**Problem statement-1: Implementing the singleton pattern.**

**Solution:**

public class Logger {

private static Logger instance;

private Logger() {

System.out.println("initialized.");

}

public static synchronized Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

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

}

}

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("Application started.");

logger2.log("Another log message.");

if (logger1 == logger2) {

System.out.println("Both logger1 and logger2 refer to the same");

} else {

System.out.println("Singleton pattern NOT working!");

}

}

}

**Output:**

initialized.

[LOG]: Application started.

[LOG]: Another log message.

Both logger1 and logger2 refer to the same

**Problem statement-2: Implementing the factory method pattern.**

**Solution:**

public interface Document {

void open();

}

public class WordDocument implements Document {

public void open() {

System.out.println("Opening a Word document.");

}

}

public class PdfDocument implements Document {

public void open() {

System.out.println("Opening a PDF document.");

}

}

public class ExcelDocument implements Document {

@Override

public void open() {

System.out.println("Opening an Excel document.");

}

}

public abstract class DocumentFactory {

public abstract Document createDocument();

}

public class WordDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

public class PdfDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

public class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

public class DocumentFactoryTest {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:**

Opening a Word document.

Opening a PDF document.

Opening an Excel document.

**Problem statement-3: E-commerce Platform Search Function**

**Solution:**

public class Product {

private int productId;

private String productName;

private String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public String getCategory() {

return category;

}

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

import java.util.Arrays;

import java.util.Comparator;

public class ProductSearch {

public static Product linearSearch(Product[] products, String targetName) {

for (Product p : products) {

if (p.getProductName().equalsIgnoreCase(targetName)) {

return p;

}

}

return null;

}

public static Product binarySearch(Product[] products, String targetName) {

Arrays.sort(products, Comparator.comparing(Product::getProductName));

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

String midName = products[mid].getProductName();

int cmp = midName.compareToIgnoreCase(targetName);

if (cmp == 0) return products[mid];

else if (cmp < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

}

Import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(101, "Laptop", "Electronics"),

new Product(102, "Smartphone", "Electronics"),

new Product(103, "Shoes", "Fashion"),

new Product(104, "Backpack", "Accessories"),

new Product(105, "Headphones", "Electronics")

};

Scanner sc=new Scanner(System.in);

System.out.println(“Enter the product Name: “);

String searchTarget=sc.nextLine();

Product result1 = ProductSearch.linearSearch(products, searchTarget);

System.out.println("Linear Search: " + (result1 != null ? result1 : "No result "));

Product result2 = ProductSearch.binarySearch(products, searchTarget);

System.out.println("Binary Search : " + (result2 != null ? result2 : "No result"));

}

}

**Output :**

Enter the product Name:

shoes

Linear Search : 103 - Shoes (Fashion)

Binary Search : 103 - Shoes (Fashion)

**Output :**

Enter the product Name:

Tablet

Linear Search : No result

Binary Search : No result

**Problem statement-4: Financial Forecasting**

**Solution:**

Import java.util.\*;

public class FinancialForecast {

public static double futureValue(double currentValue, double rate, int years) {

if (years == 0) {

return currentValue;

}

return futureValue(currentValue, rate, years - 1) \* (1 + rate);

}

public static void main(String[] args) {

double currentValue = 10000;

double annualRate = 0.05;

Scanner sc=new Scanner(System.in);

System.out.println(“Enter the number of Years: ”);

Int years=sc.nextInt();

double result = futureValue(currentValue, annualRate, years);

System.out.printf("Predicted value after %d years: %.2f\n", years, result);

}

}

**Output :**

Enter the number of Years:

5

Predicted value after 5 years: 12762.82

**Output :**

Enter the number of Years:

10

Predicted value after 10 years: 29507.27