WEEK 1 : COGNIZANT DN 4.0 FSE HANDS-ON EXERCISES

**Exercise 1: Implementing the Singleton Pattern**

**Logger.java**

package singleton;

public class Logger

{

private static Logger instance;

private Logger()

{

System.out.println("Logger initialized...");

}

public static Logger getInstance()

{

if (instance == null)

{

instance = new Logger();

}

return instance;

}

public void log(String message)

{

System.out.println("[LOG] " + message);

}

}

**LoggerTest.java**

package singleton;

public class LoggerTest

{

public static void main(String[] args)

{

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

logger1.log("First log message");

logger2.log("Second log message");

if (logger1 == logger2)

{

System.***out***.println("Both logger instances are the same (Singleton works!)");

} else

{

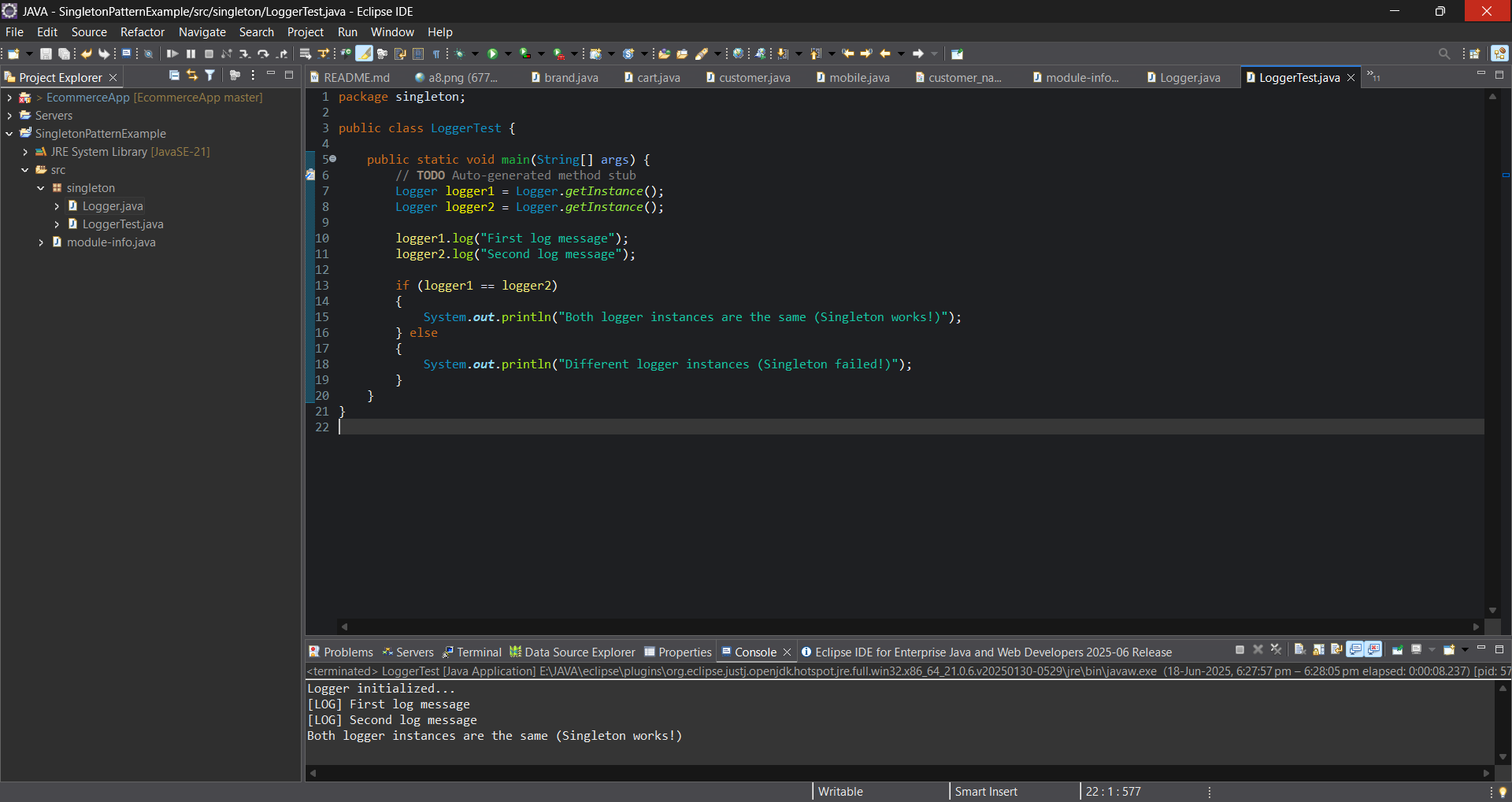
System.***out***.println("Different logger instances (Singleton failed!)");

}

}

}

**Output**



**Exercise 2: Implementing the Factory Method Pattern**

**#Creating Interface**

**Document**

public interface Document {

void open();

}

**#Creating Concrete Document Classes**

**WordDocument.java**

public class WordDocument implements Document

{

@Override

public void open()

{

System.out.println("Opening a Word document.");

}

}

**PdfDocument.java**

public class PdfDocument implements Document

{

@Override

public void open()

{

System.out.println("Opening a PDF document.");

}

}

**ExcelDocument.java**

public class ExcelDocument implements Document

{

@Override

public void open()

{

System.out.println("Opening an Excel document.");

}

}

**#Creating Abstract Factory Class**

**DocumentFactory.java**

public abstract class DocumentFactory

{

public abstract Document createDocument();

}

**#Creating Concrete Factories**

**WordFactory.java**

public class WordFactory extends DocumentFactory

{

@Override

public Document createDocument()

{

return new WordDocument();

}

}

**PdfFactory.java**

public class PdfFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

**ExcelFactory.java**

public class ExcelFactory extends DocumentFactory

{

@Override

public Document createDocument()

{

return new ExcelDocument();

}

}

**#Creating Test Class**

**Main.java**

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelFactory();

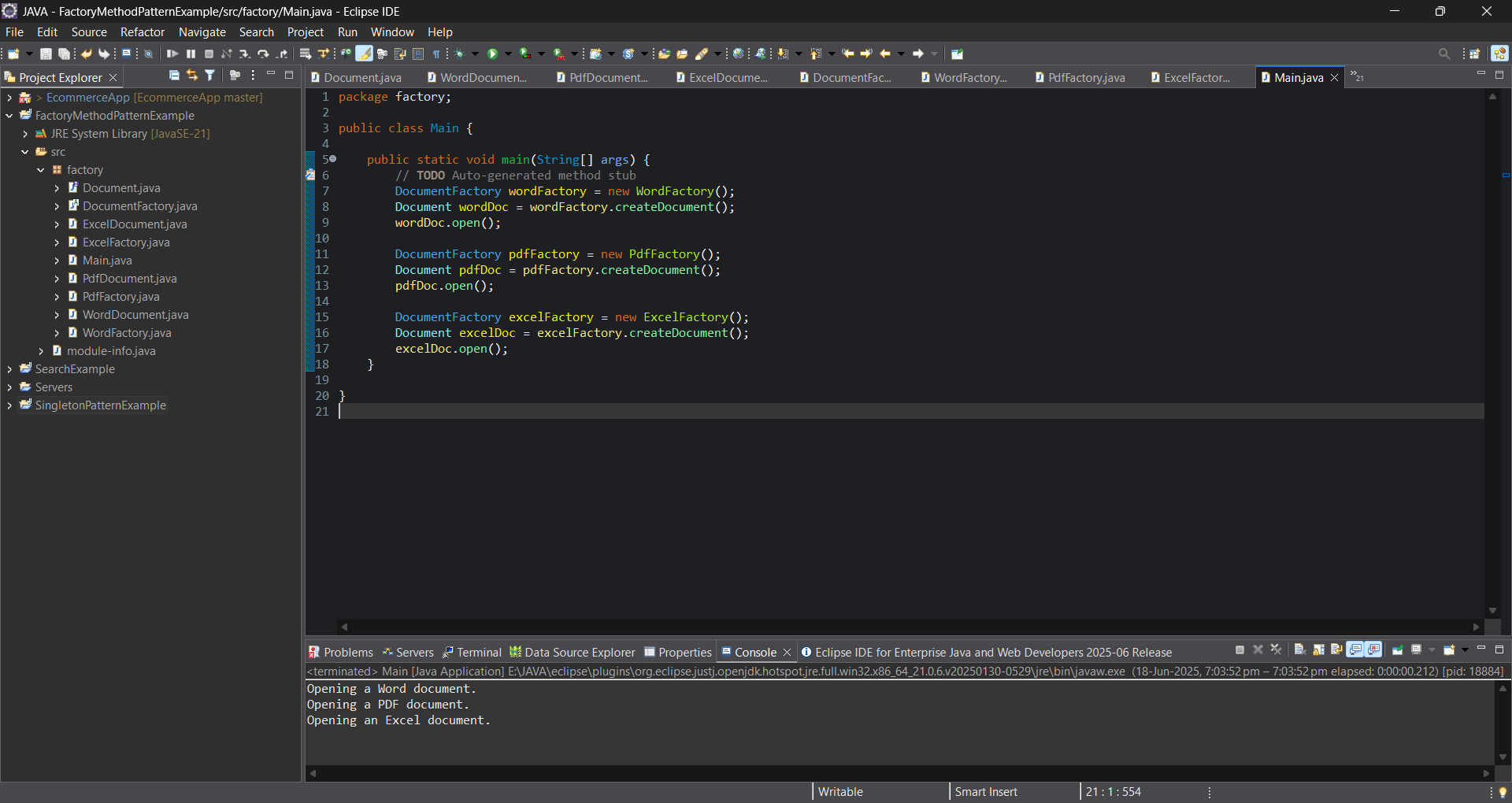
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output**

****

**Exercise 3: E-commerce Platform Search Function**

**Product.java**

package search;

public class Product

{

private int productId;

private String productName;

private String category;

public Product(int productId, String productName, String category)

{

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public String getCategory() {

return category;

}

}

**SearchEngine.java**

package search;

import java.util.Arrays;

public class SearchEngine

{

public static Product linearSearch(Product[] products, String name)

{

for (Product p : products)

{

if (p.getProductName().equalsIgnoreCase(name))

{

return p;

}

}

return null;

}

public static Product binarySearch(Product[] products, String name)

{

Arrays.sort(products, (a, b) -> a.getProductName().compareToIgnoreCase(b.getProductName()));

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int result = products[mid].getProductName().compareToIgnoreCase(name);

if (result == 0)

{

return products[mid];

}

else if (result < 0)

{

low = mid + 1;

}

else

{

high = mid - 1;

}

}

return null;

}

}

**SearchTest.java**

package search;

public class SearchTest

{

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Footwear"),

new Product(3, "Watch", "Accessories"),

new Product(4, "Mobile", "Electronics"),

new Product(5, "T-shirt", "Clothing")

};

String searchTerm = "Mobile";

System.out.println("Using Linear Search:");

Product result1 = SearchEngine.linearSearch(products, searchTerm);

printResult(result1);

System.out.println("\nUsing Binary Search:");

Product result2 = SearchEngine.binarySearch(products, searchTerm);

printResult(result2);

}

public static void printResult(Product product)

{

if (product != null)

{

System.out.println("Product Found: " + product.getProductName() + " in " + product.getCategory());

}

else

{

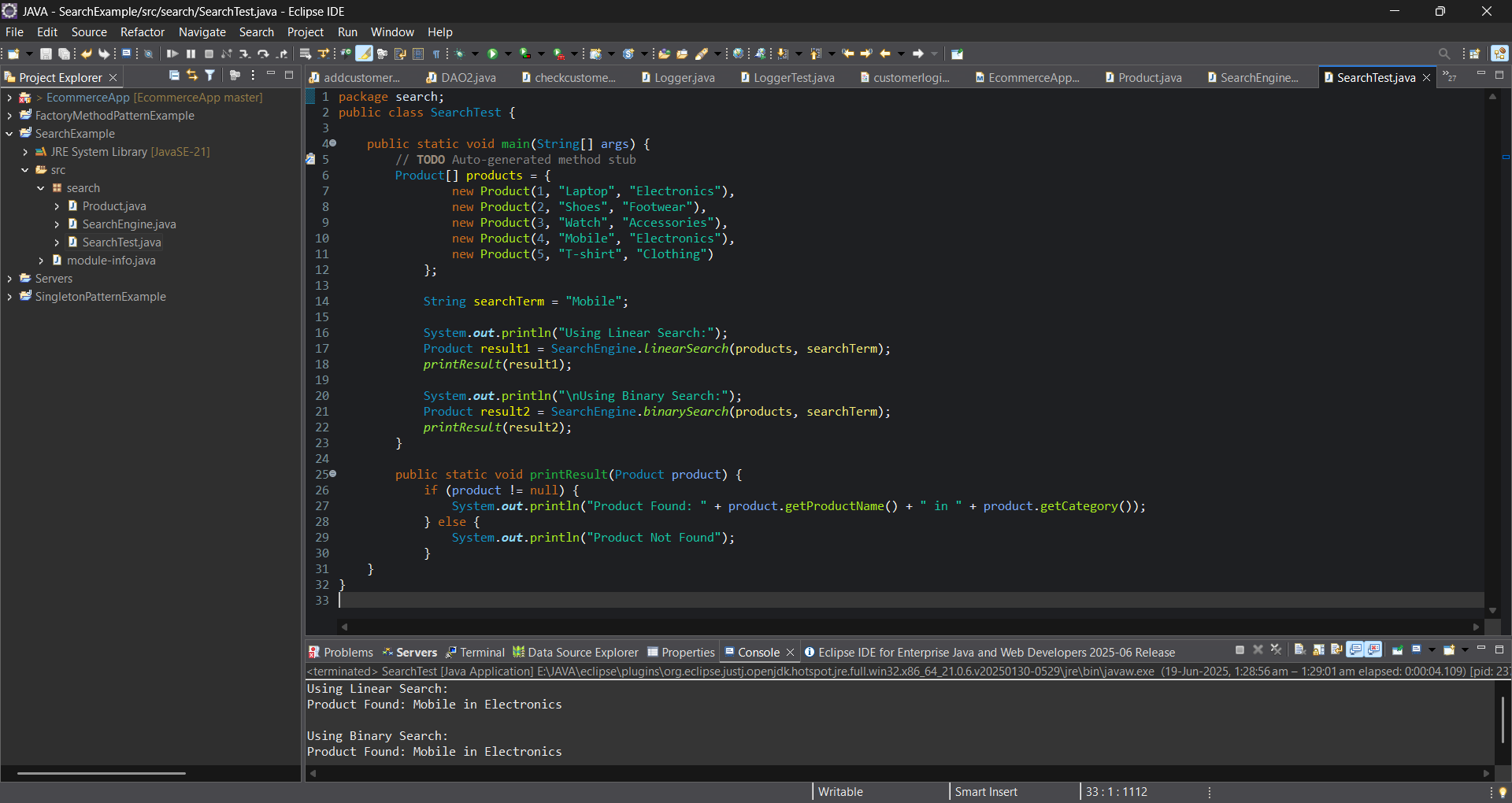
System.out.println("Product Not Found");

}

}

}

**Output**

****

**Exercise 4: Financial Forecasting**

**Forecast.java**

package forecast;

public class Forecast

{

public static double calculateFutureValue(double currentValue, double growthRate, int years) {

if (years == 0)

{

return currentValue;

}

else

{

return *calculateFutureValue*(currentValue \* (1 + growthRate), growthRate, years - 1);

}

}

public static void main(String[] args)

{

double initialValue = 10000.0;

double growthRate = 0.1;

int years = 5;

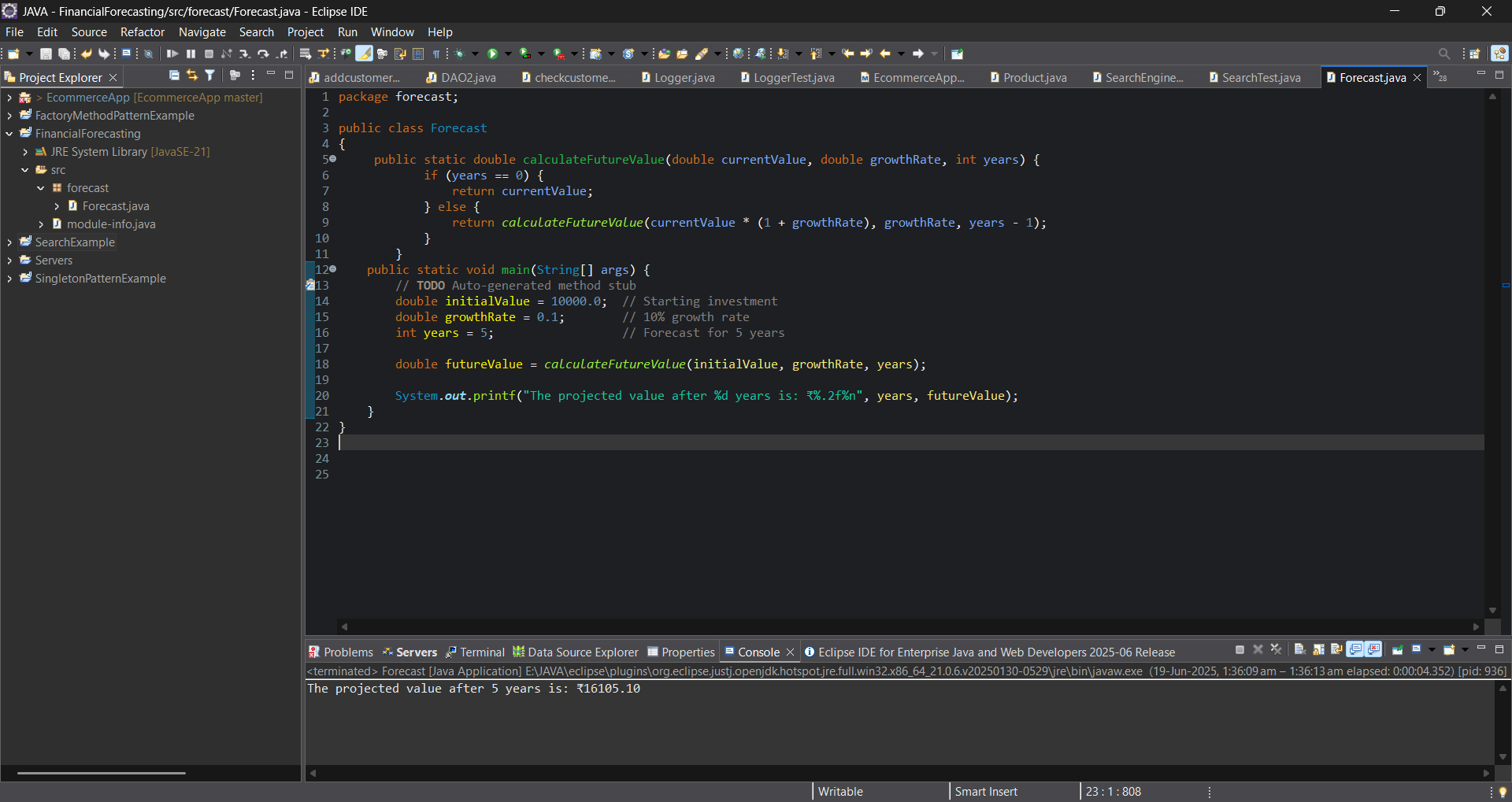
double futureValue = *calculateFutureValue*(initialValue, growthRate, years);

System.out.printf("The projected value after %d years is: ₹%.2f%n", years, futureValue);

}

}

**Output**

****

**Exercise 5: Implementing the MVC Pattern**

**#Under model package  
Student.java**

package model;

public class Student {

private String id;

private String name;

private String grade;

public Student(String id, String name, String grade) {

this.id = id;

this.name = name;

this.grade = grade;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getGrade() {

return grade;

}

public void setGrade(String grade) {

this.grade = grade;

}

}

**#Under view package  
StudentView.java**

package view;

import model.Student;

public class StudentView {

public void displayStudentDetails(Student student) {

System.*out*.println("---- Student Details ----");

System.*out*.println("ID : " + student.getId());

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

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

System.*out*.println("-------------------------\n");

}

}

**#Under Controller package**

**StudentController.java**

package controller;

import model.Student;

import view.StudentView;

public class StudentController {

private Student model;

private StudentView view;

public StudentController(Student model, StudentView view) {

this.model = model;

this.view = view;

}

public String getStudentName() {

return model.getName();

}

public String getStudentId() {

return model.getId();

}

public String getStudentGrade() {

return model.getGrade();

}

public void setStudentName(String name) {

model.setName(name);

}

public void setStudentId(String id) {

model.setId(id);

}

public void setStudentGrade(String grade) {

model.setGrade(grade);

}

public void updateView() {

view.displayStudentDetails(model);

}

}

**#Under Main package**

**Main.java**

package main;

import model.Student;

import view.StudentView;

import controller.StudentController;

public class Main {

public static void main(String[] args) {

Student student = new Student("S101", "Alice", "A");

StudentView view = new StudentView();

StudentController controller = new StudentController(student, view);

controller.updateView();

controller.setStudentName("Alice Johnson");

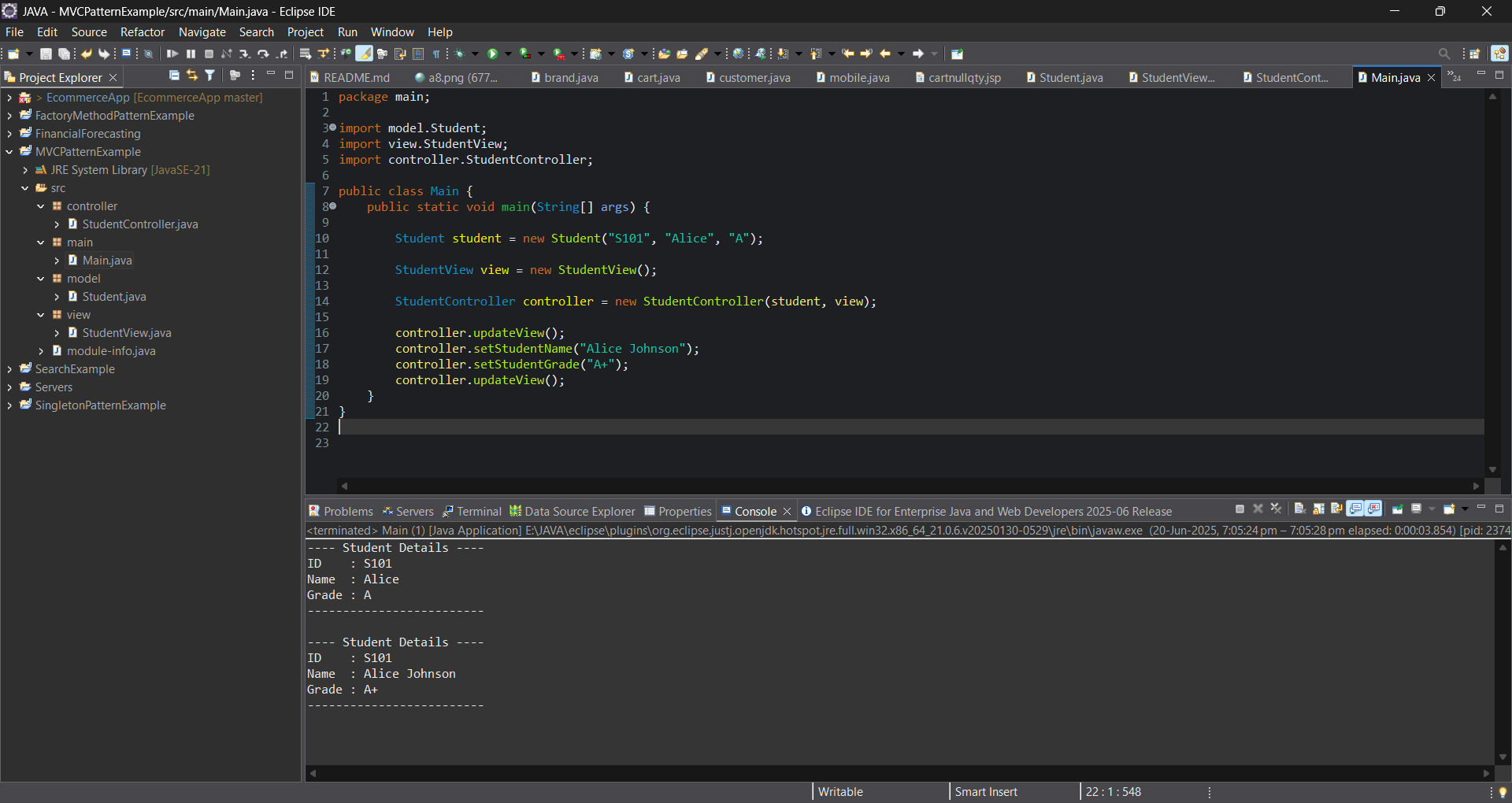
controller.setStudentGrade("A+");

controller.updateView();

}

}

**Output**

****