**LAB 4** (JAVA)

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

**CODE:**

**package** bhargavi;

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

// Add method for two integers

**public** **int** add(**int** num1, **int** num2) {

**return** num1 + num2;

}

// Add method for two double values

**public** **double** add(**double** num1, **double** num2) {

**return** num1 + num2;

}

// Add method for three integers

**public** **int** add(**int** num1, **int** num2, **int** num3) {

**return** num1 + num2 + num3;

}

// Add method for a variable number of integers

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

**int** sum = 0;

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

sum += num;

}

**return** sum;

}

// Main method to test the Calculator class

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

Calculator calc = **new** Calculator();

// Testing add() method for two integers

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

// Testing add() method for two double values

System.***out***.println("Addition of two doubles: " + calc.add(3.5, 2.5));

// Testing add() method for three integers

System.***out***.println("Addition of three integers: " + calc.add(7, 8, 9));

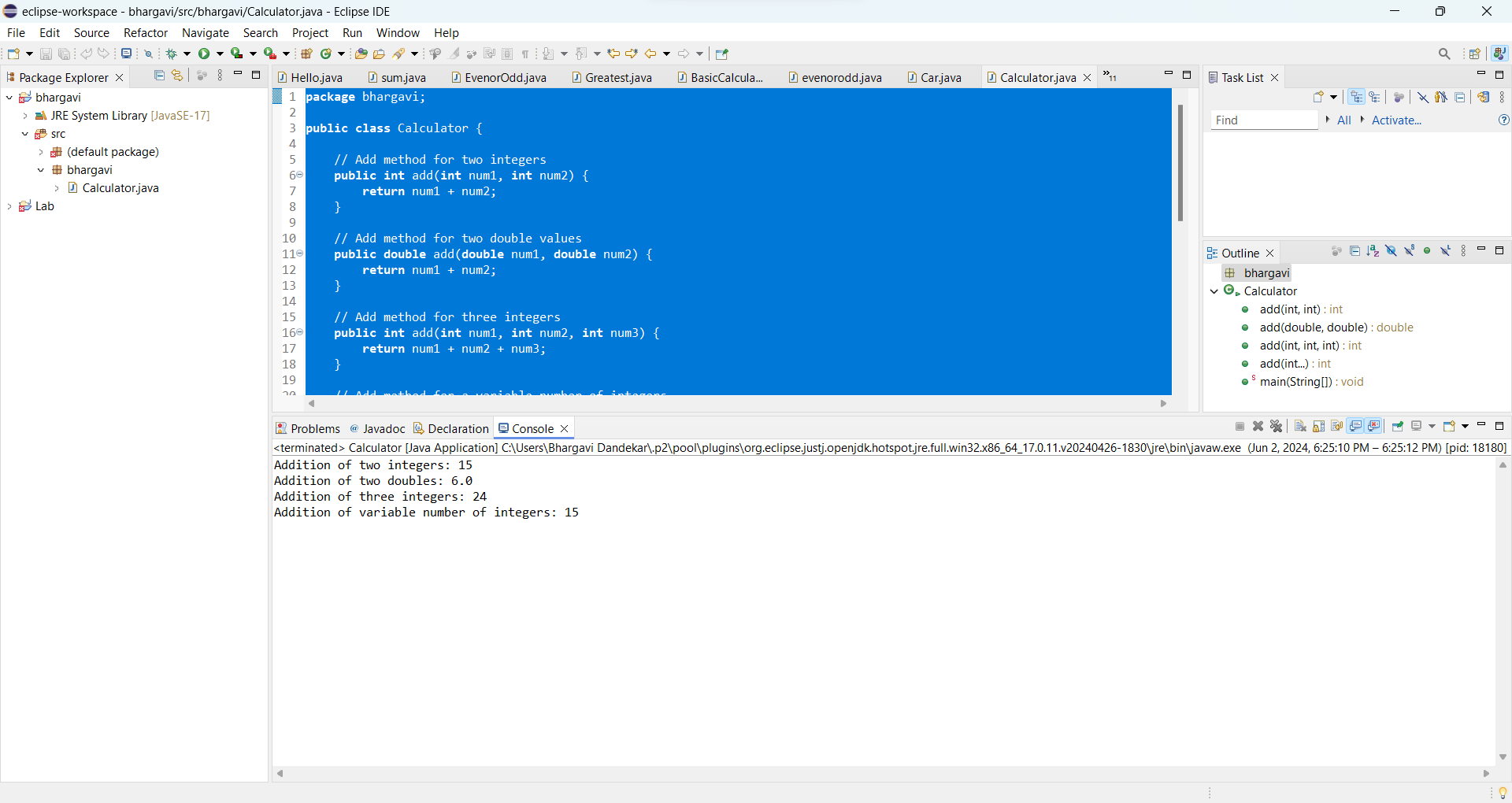
// Testing add() method for a variable number of integers

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

}

}

**OUTPUT:**



* The add() method is overloaded to handle different types and numbers of parameters.
* We have methods to add two integers, two double values, three integers, and a variable number of integers using varargs.

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.

**CODE:**

**package** bhargavi;

// Parent class Person

**class** Person {

String name;

**int** age;

// Constructor accepting and setting name and age

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

**this**.name = name;

**this**.age = age;

}

}

// Subclass Student inheriting from Person

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

**int** grade;

// Constructor initializing name and age using super keyword

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

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

**this**.grade = grade;

}

}

// Main class to demonstrate usage

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

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

// Creating a Student object

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

// Accessing the properties of the Student object

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

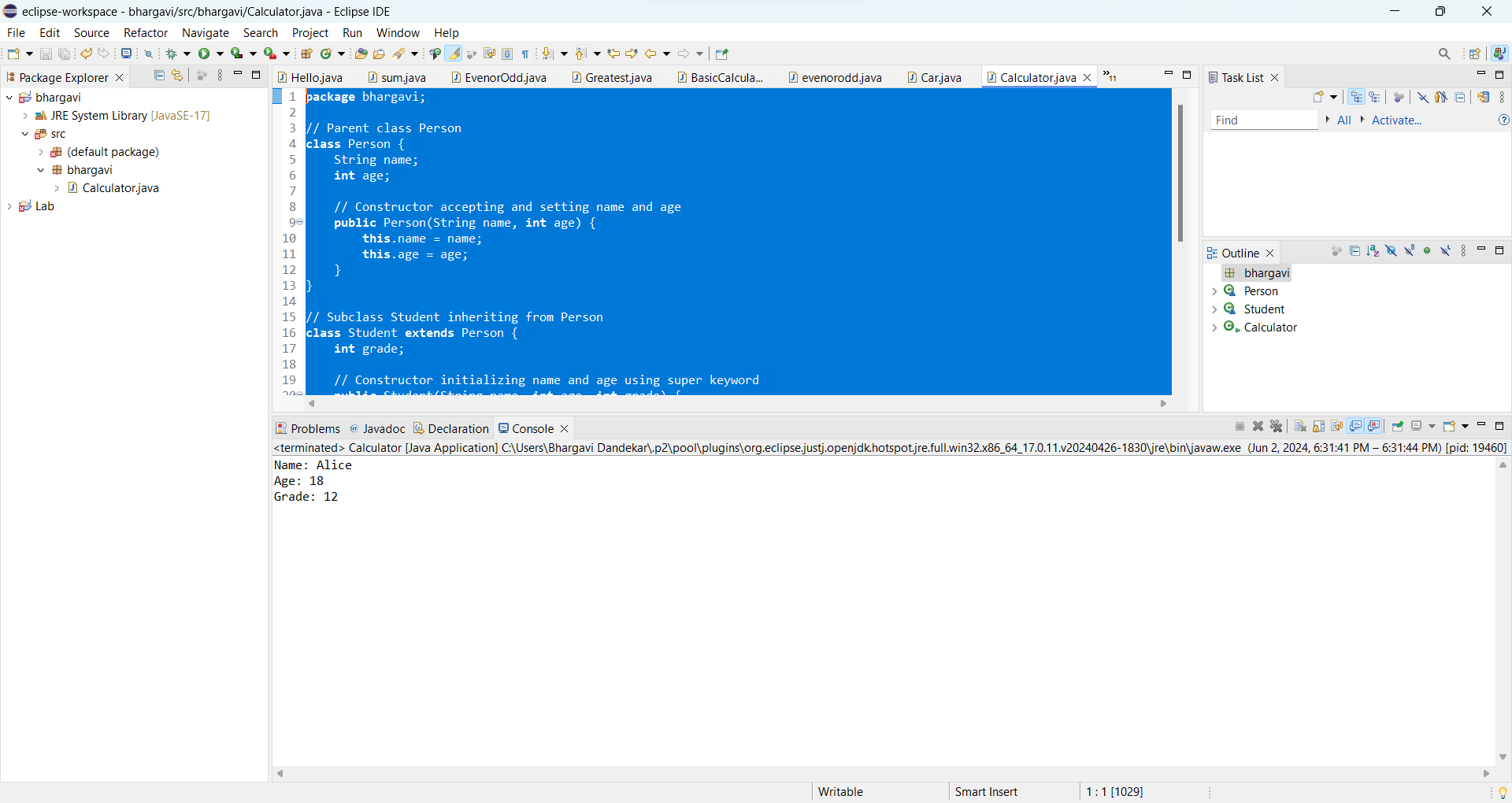
System.***out***.println("Age: " + student1.age);

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

}

}

**OUTPUT:**

****

 We have a parent class Person with properties name and age, and a constructor to initialize these properties.

 We have a subclass Student that extends Person and adds a property grade.

 In the constructor of the Student class, we use the super keyword to call the constructor of the parent class Person, passing the name and age parameters.

 We demonstrate the creation of a Student object and access its properties.

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.

**CODE:**

**package** bhargavi;

// Base class Shape

**class** Shape {

// Method to draw shape

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

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

}

}

// Subclass Circle inheriting from Shape

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

// Method to draw circle

@Override

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

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

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

}

}

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

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

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

// Creating a Circle object

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

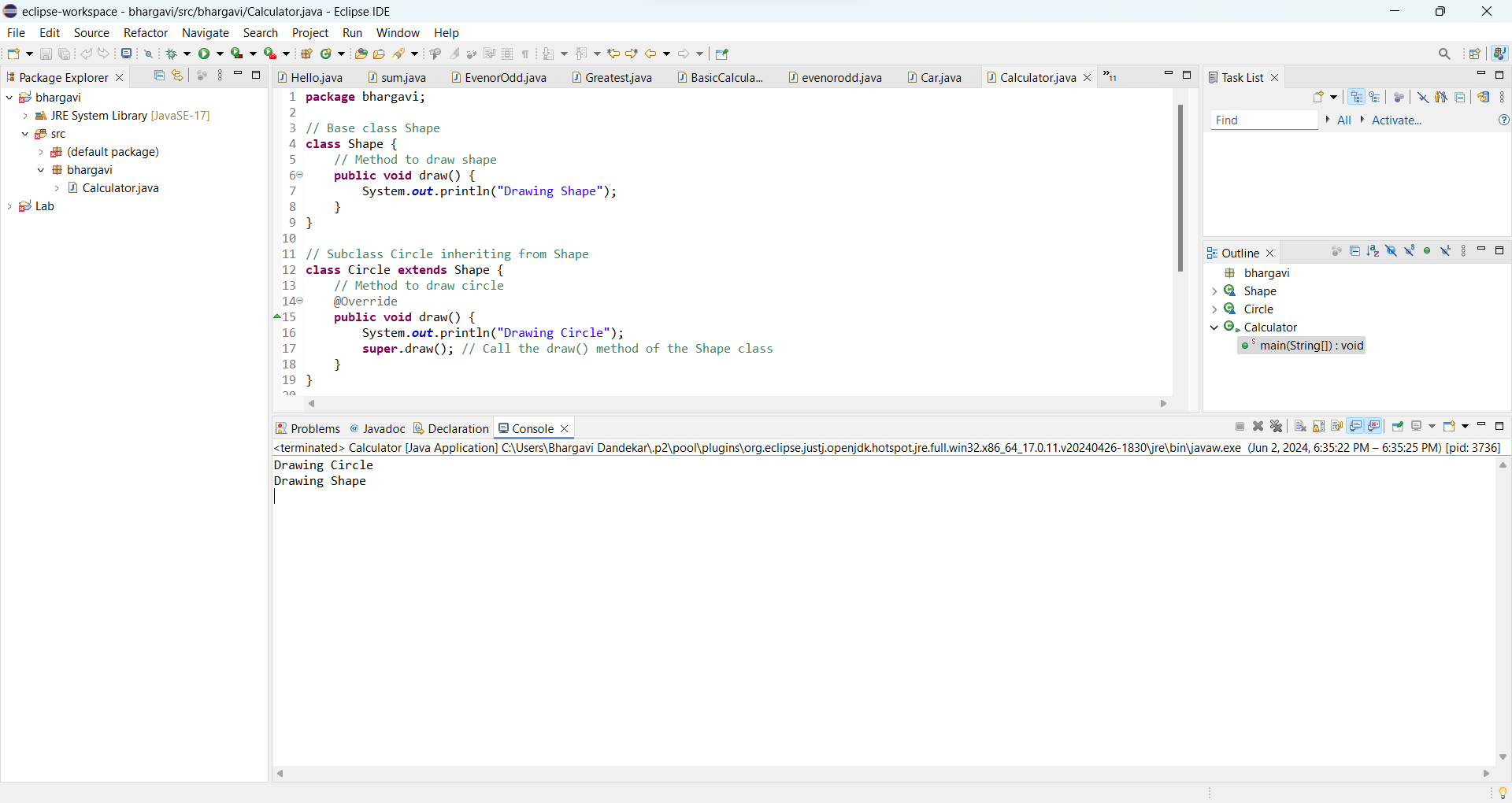
// Calling the draw() method of Circle

circle.draw();

}

}

**OUTPUT:**



* We have a base class Shape with a method draw() that prints Drawing Shape.
* We have a subclass Circle that overrides the draw() method to print Drawing Circle.
* Inside the draw() method of Circle, we call the draw() method of the Shape class using super.draw().
* We demonstrate calling the draw() method on a Circle object in the main method.

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.

**CODE:**

**package** bhargavi;

// Base class BankAccount

**class** BankAccount {

**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;

}

}

// Subclass SavingsAccount inheriting from BankAccount

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

**double** interestRate;

// Constructor to set initial balance and interest rate

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

**super**(initialBalance); // Call to the base class constructor

**this**.interestRate = interestRate;

}

// Method to deposit amount with interest

@Override

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

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

**super**.deposit(interest); // Call the deposit() method of the base class to add interest

**super**.deposit(amount); // Deposit the original amount

}

}

// Main class to demonstrate creating a SavingsAccount and depositing an amount to see the effect of interest

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

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

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

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

// Depositing $200 into the account

savingsAccount.deposit(200);

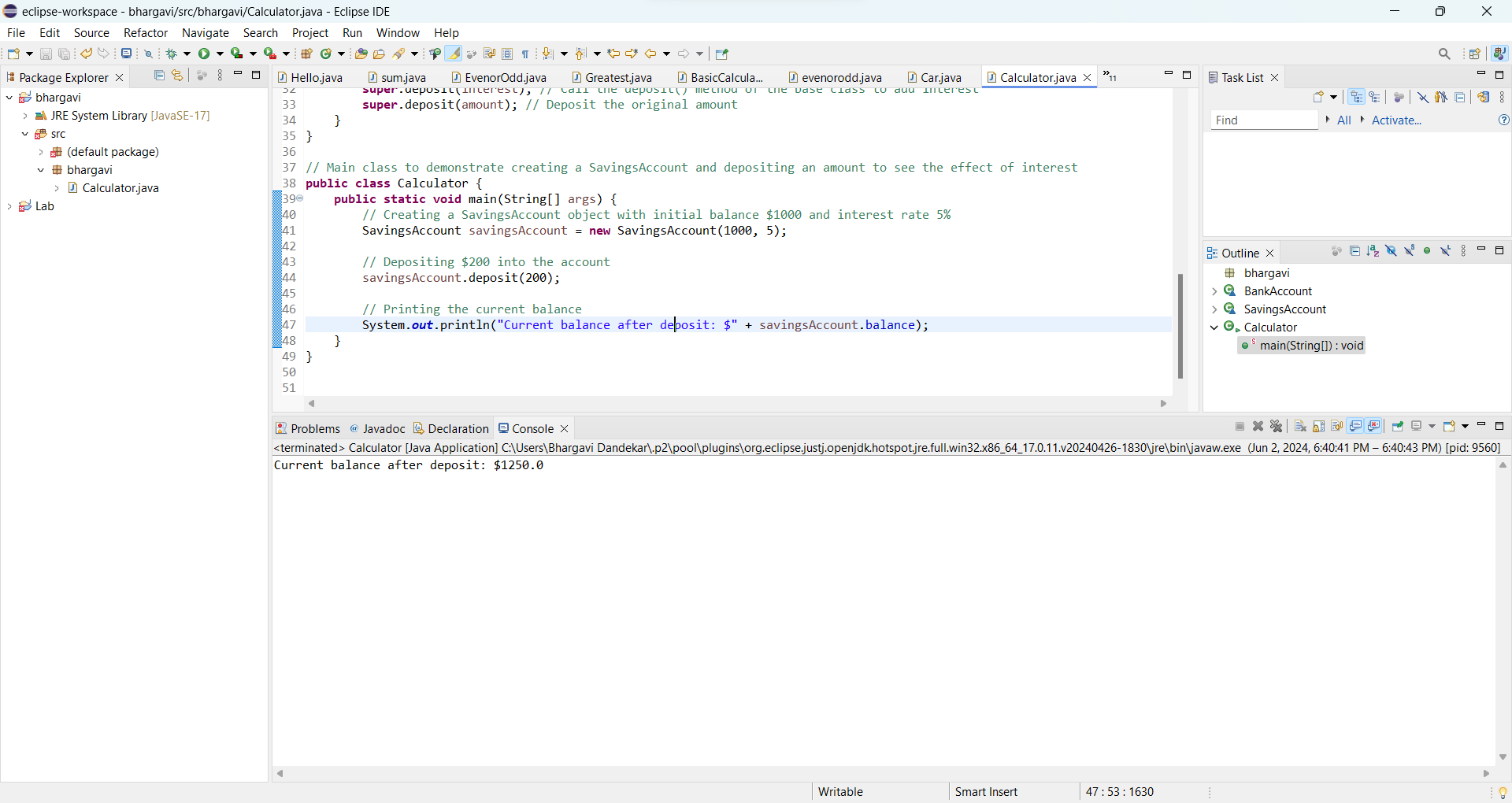
// Printing the current balance

System.***out***.println("Current balance after deposit: $" + 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.

**CODE:**

**package** bhargavi;

// Base class Employee

**class** Employee {

String name;

**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 {

String department;

// Constructor

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

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

**this**.department = department;

}

// Method to display details including department

@Override

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

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

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

}

}

// Main class

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

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

// Creating an Employee object

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

// Calling displayDetails() method of Employee

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

employee.displayDetails();

// Creating a Manager object

Manager manager = **new** Manager("Alice", 70000, "HR");

// Calling displayDetails() method of Manager

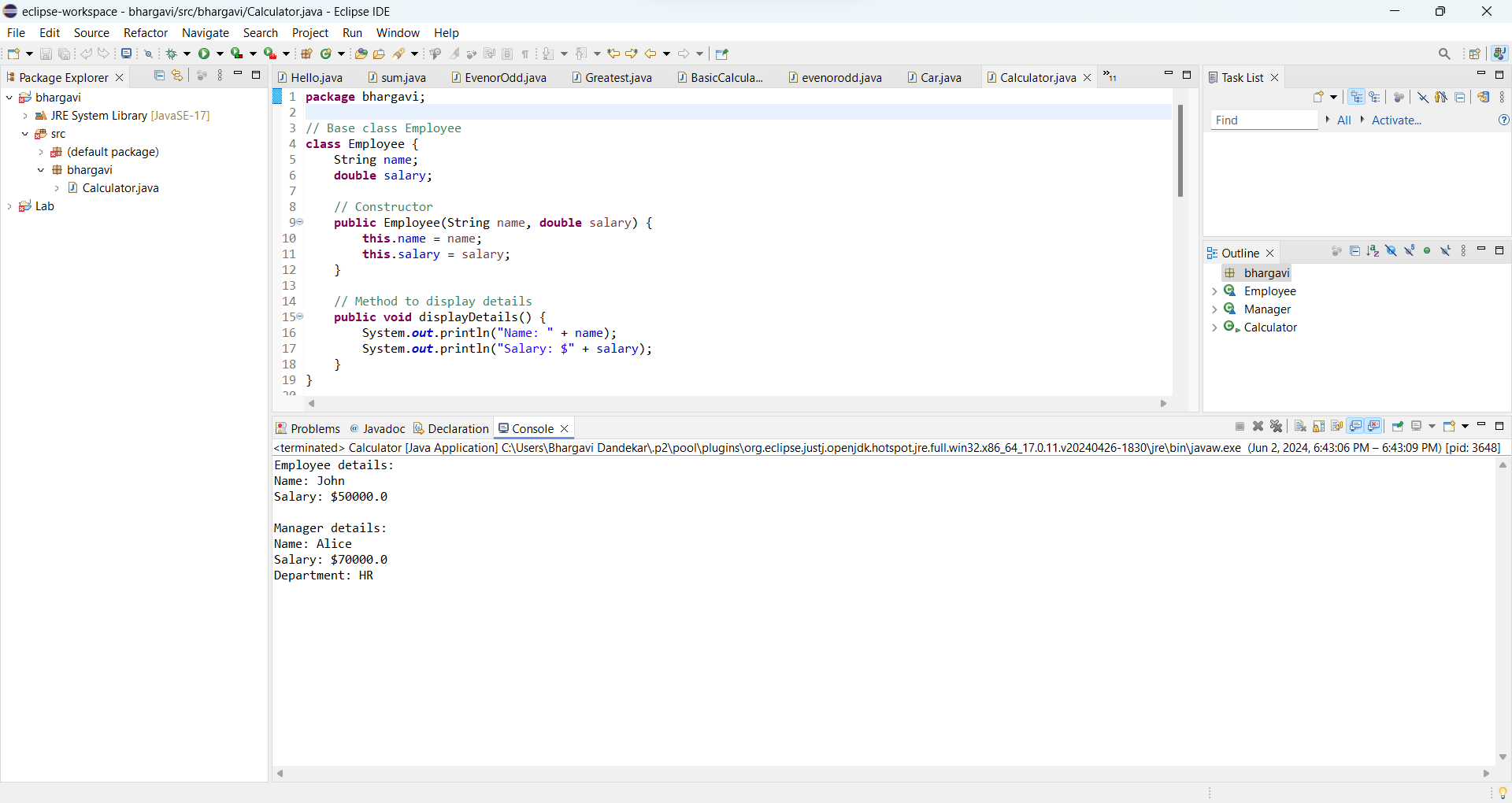
System.***out***.println("\nManager details:");

manager.displayDetails();

}

}

**OUTPUT:**

****

* We have a base class Employee with properties name and salary, and a method displayDetails() to display these details.
* We have a subclass Manager that inherits from Employee and adds a property department.
* The Manager class overrides the displayDetails() method to include department details by calling the displayDetails() method of the base class using the super keyword.
* In the main method, we demonstrate creating objects of Employee and Manager classes and calling the displayDetails() method to show the details**.**

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

**INPUT:**

**package** bhargavi;

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

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

// Constructor to initialize the object with value "Hi"

**public** Calculator() {

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

}

// Method to retrieve the value

**public** String getValue() {

**return** value;

}

// Main method for testing

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

// Creating an ImmutableExample object

Calculator example = **new** Calculator();

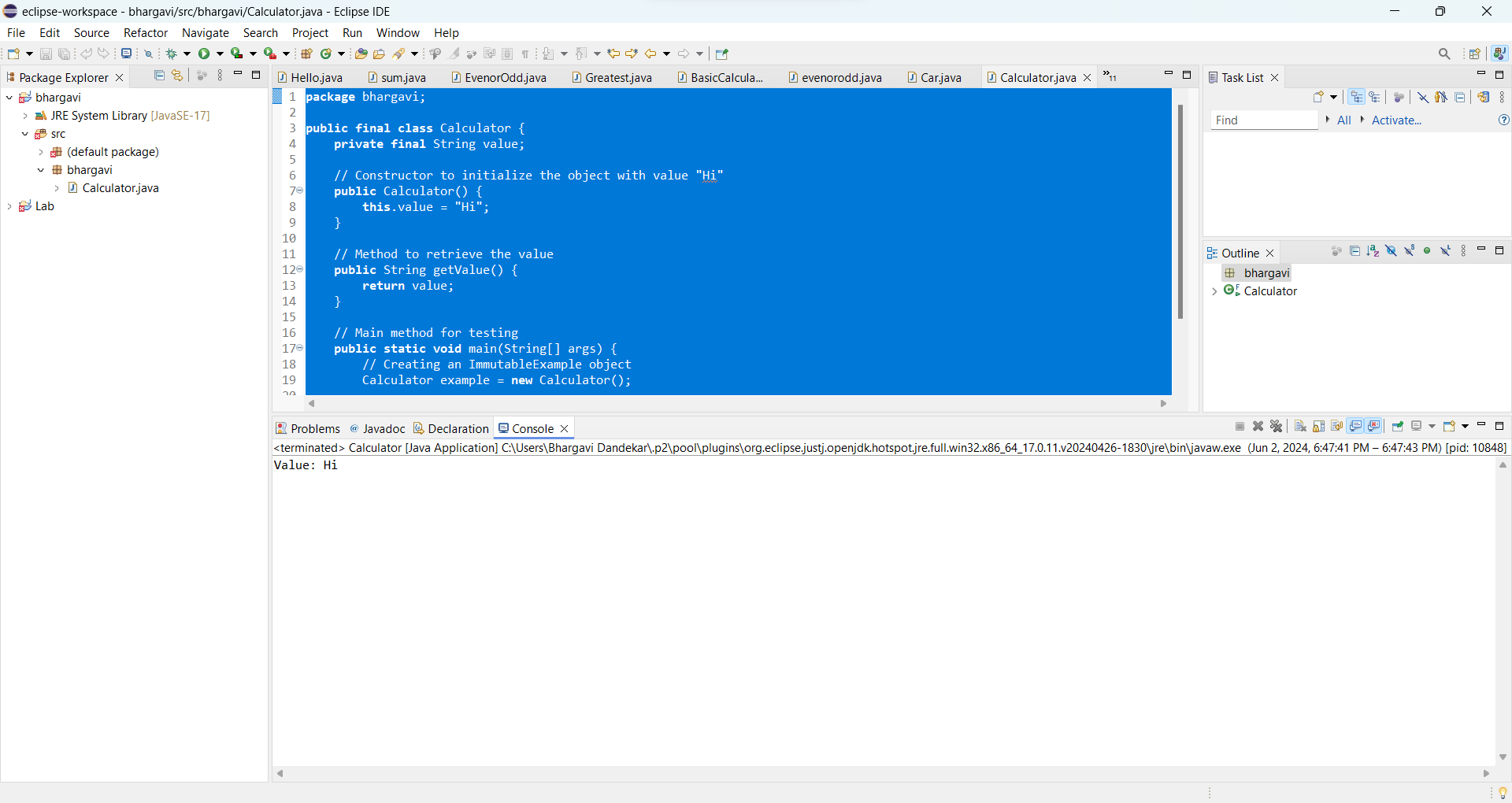
// Retrieving and printing the value

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

}

}

**OUTPUT:**

****

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

**CODE:**

**package** bhargavi;

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

**private** String value;

// Constructor to initialize the object with a value

**public** Calculator(String value) {

**this**.value = value;

}

// Method to retrieve the value

**public** String getValue() {

**return** value;

}

// Method to set the value

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

**this**.value = value;

}

// Main method for testing

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

// Creating a MutableExample object with initial value "hello 2"

Calculator example1 = **new** Calculator("hello 2");

// Retrieving and printing the value

System.***out***.println("Value 1: " + example1.getValue());

// Creating another MutableExample object with initial value "hello3"

Calculator example2 = **new** Calculator("hello3");

// Retrieving and printing the value

System.***out***.println("Value 2: " + example2.getValue());

// Modifying the value of example1

example1.setValue("hello3");

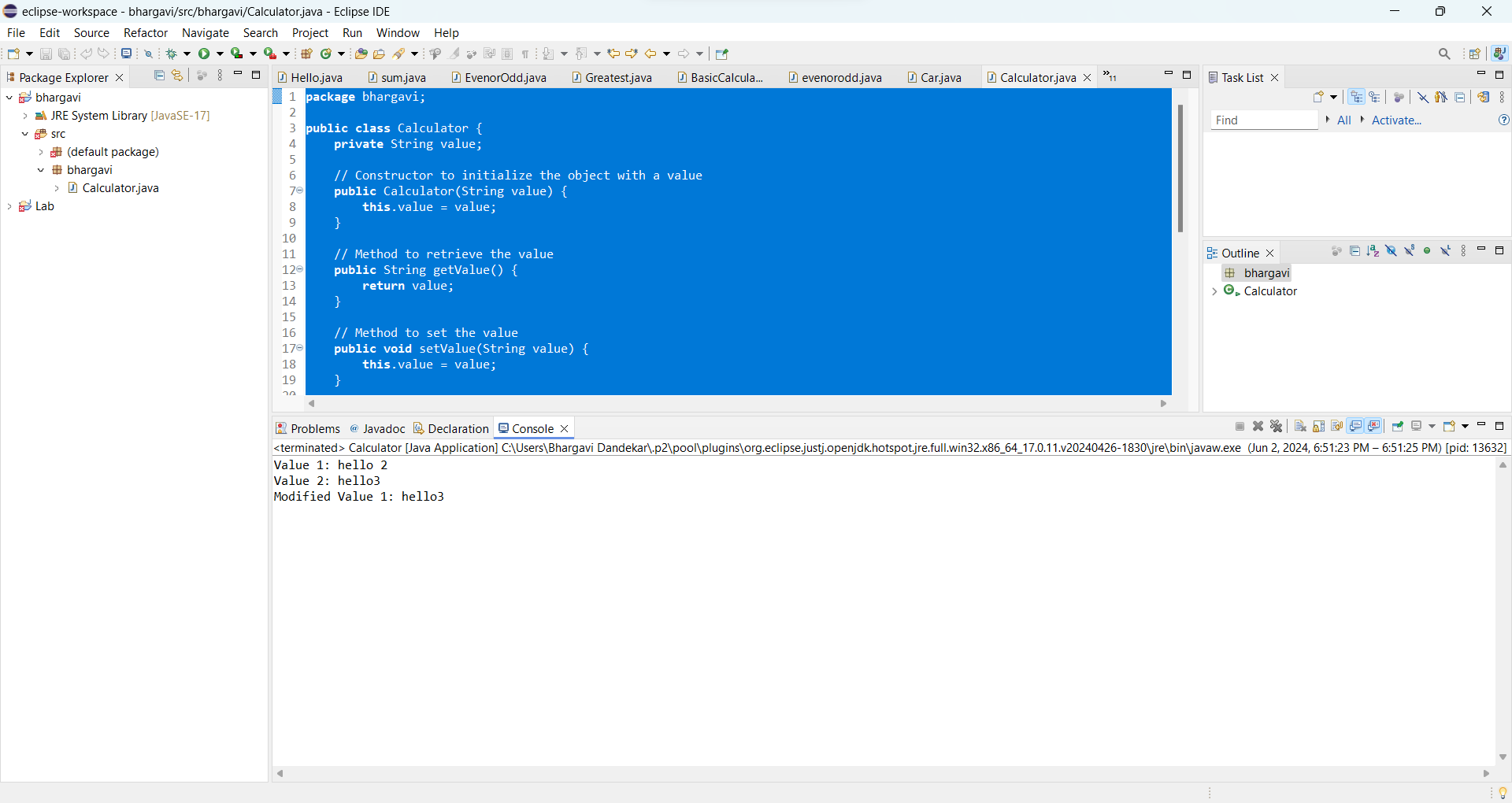
// Retrieving and printing the modified value

System.***out***.println("Modified Value 1: " + example1.getValue());

}

}

**OUTPUT:**

****

* The class MutableExample has a field value to store the object value.
* A constructor is provided to initialize the object with a value.
* Getter and setter methods (getValue() and setValue()) are provided to retrieve and modify the value of the object after creation.
* In the main method, two MutableExample objects are created with initial values "hello 2" and "hello3". The value of the first object is modified to "hello3", and the modified value is printed.
* This class allows for mutability, so you can modify the object values as needed.

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

**CODE:**

**package** bhargavi;

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

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

// Define a sample string

String str = "Hello World";

// Method: replace

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

// Method: contains

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

// Method: replaceAll

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

// Method: indexOf

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

// Method: substring

System.***out***.println("Substring from index 3: " + str.substring(3));

// Method: equals

String str2 = "Hello World";

System.***out***.println("Equals to 'Hello World': " + str.equals(str2));

// Method: lastIndexOf

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

// Method: startsWith

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

// Method: endsWith

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

// Method: equalsIgnoreCase

String str3 = "hello world";

System.***out***.println("EqualsIgnoreCase to 'hello world': " + str.equalsIgnoreCase(str3));

// Method: toLowerCase

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

// Method: toUpperCase

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

// Method: isEmpty

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

// Method: length

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

// Method: split

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

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

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

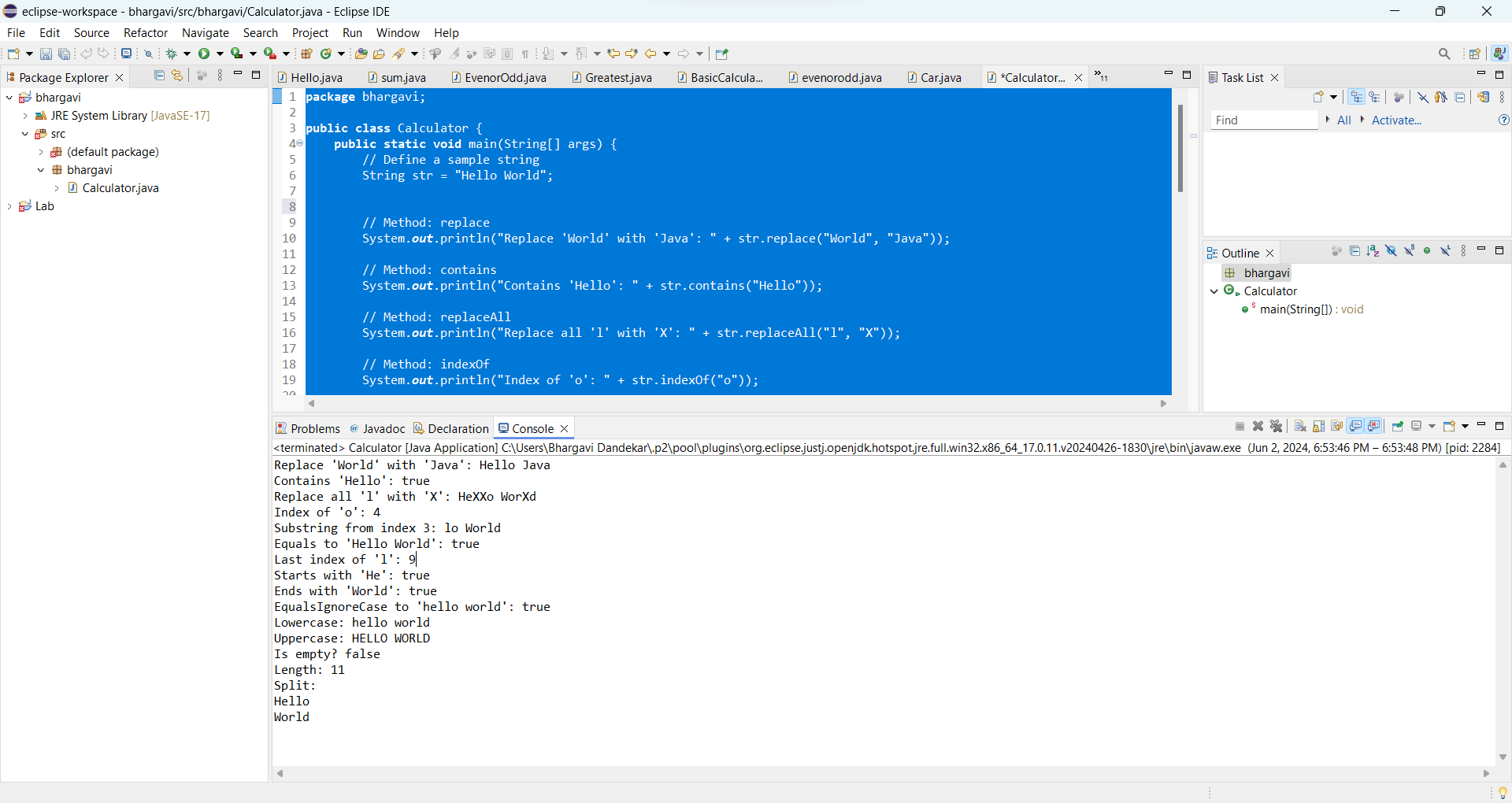
System.***out***.println(s);

}

}

}

**OUTPUT:**

****

This class demonstrates the usage of various string methods:

* replace
* contains
* replaceAll
* indexOf
* substring
* equals
* lastIndexOf
* startsWith
* endsWith
* equalsIgnoreCase
* toLowerCase
* toUpperCase
* isEmpty
* length
* split

Each method is applied to a sample string "Hello World", and the results are printed.