DESIGN PATTERNS AND PRINCIPLES:

EX:1 - Implementing the Singleton Pattern

public class Main {

static class Logger {

private static Logger instance;

private Logger() {

System.out.println("Logger Instance Created");

}

public static Logger getInstance(){

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("Log: " + message);

}

}

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("This is the first log message.");

Logger logger2 = Logger.getInstance();

logger2.log("This is the second log message.");

if (logger1 == logger2) {

System.out.println("Same instance of Logger is used.");

} else {

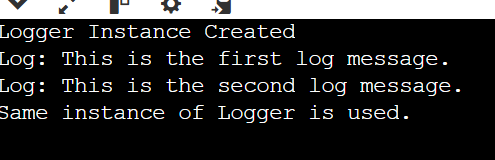
System.out.println("Different Logger instances exist.");

}

}

}

OUTPUT:



EX:2 - Implementing the Factory Method Pattern

public class Main {

interface Document {

void open();

}

static class WordDocument implements Document {

public void open() {

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

}

}

static class PdfDocument implements Document {

public void open() {

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

}

}

static class ExcelDocument implements Document {

public void open() {

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

}

}

static abstract class DocumentFactory {

public abstract Document createDocument();

}

static class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

static class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

static class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document word = wordFactory.createDocument();

word.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdf = pdfFactory.createDocument();

pdf.open();

DocumentFactory excelFactory = new ExcelFactory();

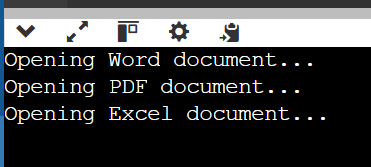
Document excel = excelFactory.createDocument();

excel.open();

}

}

OUTPUT:



DATA STRUCTURES AND ALGORITHMS

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

import java.util.Arrays;

import java.util.Comparator;

public class Main {

static class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

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

}

}

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

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

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

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int comparison = products[mid].productName.compareToIgnoreCase(targetName);

if (comparison == 0) {

return products[mid];

} else if (comparison < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Shirt", "Clothing"),

new Product(103, "Mobile", "Electronics"),

new Product(104, "Shoes", "Footwear"),

new Product(105, "Watch", "Accessories")

};

System.out.println("=== Linear Search ===");

Product result1 = linearSearch(products, "Mobile");

if (result1 != null) {

System.out.println("Product found: " + result1);

} else {

System.out.println("Product not found.");

}

Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

System.out.println("\n=== Binary Search ===");

Product result2 = binarySearch(products, "Mobile");

if (result2 != null) {

System.out.println("Product found: " + result2);

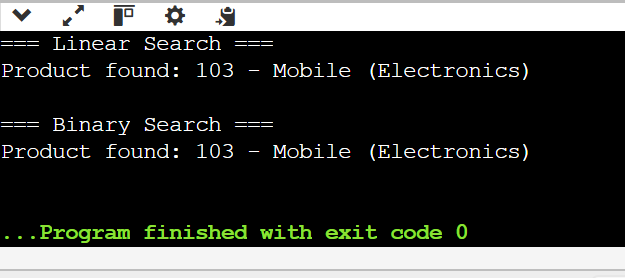
} else {

System.out.println("Product not found.");

}

}

}

OUTPUT:

EX:7 **-** Financial Forecasting

public class Main {

public static double futureValue(int year, double initialValue, double growthRate) {

if (year == 0) {

return initialValue;

}

return futureValue(year - 1, initialValue, growthRate) \* (1 + growthRate);

}

public static void main(String[] args) {

double initialInvestment = 10000.0;

double annualGrowthRate = 0.10;

int forecastYears = 5;

System.out.println("Financial Forecast for " + forecastYears + " years:");

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

double value = futureValue(i, initialInvestment, annualGrowthRate);

System.out.printf("Year %d: ₹%.2f\n", i, value);

}

}

}

OUTPUT:

