**WEEK 1**   
**Topic 1:** Design Patterns and Principles

**Exercise 1**Implementing the Singleton Pattern

**Code :**  
  
import java.util.HashMap;

import java.util.Map;

class ConfigurationManager {

private static ConfigurationManager instance;

private Map<String, String> config = new HashMap<>();

private ConfigurationManager() {

config.put("app.name", "MyApp");

config.put("app.version", "1.0.0");

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

}

public static synchronized ConfigurationManager getInstance() {

if (instance == null) {

instance = new ConfigurationManager();

}

return instance;

}

public String getProperty(String key) {

return config.get(key);

}

}

public class Main {

public static void main(String[] args) {

ConfigurationManager cm1 = ConfigurationManager.getInstance();

ConfigurationManager cm2 = ConfigurationManager.getInstance();

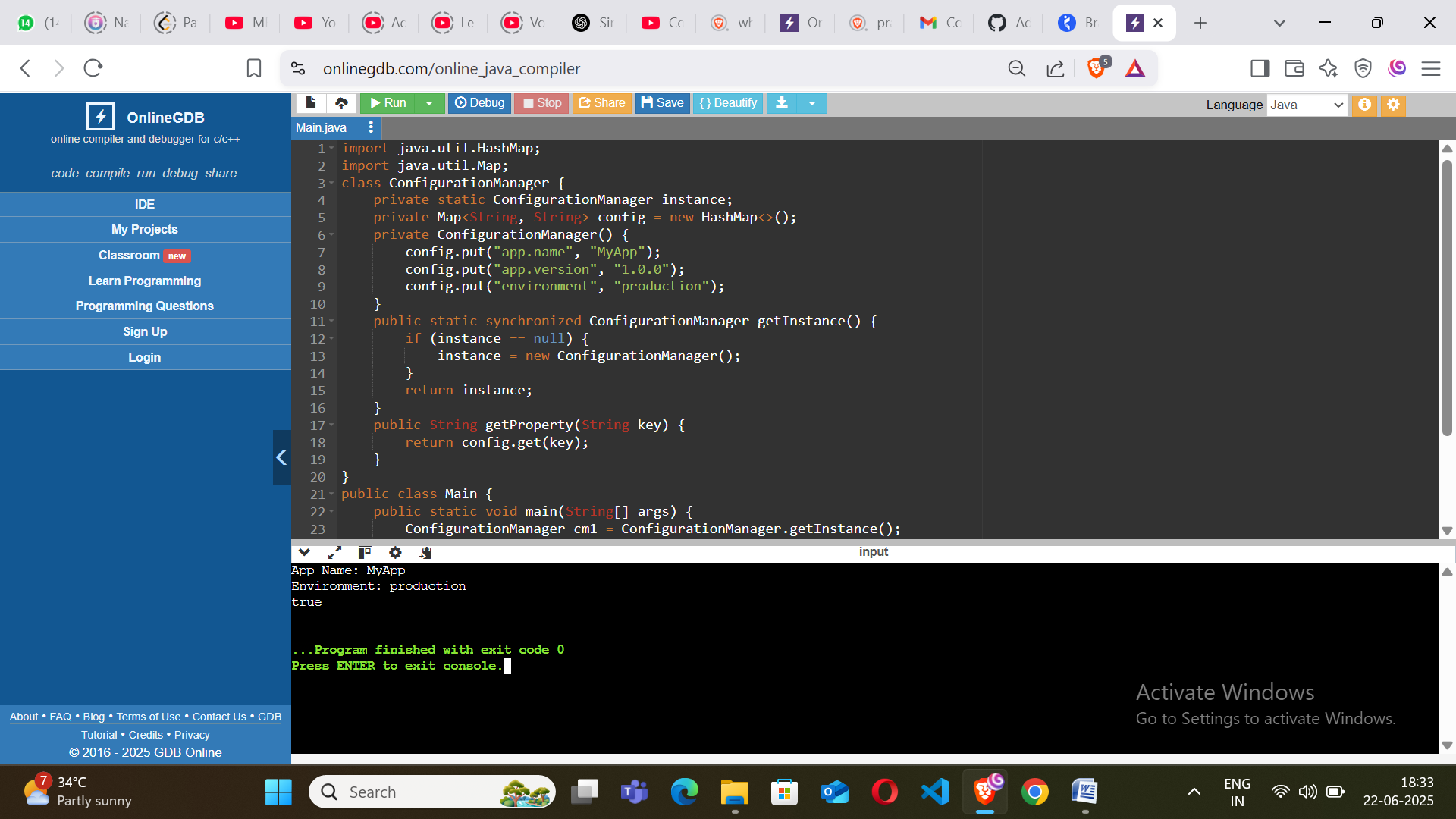
System.out.println("App Name: " + cm1.getProperty("app.name"));

System.out.println("Environment: " + cm2.getProperty("environment"));

System.out.println(cm1 == cm2);

}

}

**O/P :**   
  


**Exercise 2**Implementing the Factory Design Pattern

**Code :**

import java.util.\*;

interface Vehicle {

void start();

}

class Car implements Vehicle {

public void start() {

System.out.println("Car started with a key ignition.");

}

}

class Bike implements Vehicle {

public void start() {

System.out.println("Bike started with a kick start.");

}

}

class Truck implements Vehicle {

public void start() {

System.out.println("Truck started with a heavy-duty ignition.");

}

}

class VehicleFactory {

public static Vehicle getVehicle(String type) {

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

switch (type.toLowerCase()) {

case "car": return new Car();

case "bike": return new Bike();

case "truck": return new Truck();

default: throw new IllegalArgumentException("Unknown vehicle type: " + type);

}

}

}

public class Main {

public static void main(String[] args) {

Vehicle v1 = VehicleFactory.getVehicle("car");

v1.start();

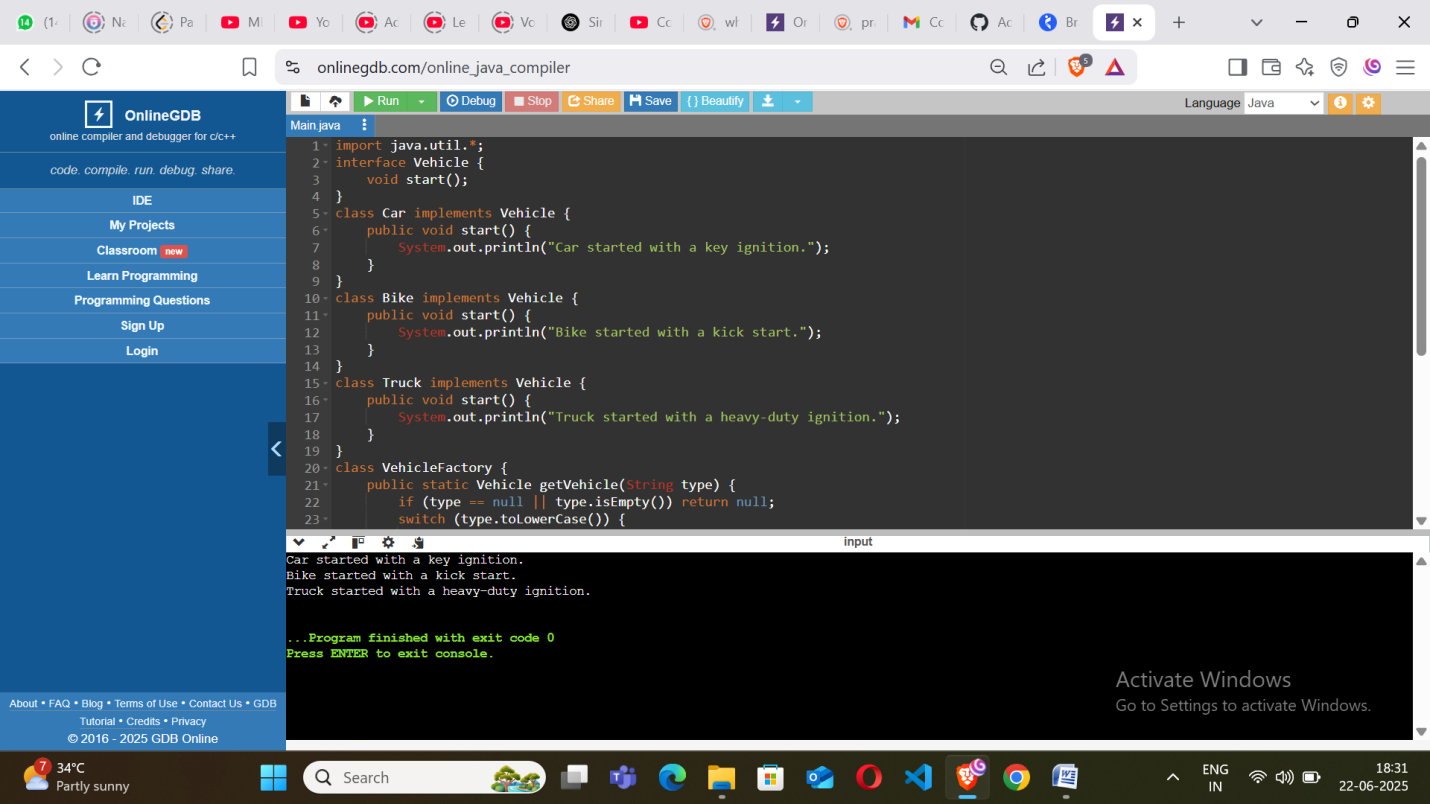
Vehicle v2 = VehicleFactory.getVehicle("bike");

v2.start();

Vehicle v3 = VehicleFactory.getVehicle("truck");

v3.start();

}

}  
  
  
**O/P:**  
  


**Topic 2:** Algorithms\_Data Structures

**Exercise 1**E-commerce Platform Search Function  
  
**Code :**  
  
import java.util.\*;

class Product {

private String id;

private String name;

private String category;

public Product(String id, String name, String category) {

this.id = id;

this.name = name.toLowerCase();

this.category = category.toLowerCase();

}

public String getName() {

return name;

}

public String getCategory() {

return category;

}

@Override

public String toString() {

return "Product [id=" + id + ", name=" + name + ", category=" + category + "]";

}

}

class ProductCatalog {

private List<Product> productList;

public ProductCatalog() {

productList = new ArrayList<>();

initializeProducts();

}

private void initializeProducts() {

productList.add(new Product("P001", "Samsung Galaxy M14", "Electronics"));

productList.add(new Product("P002", "Nike Running Shoes", "Footwear"));

productList.add(new Product("P003", "Galaxy Watch", "Accessories"));

productList.add(new Product("P004", "Apple iPhone 13", "Electronics"));

productList.add(new Product("P005", "Adidas Sports T-shirt", "Clothing"));

}

public List<Product> searchByKeyword(String keyword) {

keyword = keyword.toLowerCase();

List<Product> result = new ArrayList<>();

for (Product product : productList) {

if (product.getName().contains(keyword) || product.getCategory().contains(keyword)) {

result.add(product);

}

}

return result;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ProductCatalog catalog = new ProductCatalog();

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

String keyword = scanner.nextLine();

List<Product> matchedProducts = catalog.searchByKeyword(keyword);

if (matchedProducts.isEmpty()) {

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

}

else {

System.out.println("Search Results:");

for (Product p : matchedProducts) {

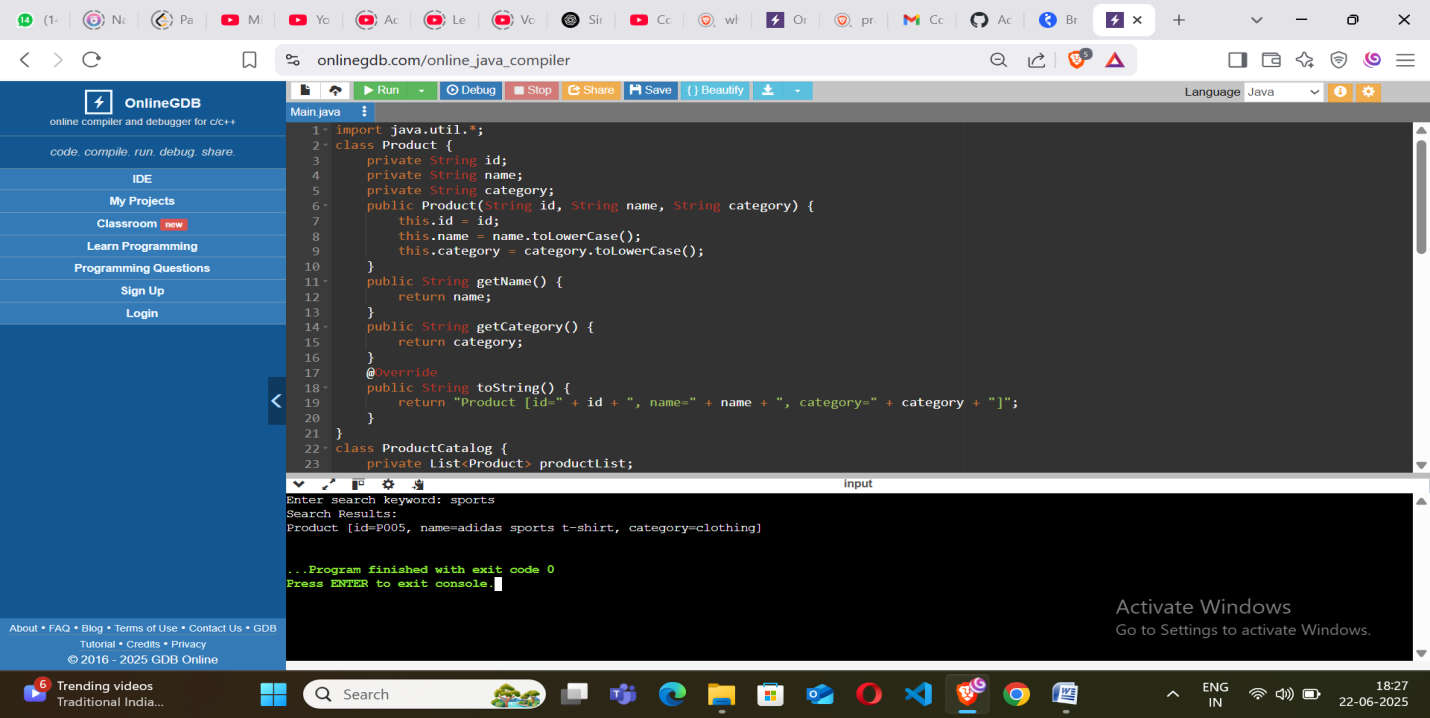
System.out.println(p);

}

}

}

}

**O/P:**  
  
**Exercise 2**Financial Forecasting  
  
**Code :**   
  
import java.util.\*;

public class Main {

static class MonthData {

int month;

int income;

int expense;

int saving;

MonthData(int month, int income, int expense) {

this.month = month;

this.income = income;

this.expense = expense;

this.saving = income - expense;

}

@Override

public String toString() {

return "Month " + month + ": Income = Rs." + income + ", Expense = Rs." + expense + ", Saving = Rs." + saving;

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of months: ");

int n = sc.nextInt();

List<MonthData> monthList = new ArrayList<>();

System.out.println("Enter income and expenses for each month:");

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

System.out.print("Month " + i + " Income: ");

int income = sc.nextInt();

System.out.print("Month " + i + " Expense: ");

int expense = sc.nextInt();

monthList.add(new MonthData(i, income, expense));

}

int totalSavings = 0;

for (MonthData md : monthList) {

totalSavings += md.saving;

}

System.out.println("\nDetailed Month-wise Report:");

for (MonthData md : monthList) {

System.out.println(md);

}

System.out.println("Total Savings after " + n + " months: Rs." + totalSavings);

System.out.print("\nEnter your target savings: ");

int target = sc.nextInt();

int cumulative = 0;

boolean reached = false;

for (MonthData md : monthList) {

cumulative += md.saving;

if (cumulative >= target) {

System.out.println("Target of Rs." + target + " will be reached in month " + md.month);

reached = true;

break;

}

}

if (!reached) {

System.out.println("Target not reached in the given duration.");

}

}

}

**O/P:**  
  
