**WEEK1:**

**Module 1 - Design Patterns and Principles**

**Exercise 1: Implementing the Singleton Pattern**

**Code:**

package com.module1;

public class SingleTon {

private static SingleTon *instance*;

// private constructor

private SingleTon() {

}

public static SingleTon getInstance() {

if (*instance* == null) {

*instance* = new SingleTon();

}

return *instance*;

}

public void showMessage() {

System.*out*.println("Hello Singleton!");

}

public static void main(String[] args) {

SingleTon obj = SingleTon.*getInstance*();

obj.showMessage();

}

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

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

**CODE:**

package com.module1;

public class FactoryMethod\_Pattern {

static abstract class Product {

public abstract void use();

}

static class Book extends Product {

public void use() {

System.*out*.println("Reading a Book");

}

}

static class Laptop extends Product {

public void use() {

System.*out*.println("Using a Laptop");

}

}

public static Product createProduct(String type) {

if (type.equalsIgnoreCase("book"))

return new Book();

else if (type.equalsIgnoreCase("laptop"))

return new Laptop();

return null;

}

public static void main(String[] args) {

FactoryMethod\_Pattern.Product product = FactoryMethod\_Pattern.*createProduct*("laptop");

if (product != null) product.use();

}

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Module 2: Data structures and Algorithms**

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

**CODE:**

package com.module2;

import java.util.\*;

public class Ecommerce {

static class Product {

String name, category;

double price;

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

this.name = name;

this.category = category;

this.price = price;

}

public String toString() {

return name + " | " + category + " | ₹" + price;

}

}

static class ProductSearch {

public List<Product> search(List<Product> products, String keyword) {

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

for (Product product : products) {

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

result.add(product);

}

}

return result;

}

public static void main(String[] args) {

List<Product> products = Arrays.*asList*(

new Product("iPhone", "Electronics", 70000),

new Product("Table", "Furniture", 5000),

new Product("T-shirt", "Clothing", 600)

);

ProductSearch search = new ProductSearch();

List<Product> result = search.search(products, "i");

for (Product p : result) {

System.*out*.println(p);

}

}

}}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 7: Financial Forecasting**

**CODE:**

package com.module2;

public class Financial\_Forecasting {

public static double predictNextYearRevenue(double[] pastRevenues) {

double growthRateSum = 0;

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

growthRateSum += (pastRevenues[i] - pastRevenues[i - 1]) / pastRevenues[i - 1];

}

double averageGrowthRate = growthRateSum / (pastRevenues.length - 1);

return pastRevenues[pastRevenues.length - 1] \* (1 + averageGrowthRate);

}

public static void main(String[] args) {

double[] revenues = {100000, 120000, 140000}; // Past 3 years

double predicted = *predictNextYearRevenue*(revenues);

System.*out*.println("Predicted revenue for next year: ₹" + predicted);

}

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**