**WEEK - 1**

**Design principles & Patterns**

1) Exercise 1: Implementing the Singleton Pattern

CODE:

public class Singleton {

// Step 1: Create a private static variable to hold the single instance

private static Singleton instance;

// Step 2: Private constructor prevents instantiation from other classes

private Singleton() {

System.out.println("Singleton instance created!");

}

// Step 3: Provide a public static method to get the instance

public static Singleton getInstance() {

if (instance == null) {

instance = new Singleton(); // Lazy initialization

}

return instance;

}

// A sample method

public void showMessage() {

System.out.println("Hello from the Singleton instance!");

}

// Step 4: Main method to test the Singleton

public static void main(String[] args) {

// Get the singleton instance

Singleton obj1 = Singleton.getInstance();

obj1.showMessage();

// Try to get another instance

Singleton obj2 = Singleton.getInstance();

obj2.showMessage();

// Check if both references are the same

if (obj1 == obj2) {

System.out.println("Both obj1 and obj2 refer to the same Singleton instance.");

} else {

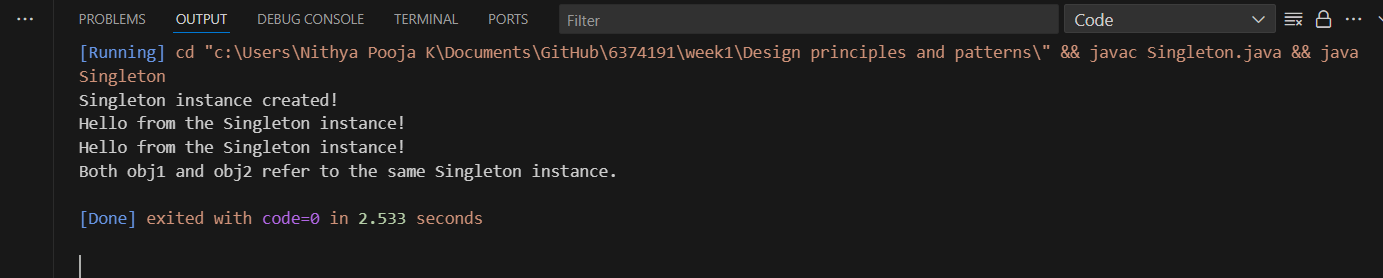
System.out.println("Different instances exist! Singleton failed.");

}

}

}

OUTPUT:



2) Exercise 2: Implementing the Factory Method Pattern

CODE:

// FactoryMethodPattern.java

// Step 1: Create the Product interface

interface Animal {

    void speak();

}

// Step 2: Create Concrete Products

class Dog implements Animal {

    public void speak() {

        System.out.println("Dog says: Woof!");

    }

}

class Cat implements Animal {

    public void speak() {

        System.out.println("Cat says: Meow!");

    }

}

// Step 3: Create the Creator (Factory)

class AnimalFactory {

    public Animal getAnimal(String type) {

        if (type == null) {

            return null;

        }

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

            return new Dog();

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

            return new Cat();

        } else {

            return null;

        }

    }

}

// Step 4: Client code (main method to test)

public class FactoryMethodPattern {

    public static void main(String[] args) {

        AnimalFactory factory = new AnimalFactory();

        Animal animal1 = factory.getAnimal("Dog");

        if (animal1 != null) {

            animal1.speak();

        }

        Animal animal2 = factory.getAnimal("Cat");

        if (animal2 != null) {

            animal2.speak();

        }

        Animal animal3 = factory.getAnimal("Tiger");

        if (animal3 == null) {

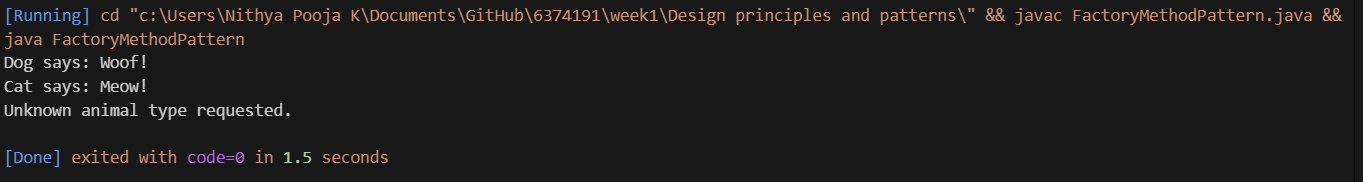
            System.out.println("Unknown animal type requested.");

        }

    }

}

OUTPUT:



3) Exercise 2: E-commerce Platform Search Function

CODE:

// EcommerceSearch.java

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Product {

    private String name;

    private String category;

    private double price;

    // Constructor

    public Product(String name, String category, double price) {

        this.name = name.toLowerCase();

        this.category = category.toLowerCase();

        this.price = price;

    }

    // Getters

    public String getName() {

        return name;

    }

    public String getCategory() {

        return category;

    }

    public double getPrice() {

        return price;

    }

    // Display product

    public void display() {

        System.out.println("Name: " + name + " | Category: " + category + " | Price: ₹" + price);

    }

}

public class EcommerceSearch {

    public static void main(String[] args) {

        // Step 1: Create some sample products

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

        productList.add(new Product("iPhone 15", "Electronics", 79999));

        productList.add(new Product("Samsung Galaxy S24", "Electronics", 69999));

        productList.add(new Product("Dell Laptop", "Computers", 55999));

        productList.add(new Product("T-shirt", "Clothing", 499));

        productList.add(new Product("Coffee Maker", "Kitchen", 2999));

        // Step 2: Get search input from user

        Scanner scanner = new Scanner(System.in);

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

        String keyword = scanner.nextLine().toLowerCase();

        System.out.println("\nSearch results for \"" + keyword + "\":");

        // Step 3: Search and display matching products

        boolean found = false;

        for (Product product : productList) {

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

                product.display();

                found = true;

            }

        }

        if (!found) {

            System.out.println("No products found matching the keyword.");

        }

        scanner.close();

    }

}

OUTPUT:



4) Exercise 7: Financial Forecasting

CODE:

// FinancialForecasting.java

import java.util.Scanner;

public class FinancialForecasting {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Step 1: Input number of months and past revenue data

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

        int months = scanner.nextInt();

        double[] revenues = new double[months];

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

            System.out.print("Enter revenue for month " + (i + 1) + ": ₹");

            revenues[i] = scanner.nextDouble();

        }

        // Step 2: Calculate average growth rate

        double totalGrowthRate = 0;

        int growthPoints = 0;

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

            if (revenues[i - 1] != 0) {

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

                totalGrowthRate += growth;

                growthPoints++;

            }

        }

        double averageGrowthRate = (growthPoints > 0) ? totalGrowthRate / growthPoints : 0;

        // Step 3: Forecast revenue for next N months

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

        int forecastMonths = scanner.nextInt();

        System.out.println("\n--- Forecast for Next " + forecastMonths + " Months ---");

        double lastRevenue = revenues[months - 1];

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

            lastRevenue = lastRevenue \* (1 + averageGrowthRate);

            System.out.printf("Month %d Forecast: ₹%.2f\n", months + i, lastRevenue);

        }

        // Step 4: Display average growth rate

        System.out.printf("\nAverage Monthly Growth Rate: %.2f%%\n", averageGrowthRate \* 100);

        scanner.close();

    }

}

OUTPUT:

