**Superset ID:** **6403222**

**MODULE 1 SOLUTIONS**

**Exercise 1: Implementing the Singleton Pattern**

import java.util.HashMap;

import java.util.Map;

// Singleton Class

class AppConfig {

// Step 1: Eager initialization of the single instance

private static final AppConfig instance = new AppConfig();

// Config values map

private Map<String, String> configValues;

// Private constructor to prevent external instantiation

private AppConfig() {

configValues = new HashMap<>();

// Simulate loading config values (could be from file, env, etc.)

configValues.put("appName", "MyApp");

configValues.put("version", "1.0.0");

configValues.put("environment", "production");

}

//Public method to provide global access

public static AppConfig getInstance() {

return instance;

}

// Step 5: Accessor method for config values

public String getValue(String key) {

return configValues.get(key);

}

}

public class Main {

public static void main(String[] args) {

// Access AppConfig Singleton

AppConfig config1 = AppConfig.getInstance();

AppConfig config2 = AppConfig.getInstance();

// Use the config values

System.out.println("App Name: " + config1.getValue("appName"));

System.out.println("Version: " + config1.getValue("version"));

System.out.println("Environment: " + config2.getValue("environment"));

// Check if both instances are same

if (config1 == config2) {

System.out.println("Both config instances are the same (Singleton confirmed).");

} else {

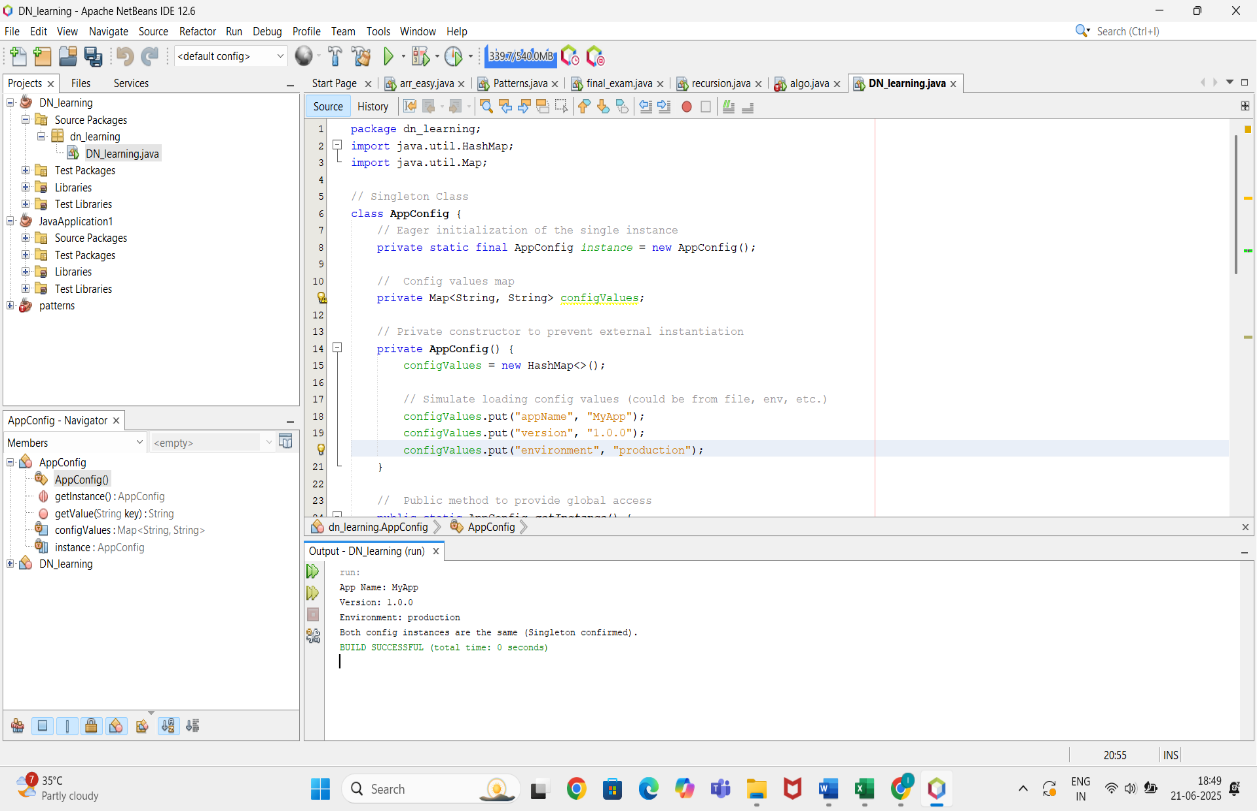
System.out.println("Different instances! (Singleton failed).");

}

}

}

OUTPUT:



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

// Notification interface

interface Notification {

void notifyUser();

}

// Concrete classes for different types of notifications

class SMSNotification implements Notification {

public void notifyUser() {

System.out.println("Sending an SMS Notification");

}

}

class EmailNotification implements Notification {

public void notifyUser() {

System.out.println("Sending an Email Notification");

}

}

class PushNotification implements Notification {

public void notifyUser() {

System.out.println("Sending a Push Notification");

}

}

// Factory class to generate Notification objects

class NotificationFactory {

public Notification createNotification(String type) {

if (type == null || type.isEmpty()) {

return null;

}

if (type.equalsIgnoreCase("SMS")) {

return new SMSNotification();

} else if (type.equalsIgnoreCase("EMAIL")) {

return new EmailNotification();

} else if (type.equalsIgnoreCase("PUSH")) {

return new PushNotification();

}

return null;

}

}

// Main class to test the Factory Method

public class Main {

public static void main(String[] args) {

NotificationFactory factory = new NotificationFactory();

// Try with different notification types

Notification notification1 = factory.createNotification("EMAIL");

notification1.notifyUser();

Notification notification2 = factory.createNotification("SMS");

notification2.notifyUser();

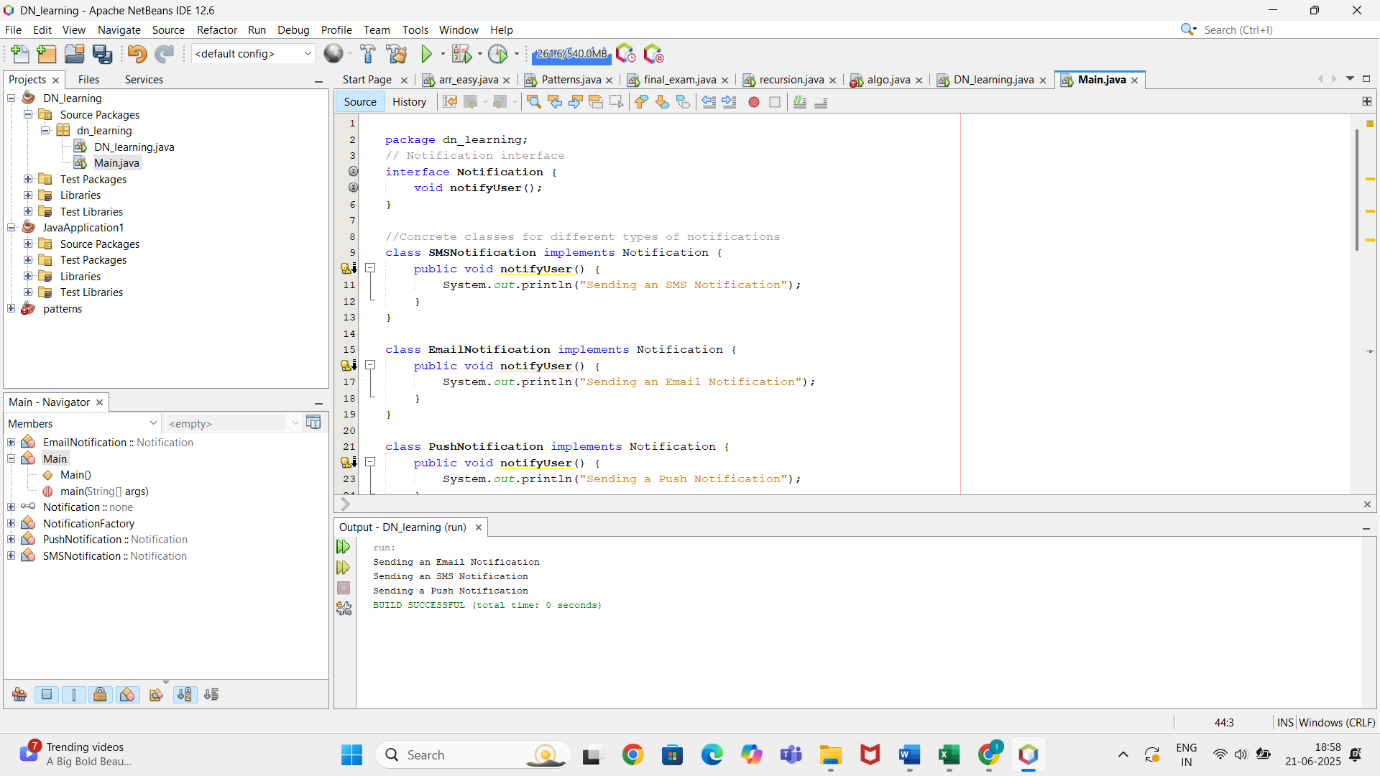
Notification notification3 = factory.createNotification("PUSH");

notification3.notifyUser();

}

}

OUTPUT:



**Exercise 2: E-commerce Platform Search Function**

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class SimpleProductSearch {

// Function to search products that start with the given prefix

public static List<String> searchProducts(List<String> products, String query) {

List<String> matchedProducts = new ArrayList<>();

for (String product : products) {

if (product.toLowerCase().startsWith(query.toLowerCase())) {

matchedProducts.add(product);

}

}

return matchedProducts;

}

public static void main(String[] args) {

// Sample product list

List<String> products = new ArrayList<>();

products.add("Laptop");

products.add("Laptop Stand");

products.add("Lap Desk");

products.add("Lamp");

products.add("Laser Printer");

products.add("Headphones");

products.add("Hat");

products.add("Hammer");

products.add("Hard Drive");

// User input

Scanner scanner = new Scanner(System.in);

System.out.print("Enter search term: ");

String searchQuery = scanner.nextLine();

// Search

List<String> results = searchProducts(products, searchQuery);

// Display results

if (results.isEmpty()) {

System.out.println("No products found.");

} else {

System.out.println("Matching products:");

for (String result : results) {

System.out.println("- " + result);

}

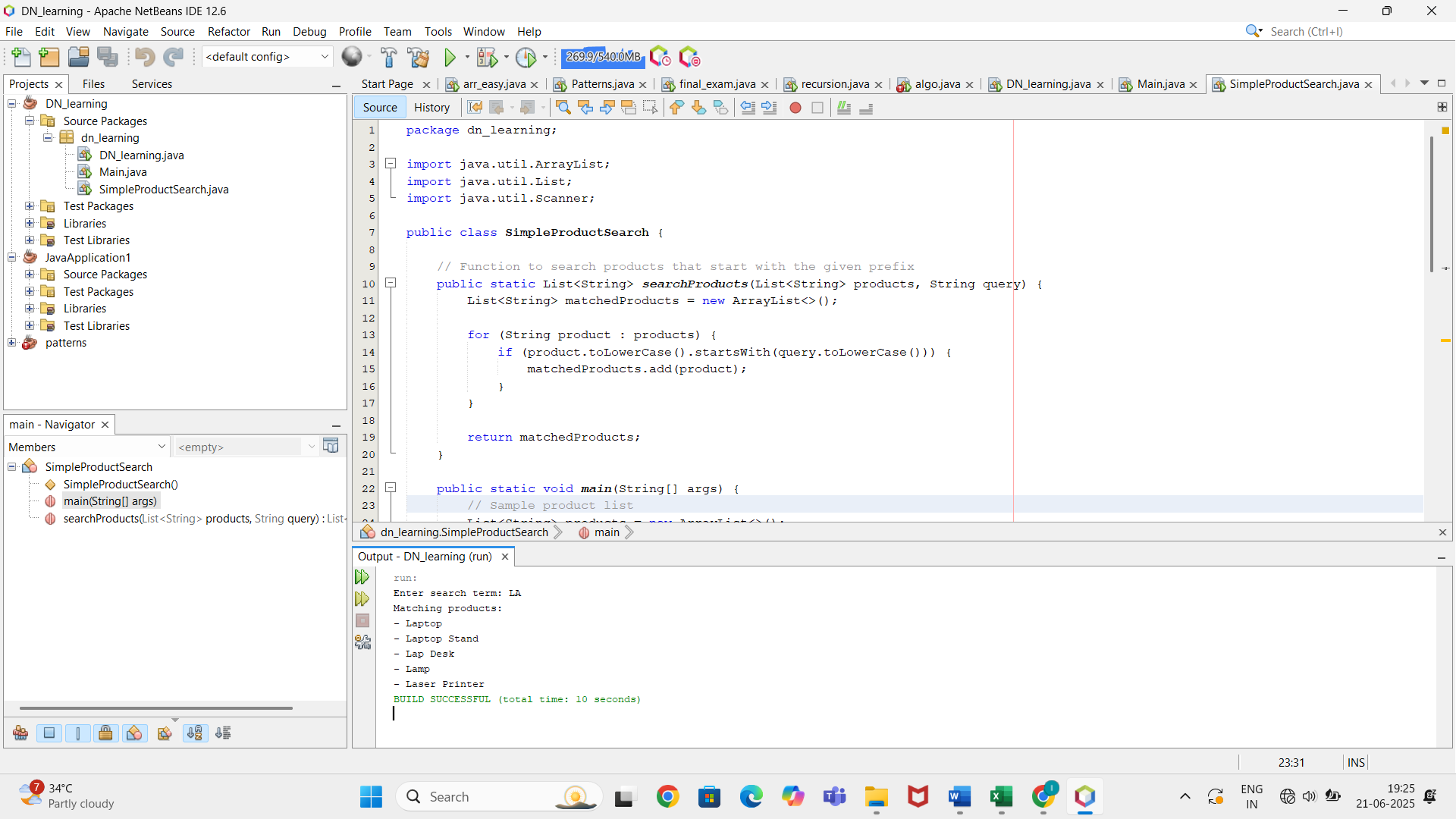
}

scanner.close();

}

}

OUTPUT:



**Exercise 7: Financial Forecasting**

import java.util.\*;

public class FinancialForecasting {

// Function to calculate average monthly growth rate

public static double calculateAverageGrowthRate(int[] data) {

double totalGrowth = 0.0;

for (int i = 1; i < data.length; i++) {

double growth = ((double) data[i] - data[i - 1]) / data[i - 1];

totalGrowth += growth;

}

return totalGrowth / (data.length - 1);

}

// Function to forecast future values

public static List<Double> forecastNextMonths(int[] pastData, int monthsToForecast) {

List<Double> forecast = new ArrayList<>();

double lastValue = pastData[pastData.length - 1];

double avgGrowthRate = calculateAverageGrowthRate(pastData);

for (int i = 0; i < monthsToForecast; i++) {

lastValue = lastValue \* (1 + avgGrowthRate); // applying growth

forecast.add(lastValue);

}

return forecast;

}

public static void main(String[] args) {

// Sample revenue for last 6 months

int[] pastRevenue = {10000, 12000, 13000, 15000, 17000, 20000};

// Forecast next 3 months

int forecastMonths = 3;

List<Double> forecastedRevenue = forecastNextMonths(pastRevenue, forecastMonths);

System.out.println("Forecasted Revenue for Next " + forecastMonths + " Months:");

for (int i = 0; i < forecastedRevenue.size(); i++) {

System.out.printf("Month %d: ₹%.2f\n", i + 1, forecastedRevenue.get(i));

}

}

}

OUTPUT:

