|  |  |
| --- | --- |
| A picture of a winding road and trees  Hands-On Exercises Submission  Module 1 – Design Patterns and Principles | Name: Rohini Vaiyathilingam  **Skill:** Design Principles & Patterns |

WEEK 1 : HANDS ON EXERCISES

Exercise 1: Implementing the Singleton Pattern

//CODE

import java.text.SimpleDateFormat;

import java.util.Date;

public class Logger {

private static volatile Logger instance = null;

private Logger() {

System.out.println("[System] Logger initialized at " + getCurrentTimestamp());

}

public static Logger getInstance() {

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}

}

return instance;

}

public void log(String message) {

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

}

private String getCurrentTimestamp() {

return new SimpleDateFormat("HH:mm:ss.SSS").format(new Date());

}

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("Singleton pattern is awesome!");

logger1.log("Logging a second message just to test.");

Logger logger2 = Logger.getInstance();

logger2.log("Logger still works with the same instance.");

logger2.log("Same instance? " + (logger1 == logger2));

Runnable task = () -> {

Logger logger = Logger.getInstance();

logger.log("Logging from thread " + Thread.currentThread().getName());

};

Thread t1 = new Thread(task, "Worker-1");

Thread t2 = new Thread(task, "Worker-2");

Thread t3 = new Thread(task, "Worker-3");

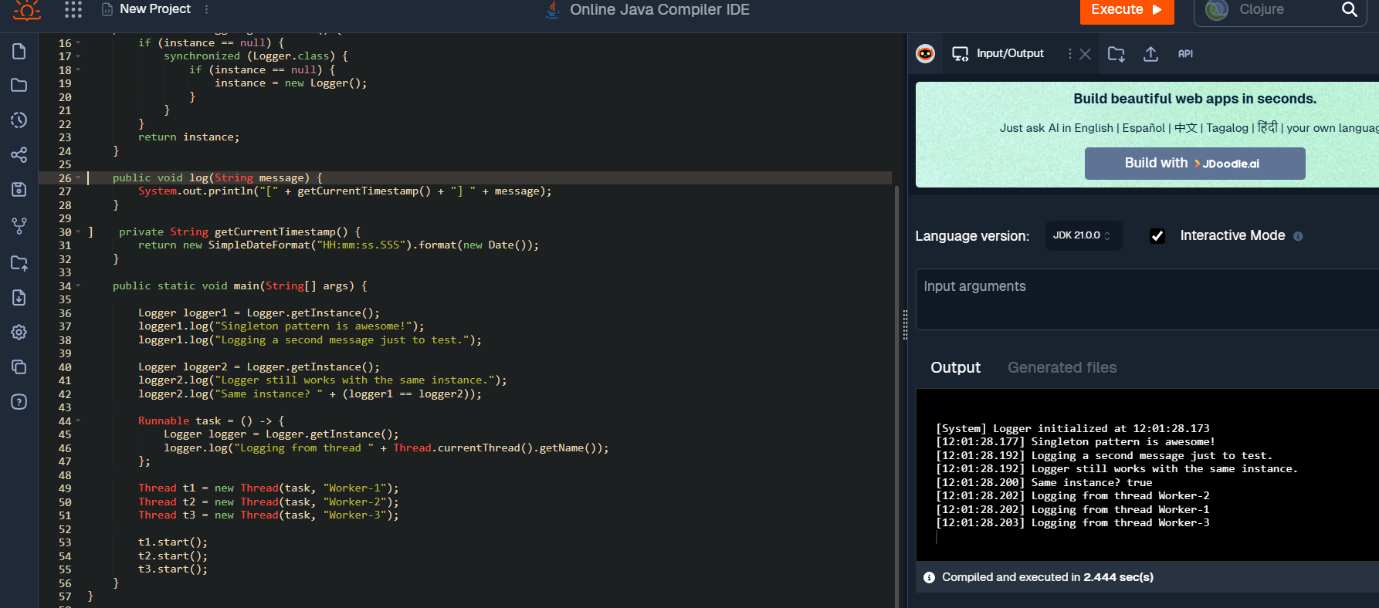
t1.start();

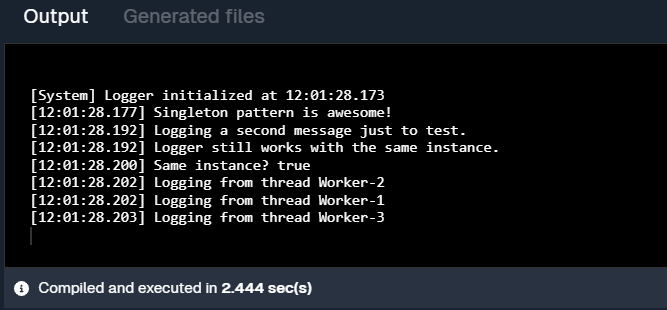
t2.start();

t3.start();

}

}





Exercise 2: Implementing the Factory Method Pattern

//Code

interface Notification {

void notifyUser();

}

class EmailNotification implements Notification {

public void notifyUser() {

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

}

}

class SMSNotification implements Notification {

public void notifyUser() {

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

}

}

class PushNotification implements Notification {

public void notifyUser() {

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

}

}

class NotificationFactory {

public static Notification createNotification(String type) {

if (type == null || type.isEmpty()) return null;

switch (type.toLowerCase()) {

case "email": return new EmailNotification();

case "sms": return new SMSNotification();

case "push": return new PushNotification();

default: return null;

}

}

}

public class NotificationService {

public static void main(String[] args) {

System.out.println("[System] Notification Service Started 🚀");

Notification notification1 = NotificationFactory.createNotification("email");

notification1.notifyUser();

Notification notification2 = NotificationFactory.createNotification("sms");

notification2.notifyUser();

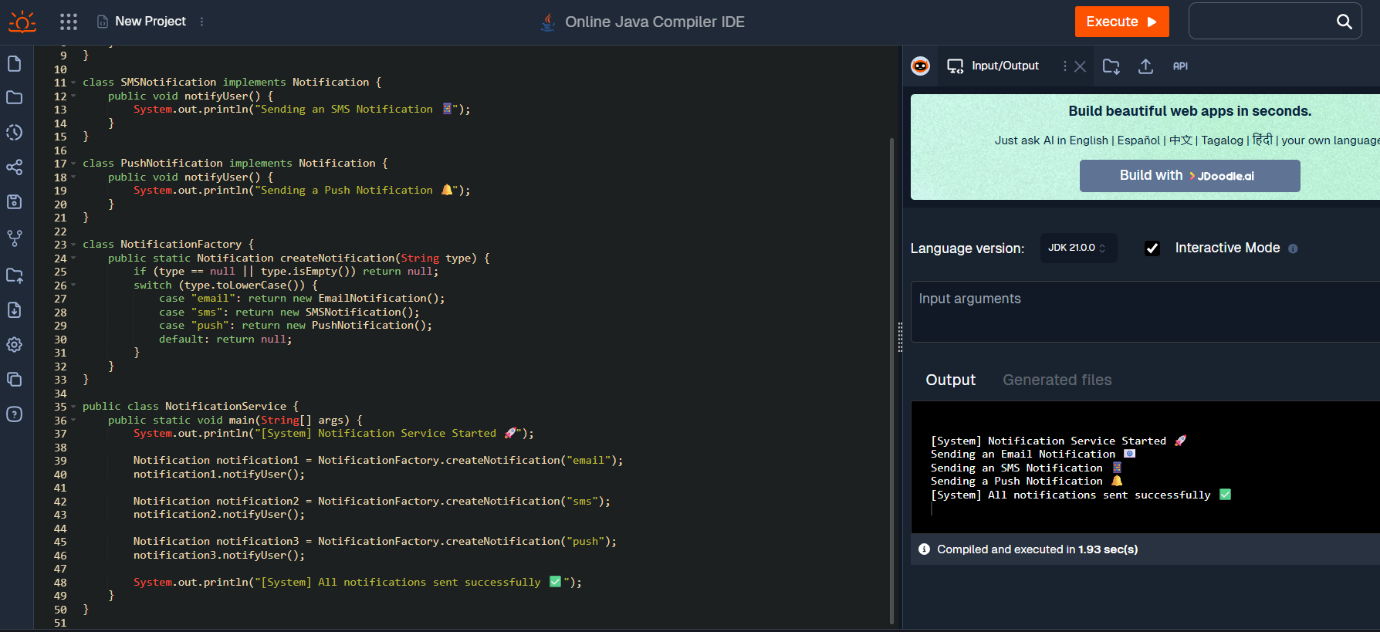
Notification notification3 = NotificationFactory.createNotification("push");

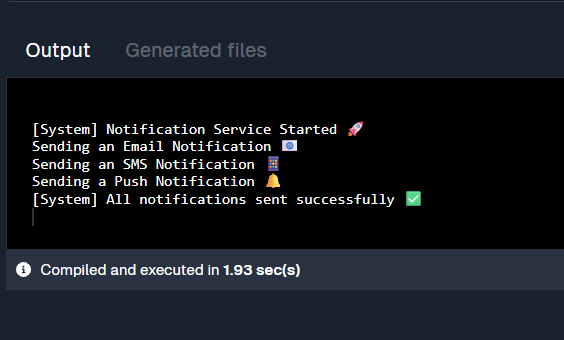
notification3.notifyUser();

System.out.println("[System] All notifications sent successfully ✅");

}

}





Exercise 2: E-commerce Platform Search Function

//Code

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class ProductSearch {

private static List<String> getProductList() {

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

products.add("iPhone 15 Pro");

products.add("Samsung Galaxy S24");

products.add("MacBook Air M2");

products.add("Dell XPS 13");

products.add("Sony WH-1000XM5 Headphones");

products.add("Apple Watch Series 9");

products.add("HP Pavilion Laptop");

return products;

}

private static List<String> searchProducts(String keyword, List<String> products) {

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

for (String product : products) {

if (product.toLowerCase().contains(keyword.toLowerCase())) {

results.add(product);

}

}

return results;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

List<String> productList = getProductList();

System.out.println("🛒 Welcome to the E-commerce Search Platform");

System.out.print("🔍 Enter a keyword to search for products: ");

String keyword = scanner.nextLine();

List<String> searchResults = searchProducts(keyword, productList);

System.out.println("\n🔎 Search Results:");

if (searchResults.isEmpty()) {

System.out.println("❌ No products found matching \"" + keyword + "\".");

} else {

for (String product : searchResults) {

System.out.println("✅ " + product);

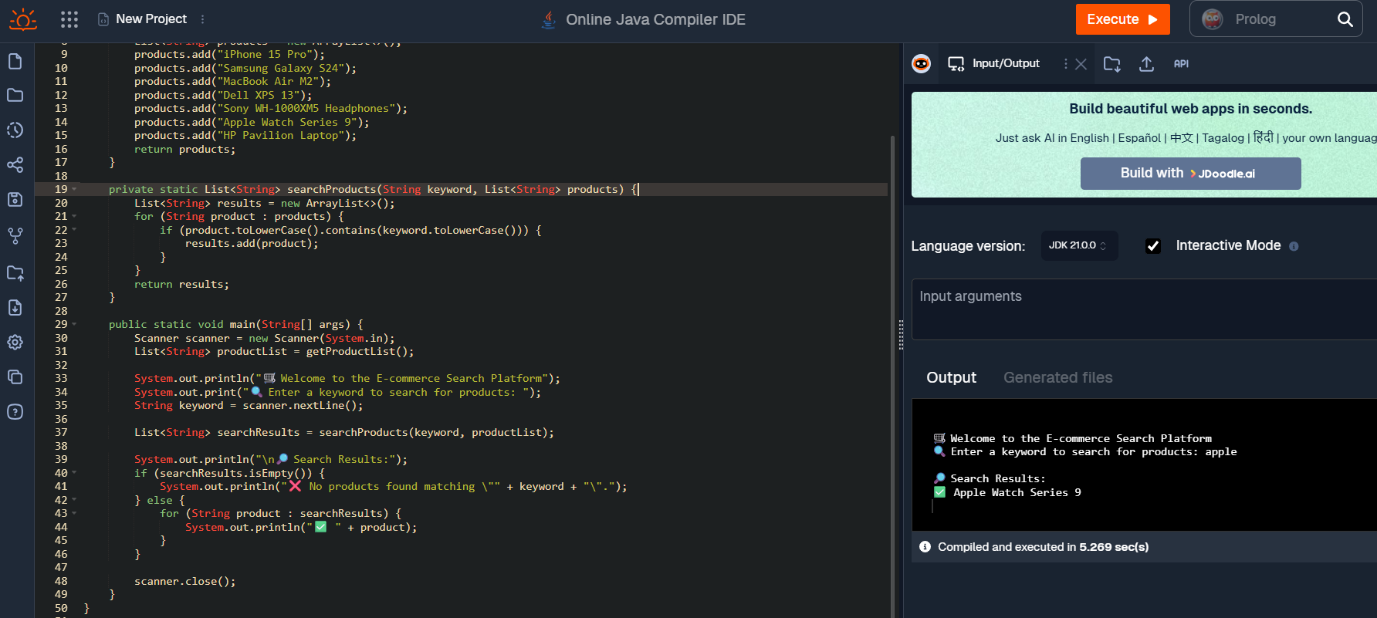
}

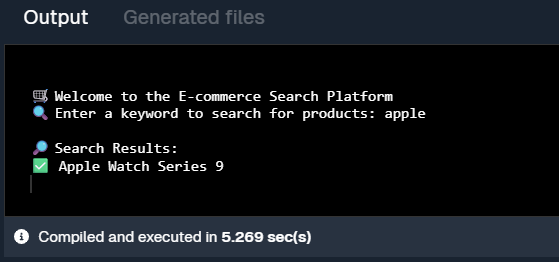
}

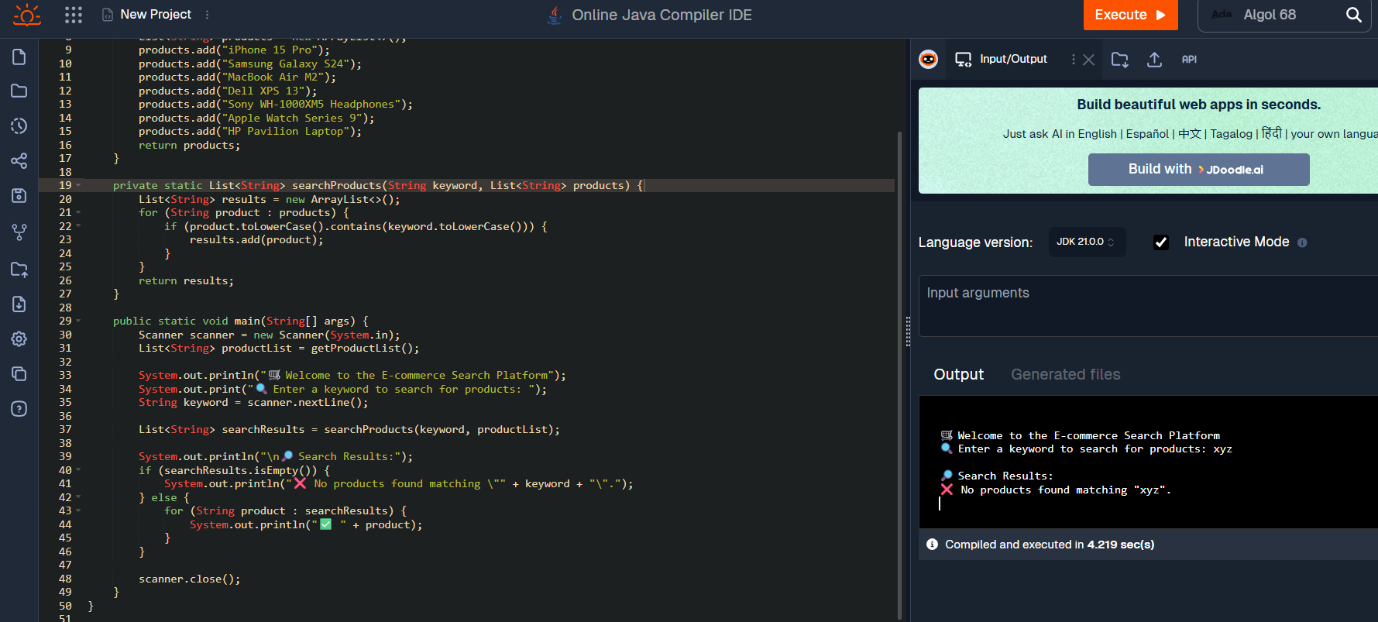
scanner.close();

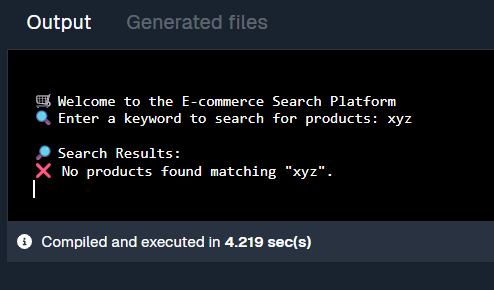
}

}









Exercise 7: Financial Forecasting

//Code

import java.util.Scanner;

public class FinancialForecast {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("📊 Financial Forecast Tool");

System.out.print("💼 Enter Initial Investment Amount: ₹");

double initialInvestment = sc.nextDouble();

System.out.print("💰 Enter Monthly Income: ₹");

double monthlyIncome = sc.nextDouble();

System.out.print("💸 Enter Monthly Expenses: ₹");

double monthlyExpenses = sc.nextDouble();

System.out.print("📆 Enter Number of Months to Forecast: ");

int months = sc.nextInt();

System.out.println("\n📈 Forecast Report:");

double currentBalance = initialInvestment;

for (int i = 1; i <= months; i++) {

currentBalance += (monthlyIncome - monthlyExpenses);

System.out.printf("Month %02d: ₹%.2f\n", i, currentBalance);

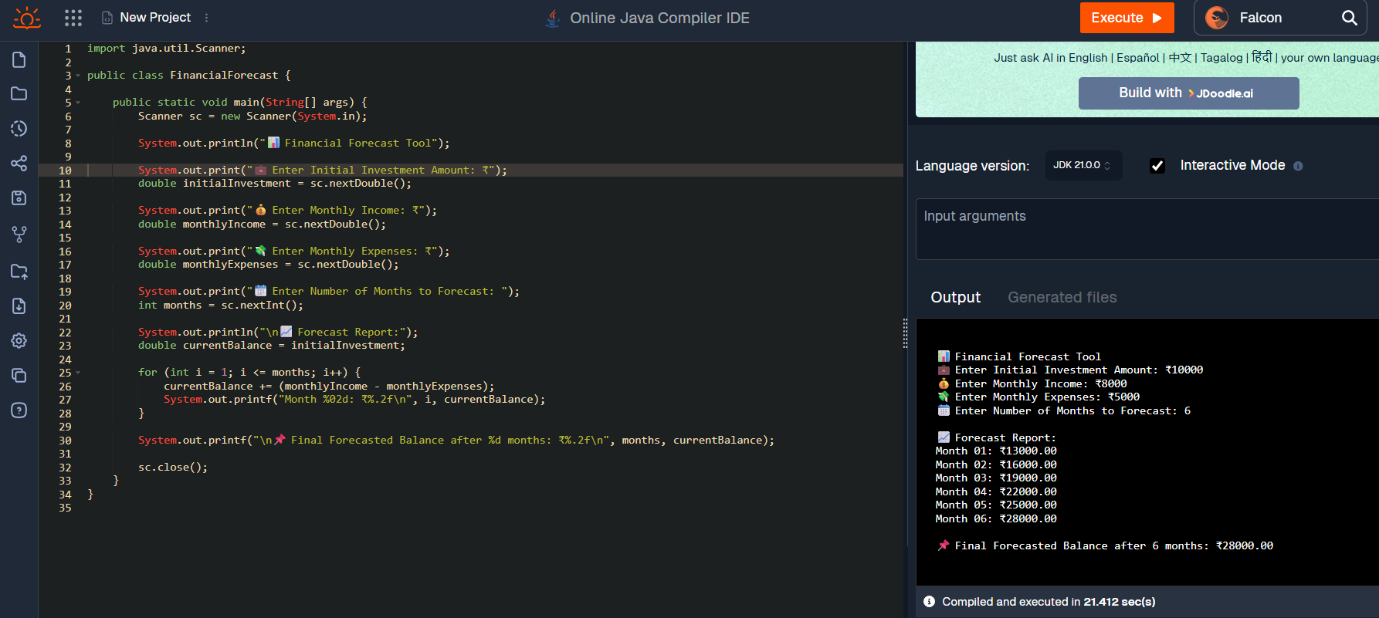
}

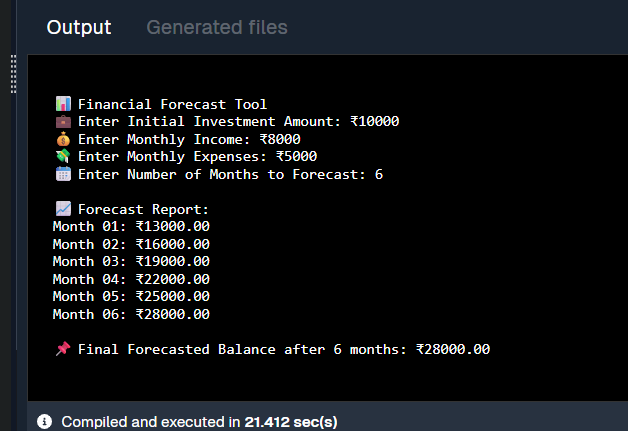
System.out.printf("\n📌 Final Forecasted Balance after %d months: ₹%.2f\n", months, currentBalance);

sc.close();

}

}





public class NotificationService {

public static void main(String[] args) {

System.out.println("[System] Notification Service Started 🚀");

Notification notification1 = NotificationFactory.createNotification("email");

notification1.notifyUser();

Notification notification2 = NotificationFactory.createNotification("sms");

notification2.notifyUser();

Notification notification3 = NotificationFactory.createNotification("push");

notification3.notifyUser();

System.out.println("[System] All notifications sent successfully ✅");

}

}