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

**package** lab4;

**public** **class** Calculator {

// Method to add two double values

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

**return** a + b;

}

// Method to add three integers

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

**return** a + b;

}

// 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) {

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

// Test adding two integers

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

// Test adding two doubles

System.***out***.println("Sum of 2.5 and 3.5: " + c.add(2.5, 3.5));

// Test adding three integers

System.***out***.println("Sum of 3, 6, and 9: " + c.add(3, 6, 9));

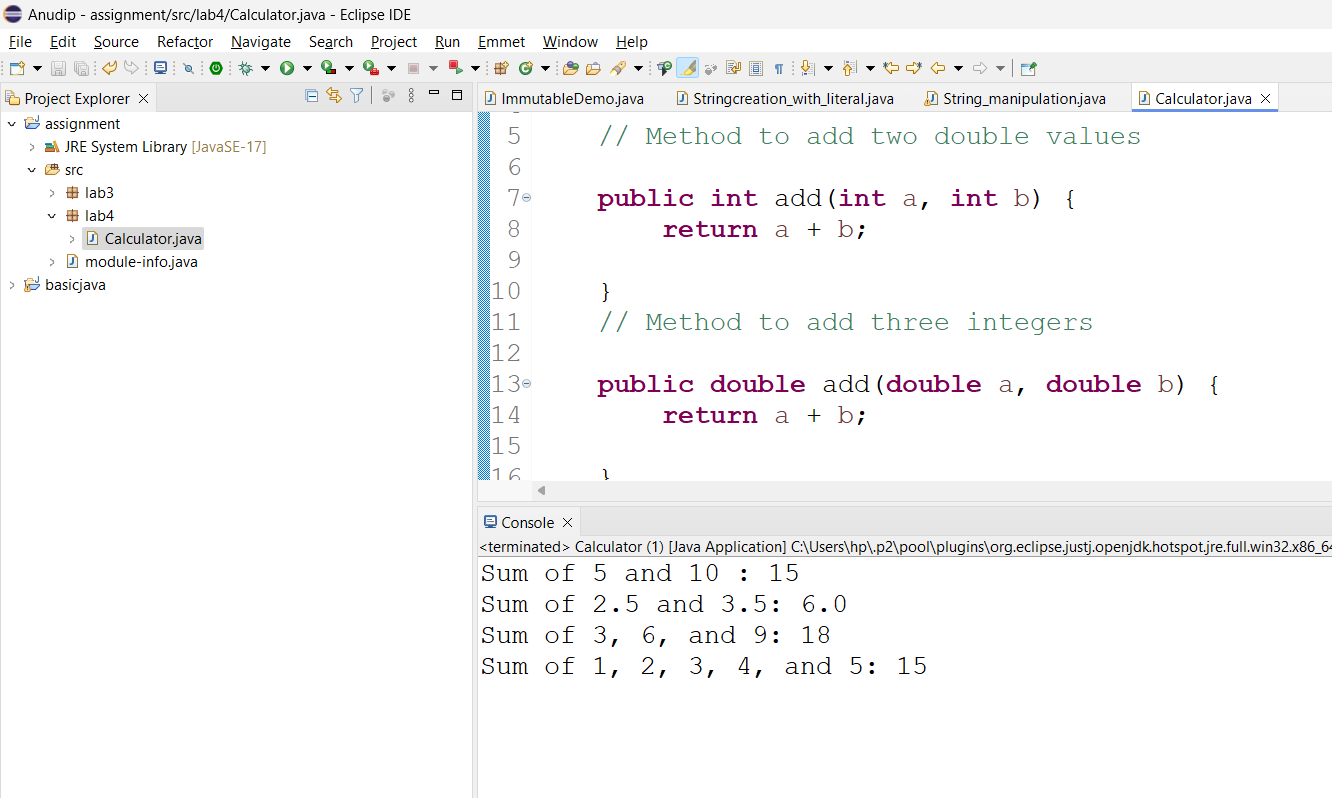
// Test adding a variable number of integers

System.***out***.println("Sum of 1, 2, 3, 4, and 5: " + c.add(1, 2, 3, 4, 5));

}

}

**OutPut:**



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.

**package** lab4;

**class** People {

**private** String name;

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

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

**super**();

**this**.name = name;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

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

**return** age;

}

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

**this**.age = age;

}

}

**public** **class** Emp **extends** People {

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

// Constructor

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

**super**(name, age); // Calling the parent class constructor

**this**.grade = grade;

}

// Getter

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

**return** grade;

}

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

Emp e = **new** Emp("vijaya", 12, 22);

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

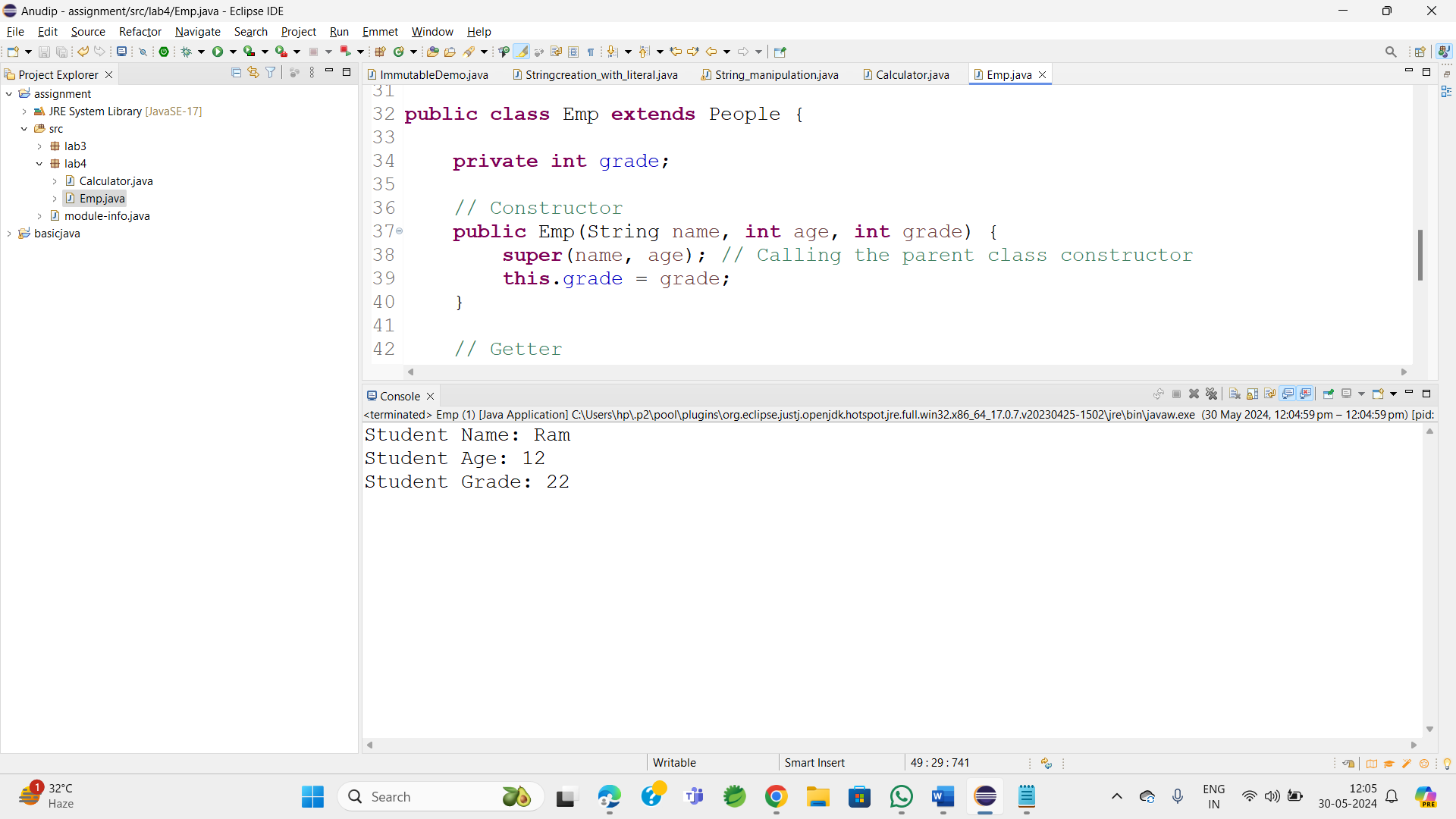
System.***out***.println("Student Age: " + e.getAge());

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

}

}

**OutPut:**

****

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.

**package** lab4;

**class** Shape {

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

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

}

}

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

@Override

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

**super**.draw();// Calling the draw() method of the Shape class

}

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

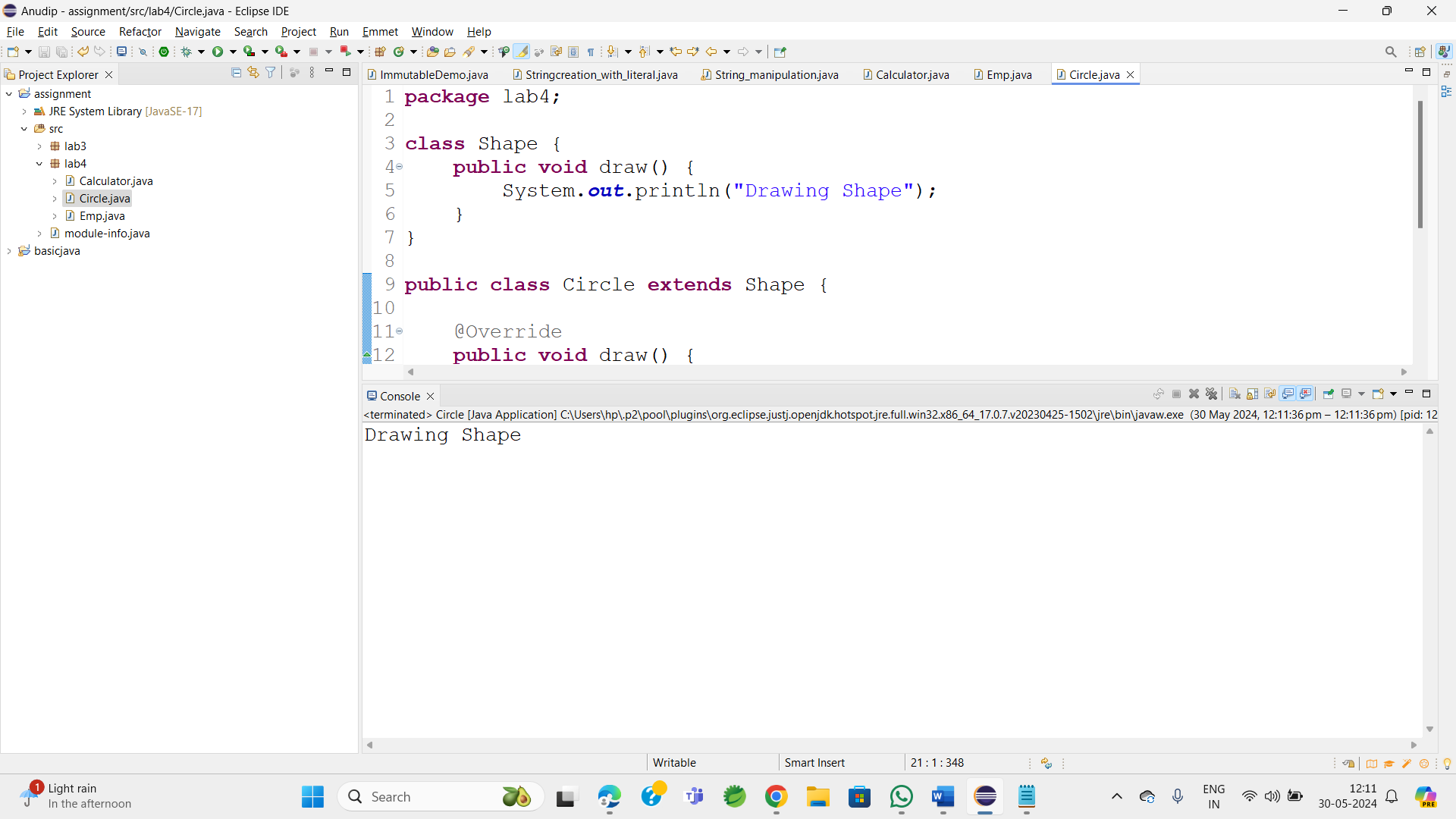
Circle circle = **new** Circle();

circle.draw();

}

}

**OutPut:**

****

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.

**package** lab4;

**class** BankAccount {

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

// Constructor to set initial balance

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

**this**.balance = initialBalance;

}

// Method to deposit amount

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

balance += amount;

}

}

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

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

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

**super**(initialBalance);

**this**.interestRate = interestRate;

}

@Override

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

**double** interest = balance \* (interestRate / 100);

**super**.deposit(interest); // Calling deposit method of the base class

**super**.deposit(amount);

}

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

SavingsAccount savingsAccount = **new** SavingsAccount(1000, 5); // Initial balance: 1000, Interest Rate: 5%

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

// Deposit an amount

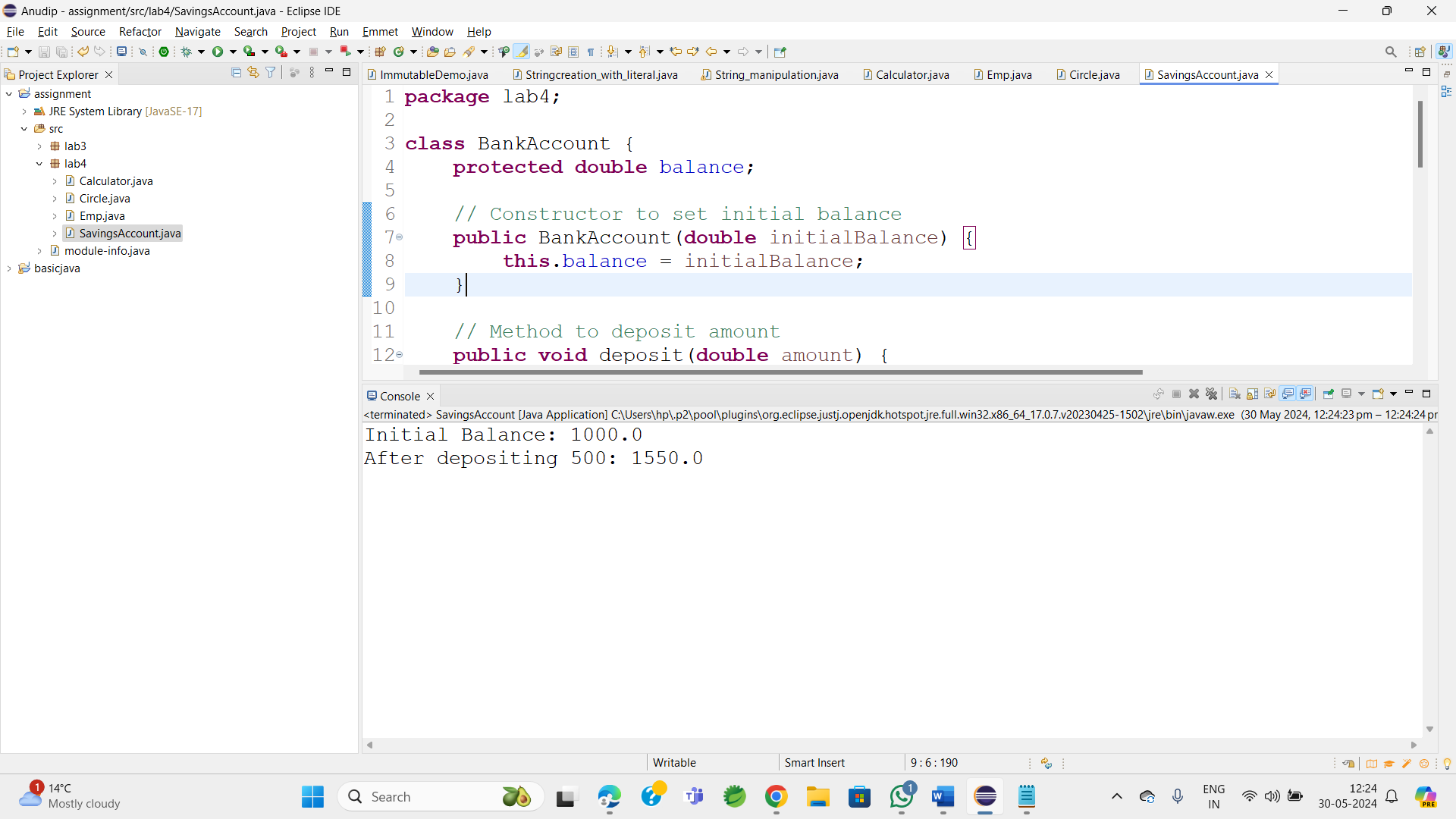
savingsAccount.deposit(500);

System.***out***.println("After depositing 500: " + savingsAccount.balance);

}

}

**OutPut:**



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.

**package** lab4;

**class** Employee {

**protected** String name;

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

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

**super**();

**this**.name = name;

**this**.salary = salary;

}

// Method to display details

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

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

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

}

}

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

**private** String department;

// Constructor

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

**super**(name, salary);

**this**.department = department;

}

@Override

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

**super**.displayDetails();

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

}

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

Employee emp = **new** Employee("Jay", 50000);

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

emp.displayDetails();

System.***out***.println(); // Adding a line break for better readability

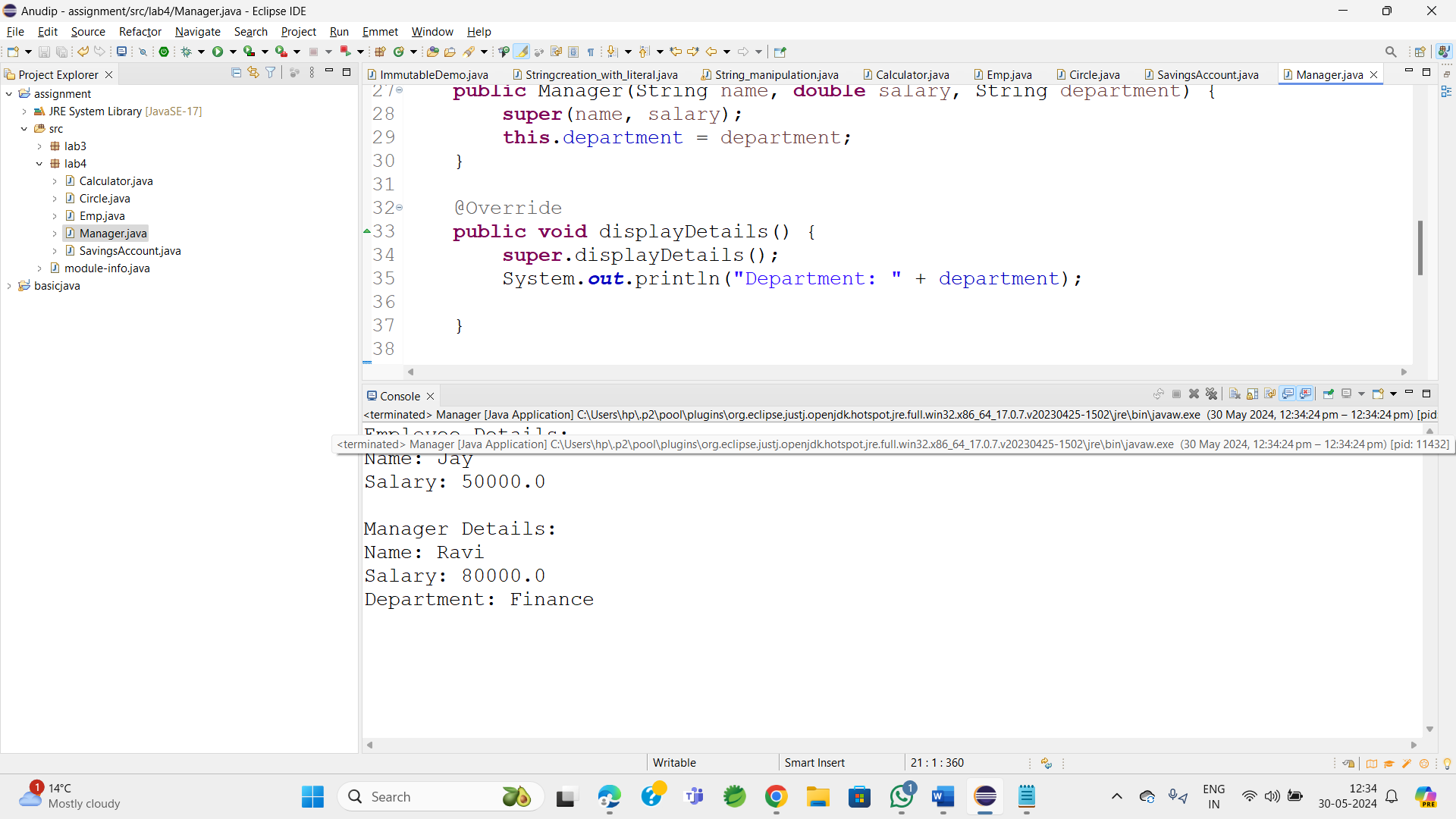
// Creating Manager object

Manager manager = **new** Manager("Ravi", 80000, "Finance");

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

manager.displayDetails();

}

}**OutPut: **

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

**package** lab4;

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

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

// Constructor

**public** ImmutableExample() {

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

}

// Getter method

**public** String getValue() {

**return** value;

}

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

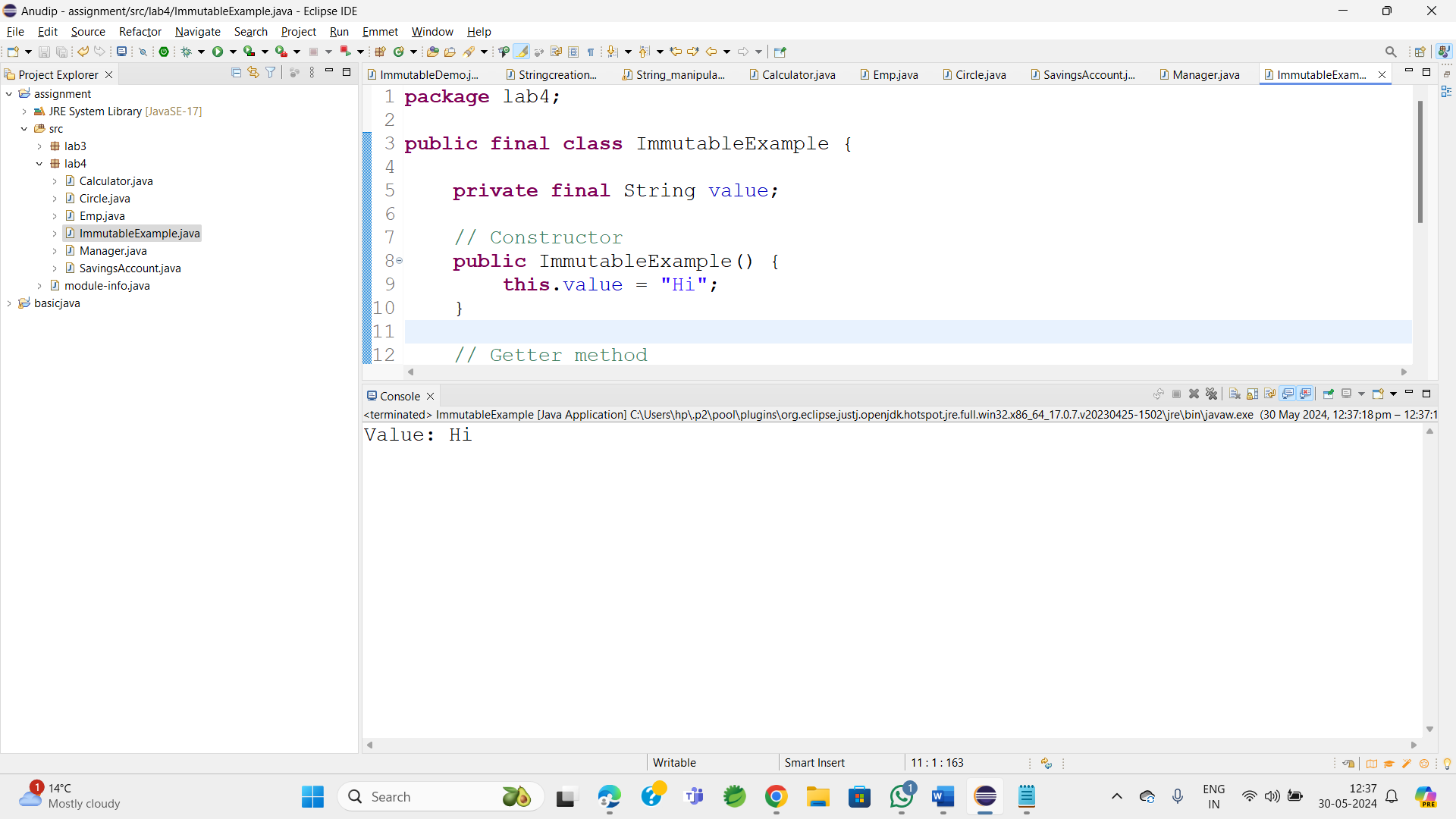
ImmutableExample obj = **new** ImmutableExample();

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

}

}

OutPut:



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

**package** lab4;

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

**private** String value;

**public** MutableExample(String value) {

**super**();

**this**.value = value;

}

**public** String getvalue() {

**return** value;

}

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

**this**.value = value;

}

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

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

System.***out***.println("Initial Value: " + m.getvalue());

// Modify value to "hello 2"

m.setvalue("hello 2");

System.***out***.println("Modified Value: " + m.getvalue());

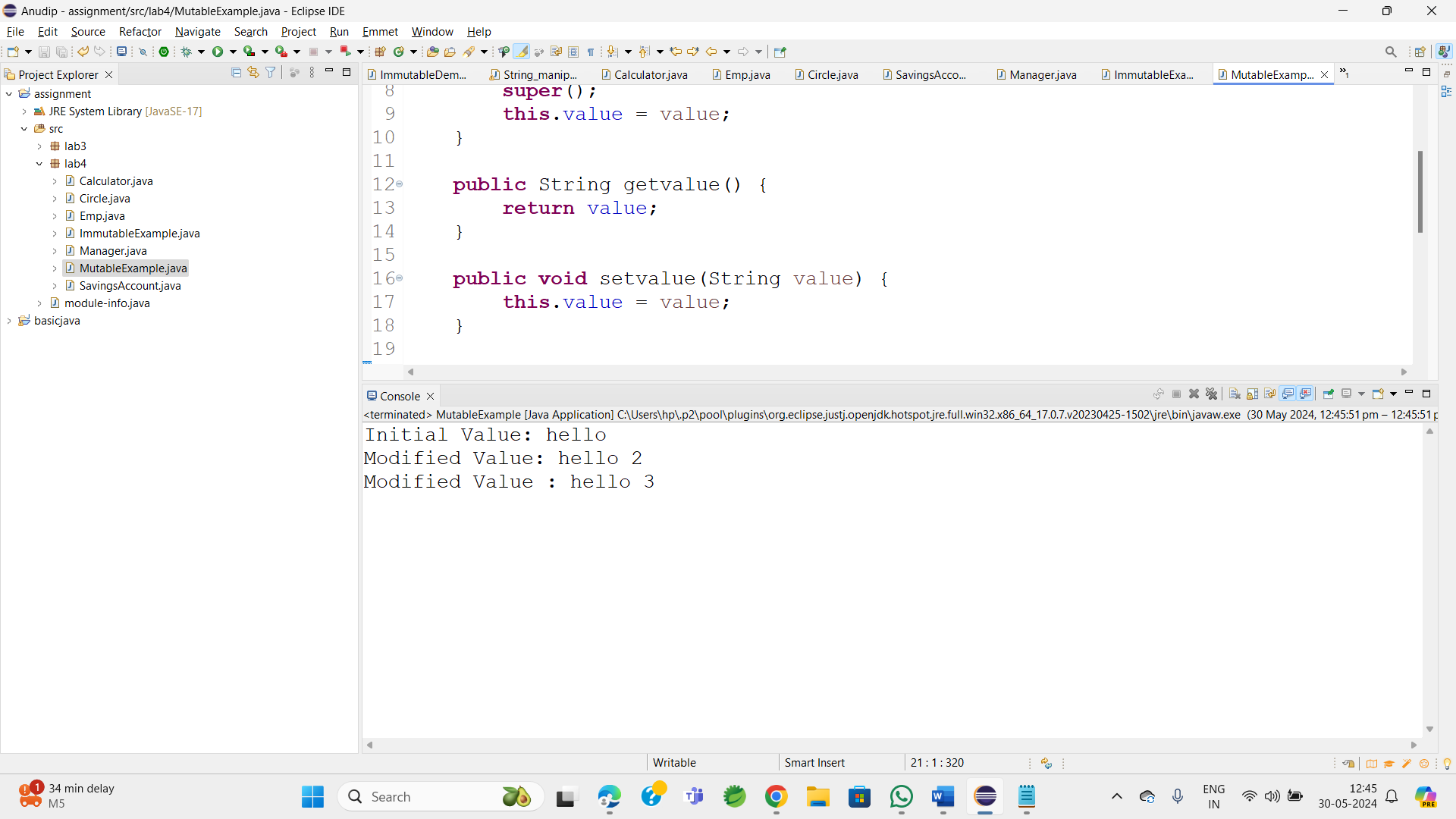
// Modify value to "hello3"

m.setvalue("hello 3");

System.***out***.println("Modified Value : " + m.getvalue());

}}

Output:



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

**package** lab4;

**public** **class** StringMethodsDemo {

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

String str = "Hello World";

// replace

String replacedStr = str.replace('l', 'z');

System.***out***.println("Replace: " + replacedStr);

// contains

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

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

// replaceAll

String replacedAllStr = str.replaceAll("World", "Java");

System.***out***.println("ReplaceAll: " + replacedAllStr);

// indexOf

**int** index = str.indexOf('o');

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

// substring

String subStr = str.substring(6);

System.***out***.println("Substring from index 6: " + subStr);

// equals

String anotherStr = "Hello World";

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

System.***out***.println("Equals: " + isEqual);

// lastIndexOf

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

System.***out***.println("Last index of 'l': " + lastIndex);

// startsWith

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

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

// endsWith

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

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

// equalsIgnoreCase

**boolean** equalsIgnoreCase = str.equalsIgnoreCase("hello world");

System.***out***.println("Equals Ignore Case 'hello world': " + equalsIgnoreCase);

// toLowerCase

String lowerCaseStr = str.toLowerCase();

System.***out***.println("Lowercase: " + lowerCaseStr);

// toUpperCase

String upperCaseStr = str.toUpperCase();

System.***out***.println("Uppercase: " + upperCaseStr);

// isEmpty

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

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

// length

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

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

// split

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

System.***out***.print("Split: ");

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

System.***out***.print(s + ", ");

}

}

}

OutPut:

