**PROBLEM 1**

class Logger {

private static Logger singleInstance;

private Logger() {

System.out.println("Logger created!");

}

public static Logger getInstance() {

if (singleInstance == null) {

singleInstance = new Logger();

}

return singleInstance;

}

public void log(String message) {

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

}

}

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("Starting the system...");

Logger logger2 = Logger.getInstance();

logger2.log("Continuing with operation...");

if (logger1 == logger2) {

System.out.println("Only one Logger instance is used!");

} else {

System.out.println("Multiple Logger instances exist!");

}

}

}

**PROBLEM 2**

interface Document {

void open();

}

class WordDocument implements Document {

public void open() {

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

}

}

class PdfDocument implements Document {

public void open() {

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

}

}

class ExcelDocument implements Document {

public void open() {

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

}

}

abstract class DocumentFactory {

public abstract Document createDocument();

}

class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

public class Main {

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();

}

}

**PROBLEM 3**

import java.util.Arrays;

import java.util.Comparator;

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 class Main {

// Linear Search

public static Product linearSearch(Product[] products, int targetId) {

for (Product p : products) {

if (p.productId == targetId) {

return p;

}

}

return null;

}

// Binary Search (Assumes array is sorted by productId)

public static Product binarySearch(Product[] products, int targetId) {

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (products[mid].productId == targetId) {

return products[mid];

} else if (products[mid].productId < targetId) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(101, "Shoes", "Fashion"),

new Product(102, "Book", "Education"),

new Product(104, "Phone", "Electronics")

};

// Linear Search

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

Product result1 = linearSearch(products, 102);

System.out.println(result1 != null ? "Found: " + result1 : "Product not found");

// Binary Search (requires sorting first)

Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

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

Product result2 = binarySearch(products, 102);

System.out.println(result2 != null ? "Found: " + result2 : "Product not found");

}

}

**PROBLEM 4**

public class FinancialForecast {

public static double predictFutureValue(double initialValue, double growthRate, int years) {

if (years == 0) {

return initialValue;

} else {

return predictFutureValue(initialValue, growthRate, years - 1) \* (1 + growthRate);

}

}

public static void main(String[] args) {

double initialValue = 10000;

double growthRate = 0.1;

int years = 5;

double futureValue = predictFutureValue(initialValue, growthRate, years);

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

}

}