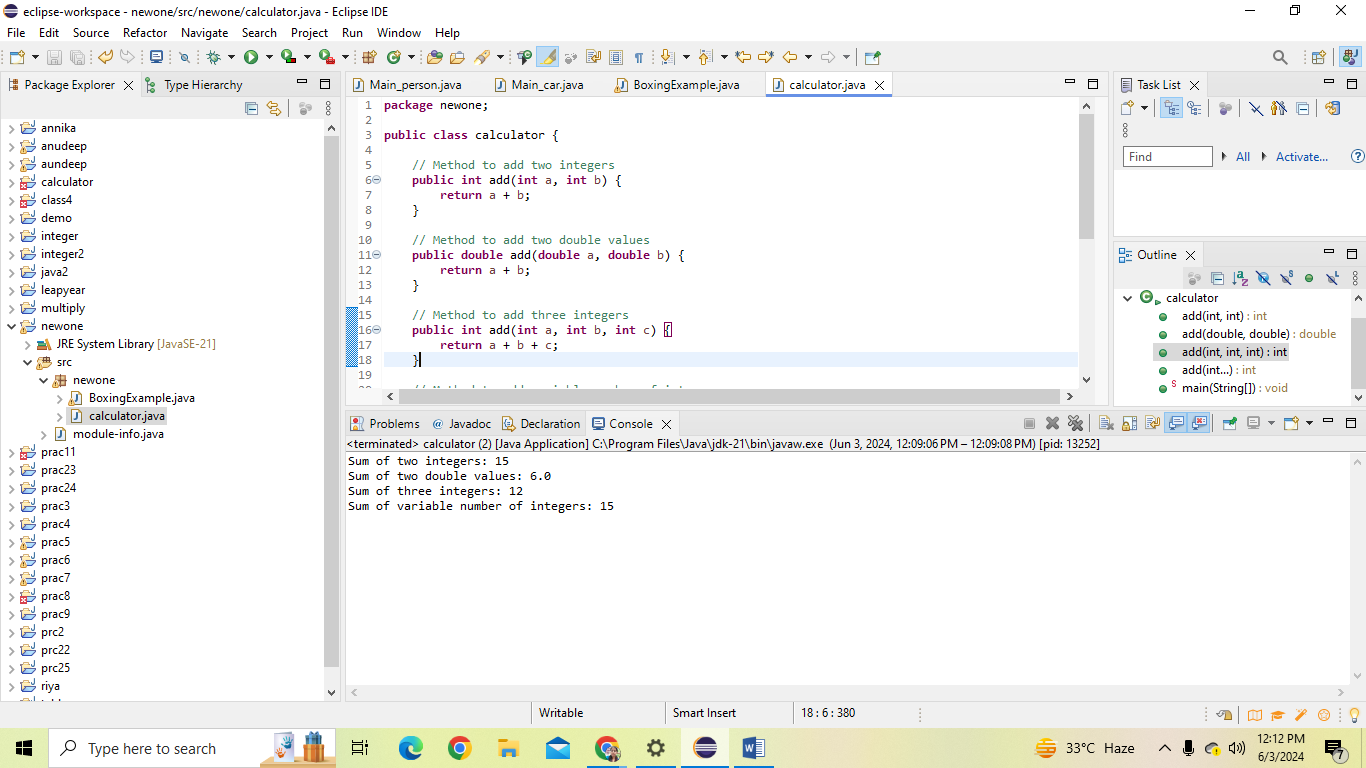
System.***out***.println("Sum of two double values: " + calculator.add(3.5, 2.5));

System.***out***.println("Sum of three integers: " + calculator.add(2, 4, 6));

System.***out***.println("Sum of variable number of integers: " + calculator.add(1, 2, 3, 4, 5));

}

}

**Output:** 

**Practical2:** 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

**Program: package** prac24;

**class** Person {

**private** String name;

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

// Constructor accepting and setting name and age

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

**this**.name = name;

**this**.age = age;

}

// Getter methods

**public** String getName() {

**return** name;

}

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

**return** age;

}

}

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

**private** **int** grade;

// Constructor of Student class using super to call the constructor of the parent class

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

**super**(name, age); // Call to parent class constructor

**this**.grade = grade;

}

// Getter method for grade

**public** **int** getGrade() {

**return** grade;

}

}

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

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

// Creating a Student object using the Student constructor

Student student1 = **new** Student("riya", 18, 12);

// Accessing properties using getters

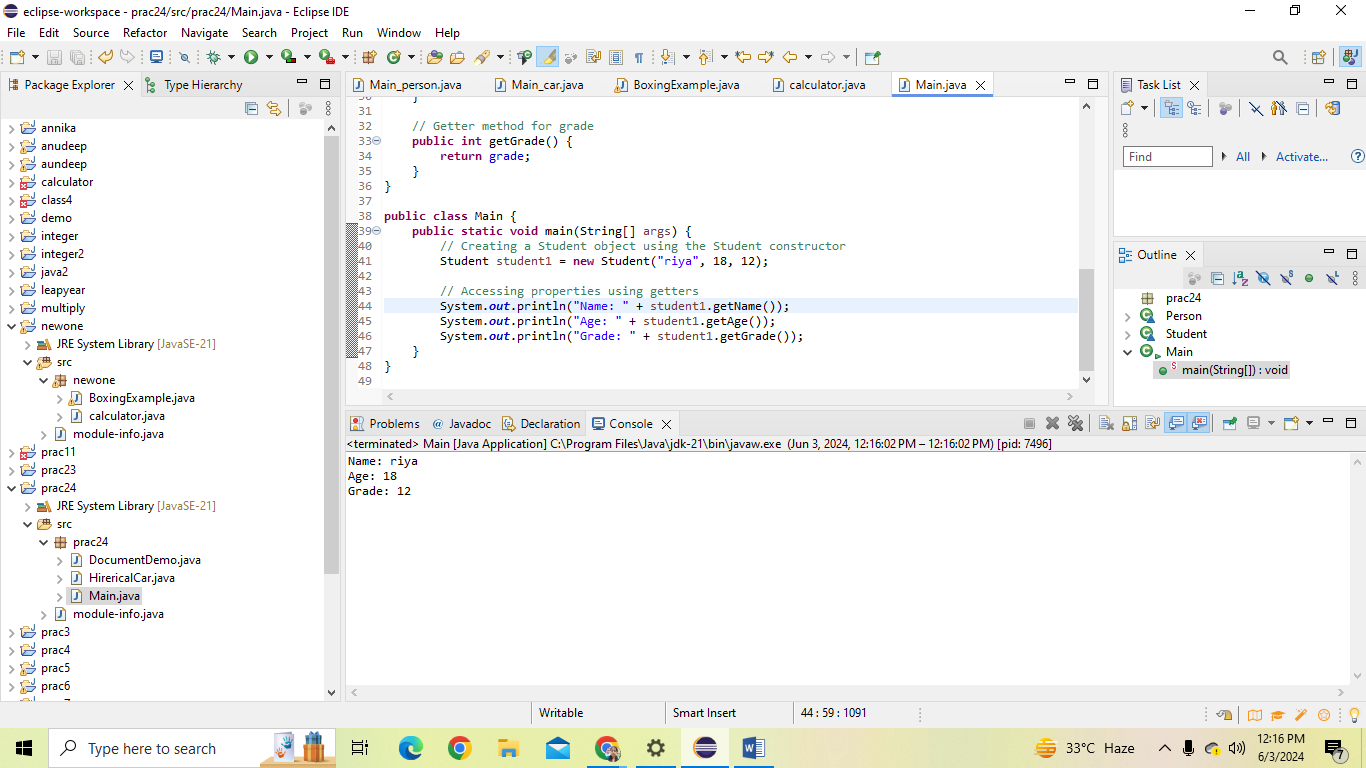
System.***out***.println("Name: " + student1.getName());

System.***out***.println("Age: " + student1.getAge());

System.***out***.println("Grade: " + student1.getGrade());

}

}

**Output:** 

Practical3:  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**: **package** prac24;

**class** Shape {

// Method draw() that prints "Drawing Shape"

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

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

}

}

//Subclass Circle inheriting from Shape

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

// Overriding draw() method to print "Drawing Circle"

@Override

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

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

// Call draw() method of the Shape class using super.draw()

**super**.draw();

}

}

//Main class to demonstrate calling draw() on a Circle object

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

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

// Create a Circle object

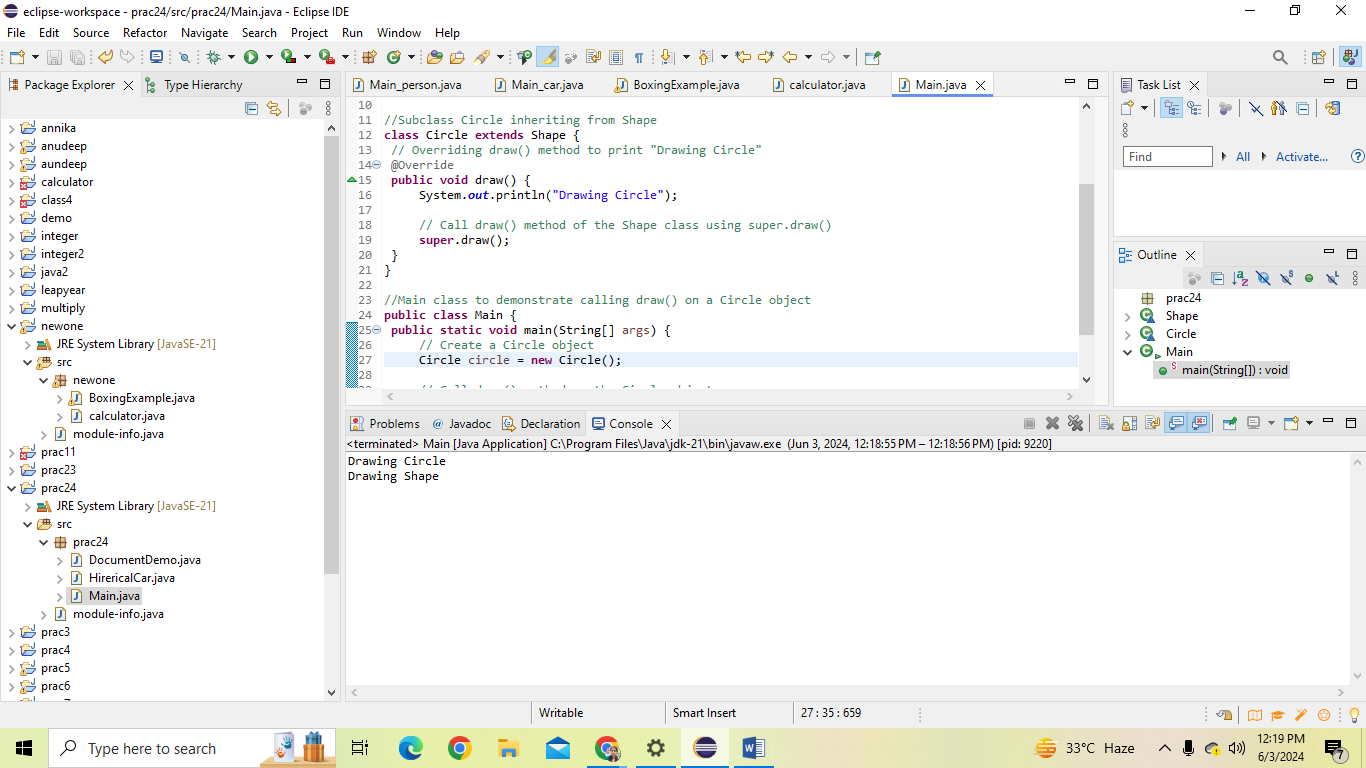
Circle circle = **new** Circle();

// Call draw() method on the Circle object

circle.draw();

}

}

**Output:** 

Practical 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: package** prac24;

//Base class BankAccount

**class** BankAccount {

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

// Constructor to set the initial balance

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

**this**.balance = initialBalance;

}

// Method to deposit amount

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

balance += amount;

}

}

//Subclass SavingsAccount inheriting from BankAccount

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

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

// Constructor to set the initial balance and interest rate

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

**super**(initialBalance);

**this**.interestRate = interestRate;

}

// Override deposit method to add interest before depositing

@Override

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

**double** interest = balance \* interestRate / 100; // Calculate interest

balance += interest; // Add interest to balance

**super**.deposit(amount); // Call deposit method of the base class

}

}

//Main class to demonstrate creating a SavingsAccount and depositing an amount

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

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

// Create a SavingsAccount object with initial balance $1000 and interest rate 5%

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

// Deposit an amount of $200

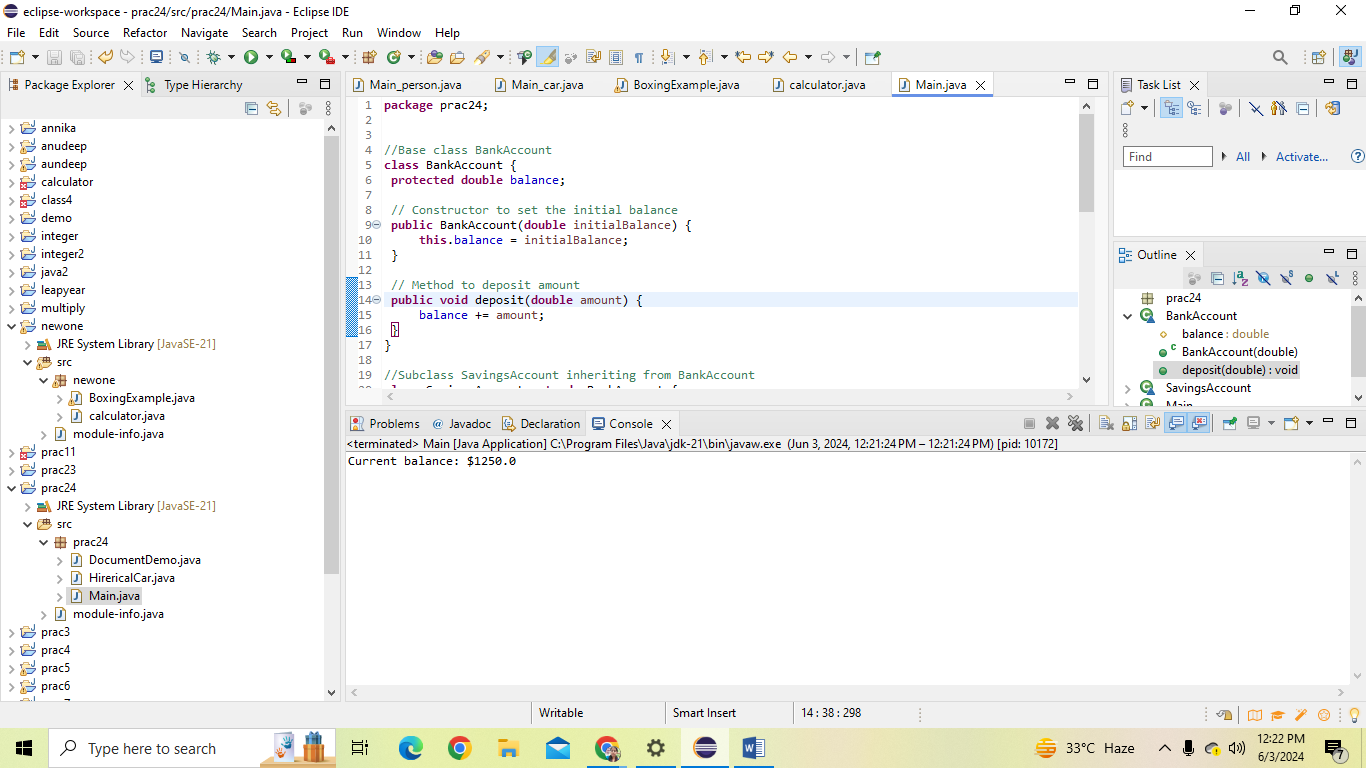
savingsAccount.deposit(200);

// Display the current balance

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

}

}

**Output:** 

Practical5: 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: package** demo;

//Base class Employee

**class** Employee {

**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);

}

}

//Subclass Manager inheriting from Employee

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

**private** String department;

// Constructor

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

**super**(name, salary); // Call to the constructor of the base class

**this**.department = department;

}

// Override displayDetails method to include department details

@Override

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

**super**.displayDetails(); // Call displayDetails method of the base class

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

}

}

//Main class

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

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

// Create an Employee object

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

// Create a Manager object

Manager manager = **new** Manager("manisha", 75000, "Marketing");

// Call displayDetails to show details for Employee

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

employee.displayDetails();

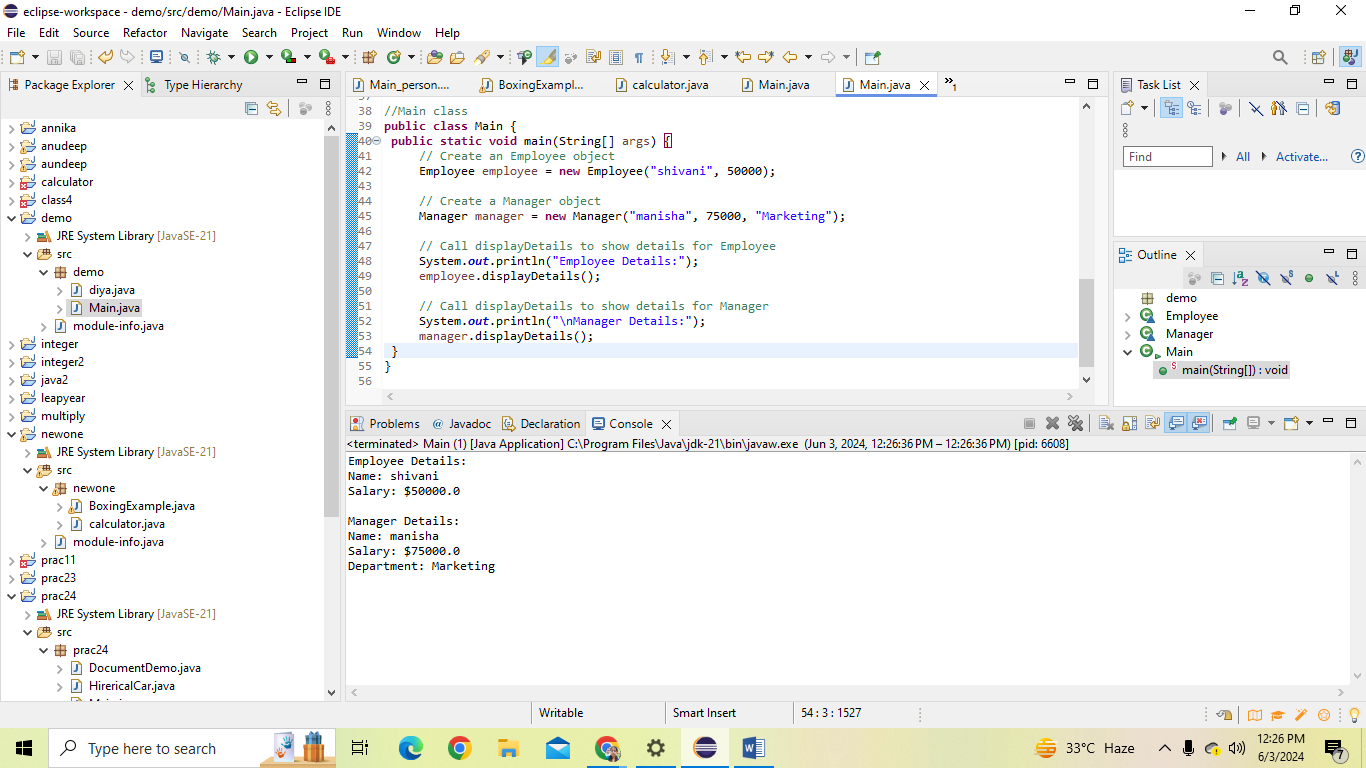
// Call displayDetails to show details for Manager

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

manager.displayDetails();

}

}

**Output:** 

Practical 6: Write the same programme for the class ImmutableExample, to achieve object value ‘Hi’.

**Program**: **package** anudeep;

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

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

// Constructor to initialize the value

**public** ImmutableExample(String value) {

**this**.value = value;

}

// Method to return the value

**public** String getValue() {

**return** value;

}

// No setter methods to make the class immutable

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

// Create an object of ImmutableExample

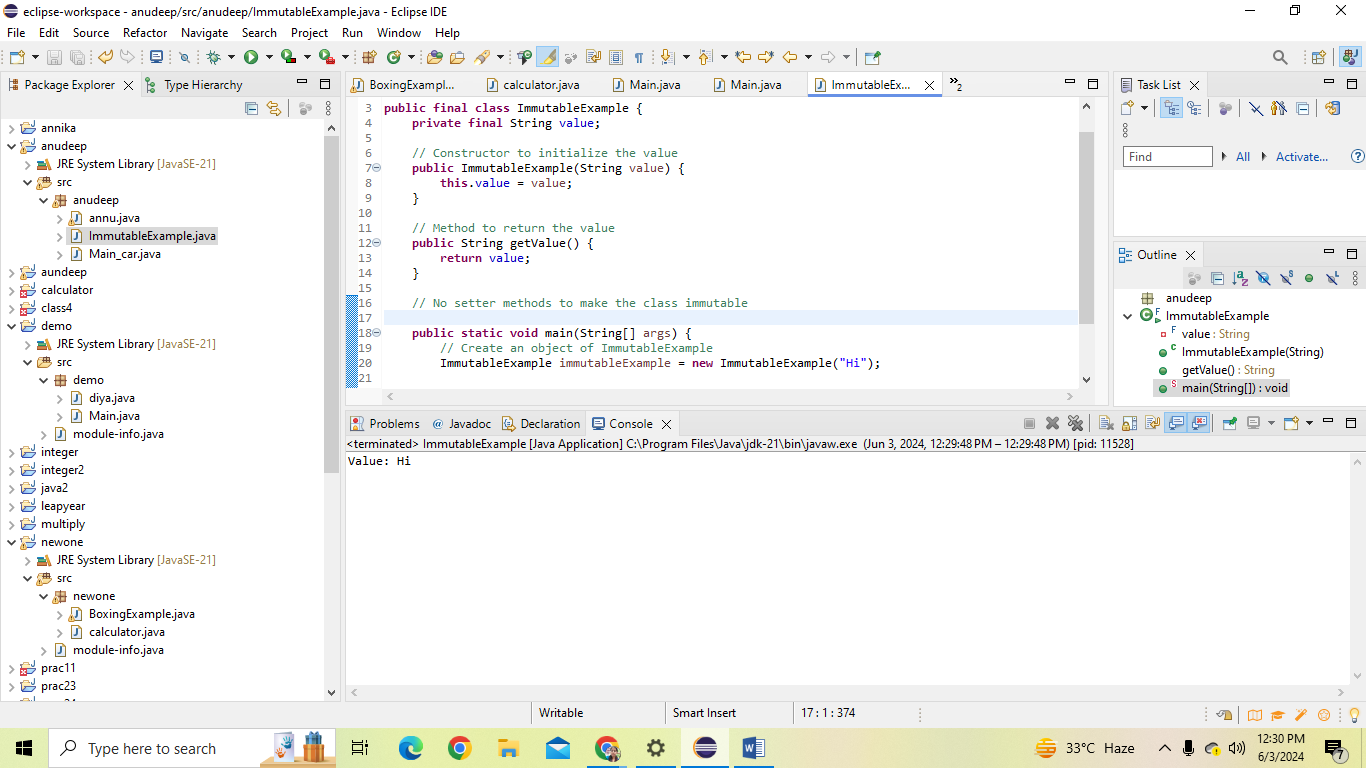
ImmutableExample immutableExample = **new** ImmutableExample("Hi");

// Display the value

System.***out***.println("Value: " + immutableExample.getValue());

}

}

Output: 

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

**Program**: **package** annika;

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

**private** String value;

// Constructor to initialize the value

**public** MutableExample(String value) {

**this**.value = value;

}

// Method to set the value

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

**this**.value = value;

}

// Method to get the value

**public** String getValue() {

**return** value;

}

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

// Create an object of MutableExample

MutableExample mutableExample = **new** MutableExample("hello");

// Output the initial value

System.***out***.println("Initial value: " + mutableExample.getValue());

// Set new value and output

mutableExample.setValue("hello 2");

System.***out***.println("New value: " + mutableExample.getValue());

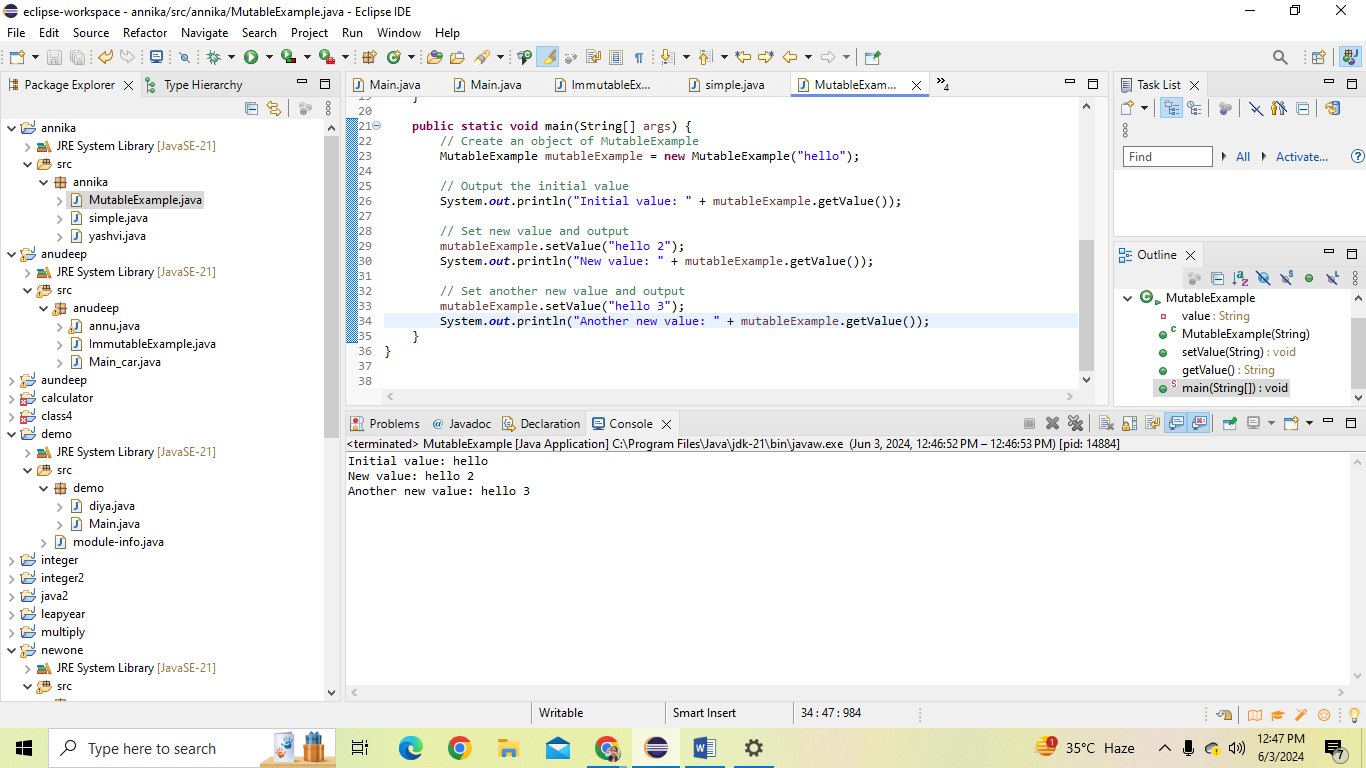
// Set another new value and output

mutableExample.setValue("hello 3");

System.***out***.println("Another new value: " + mutableExample.getValue());

}

}

Output: 

Practical 8: Write a java class to implement any 10 string methods:

**Program: package** riya;

**public** **class** main {

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

// Example string

String str = "Hello, World!";

// Example usage of string methods

System.***out***.println("Original String: " + str);

System.***out***.println("Replace 'Hello' with 'Hi': " + str.replace("Hello", "Hi"));

System.***out***.println("Contains 'World': " + str.contains("World"));

System.***out***.println("Replace all 'l' with 'x': " + str.replaceAll("l", "x"));

System.***out***.println("Index of 'o': " + str.indexOf("o"));

System.***out***.println("Substring from index 7: " + str.substring(7));

System.***out***.println("Equals 'Hello, World!': " + str.equals("Hello, World!"));

System.***out***.println("Last index of 'o': " + str.lastIndexOf("o"));

System.***out***.println("Starts with 'Hello': " + str.startsWith("Hello"));

System.***out***.println("Ends with '!': " + str.endsWith("!"));

System.***out***.println("Equals ignore case 'hello, world!': " + str.equalsIgnoreCase("hello, world!"));

System.***out***.println("To lower case: " + str.toLowerCase());

System.***out***.println("To upper case: " + str.toUpperCase());

System.***out***.println("Is empty: " + str.isEmpty());

System.***out***.println("Length of string: " + str.length());

System.***out***.println("Split by comma: ");

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

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

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

}

}

}

**Output:** 