**Week1:**

**Design Principles and Patterns**

**HandsON**

**Exercise 1: Implementing the Singleton Pattern**

*Filename: Logger.java*

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

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

}

}

*Filename: Main.java  
  
public class Main {*

*public static void main(String[] args) {*

*Logger logger1 = Logger.getInstance();*

*logger1.log("First log message is received.");*

*Logger logger2 = Logger.getInstance();*

*logger2.log("Second log message is received.");*

*if (logger1 == logger2) {*

*System.out.println("Both logger instances are the same - singleton confirmed.");*

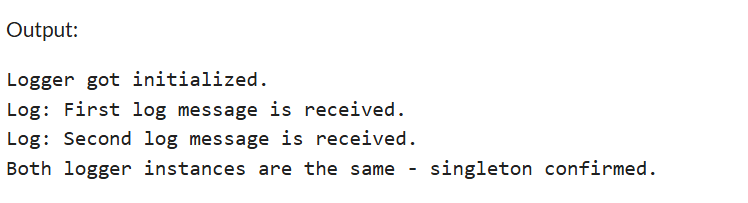
*} else {*

*System.out.println("Different instances so singleton failed.");*

*}*

*}*

*}*



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

Filename: DocumentFactoryDemo.java

public class DocumentFactoryDemo {

interface Document {

void open();

}

static class WordDocument implements Document {

public void open() {

System.out.println("Word Document is Opening ...");

}

}

static class PdfDocument implements Document {

public void open() {

System.out.println("PDF Document is Opening ...");

}

}

static class ExcelDocument implements Document {

public void open() {

System.out.println("Excel Document is Opening ...");

}

}

static abstract class DocumentFactory {

public abstract Document createDocument();

}

static class WordDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

static class PdfDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

static class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

}

Filename: Main.java

public class Main {

public static void main(String[] args) {

DocumentFactoryDemo.DocumentFactory wordFactory = new DocumentFactoryDemo.WordDocumentFactory();

DocumentFactoryDemo.Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactoryDemo.DocumentFactory pdfFactory = new DocumentFactoryDemo.PdfDocumentFactory();

DocumentFactoryDemo.Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

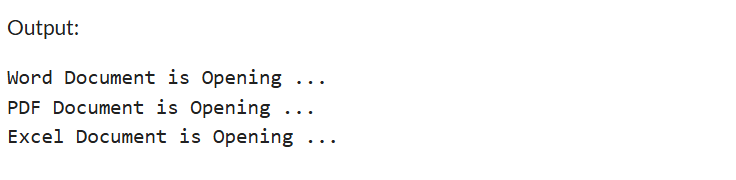
DocumentFactoryDemo.DocumentFactory excelFactory = new DocumentFactoryDemo.ExcelDocumentFactory();

DocumentFactoryDemo.Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}



**Data structures and algorithms**

**HandsON**

**Exercise 1: Inventory Management System**

*Filename: Product.java*public class Product {

private String productId;

private String productName;

private int quantity;

private double price;

public Product(String productId, String productName, int quantity, double price) {

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

public String getProductId() { return productId; }

public String getProductName() { return productName; }

public int getQuantity() { return quantity; }

public double getPrice() { return price; }

public void setProductName(String productName) { this.productName = productName; }

public void setQuantity(int quantity) { this.quantity = quantity; }

public void setPrice(double price) { this.price = price; }

@Override

public String toString() {

return "Product[ID=" + productId + ", Name=" + productName + ", Qty=" + quantity + ", Price=" + price + "]";

}

}

Filename: InventoryManager.java

import java.util.HashMap;

public class InventoryManager {

private HashMap<String, Product> inventory = new HashMap<>();

public void addProduct(Product product) {

inventory.put(product.getProductId(), product);

System.out.println("Product added: " + product);

}

public void updateProduct(String productId, String name, int quantity, double price) {

Product product = inventory.get(productId);

if (product != null) {

product.setProductName(name);

product.setQuantity(quantity);

product.setPrice(price);

System.out.println("Product updated: " + product);

} else {

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

}

}

public void deleteProduct(String productId) {

if (inventory.containsKey(productId)) {

inventory.remove(productId);

System.out.println("Product removed: " + productId);

} else {

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

}

}

public void showInventory() {

System.out.println("-------- Inventory --------");

for (Product product : inventory.values()) {

System.out.println(product);

}

}

}

*Filename:Main.java*

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

Product p1 = new Product("P1", "Desktop", 10, 75000.0);

Product p2 = new Product("P2", "Keyboard", 50, 1500.0);

Product p3 = new Product("P3", "HardDisk", 40, 800.0);

manager.addProduct(p1);

manager.addProduct(p2);

manager.addProduct(p3);

manager.showInventory();

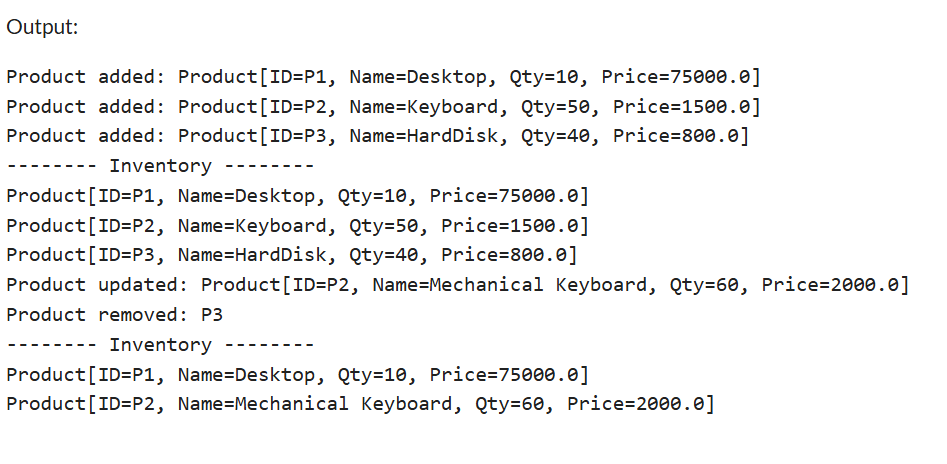
manager.updateProduct("P2", "Mechanical Keyboard", 60, 2000.0);

manager.deleteProduct("P3");

manager.showInventory();

}

}



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

*Filename: Product.java*

public class Product {

private String productId;

private String productName;

private String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String getProductName() {

return productName;

}

public String getProductId() {

return productId;

}

public String getCategory() {

return category;

}

@Override

public String toString() {

return "[" + productId + "] " + productName + " (" + category + ")";

}

}

*Filename: SearchEngine.java*

import java.util.Arrays;

import java.util.Comparator;

public class SearchEngine {

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

for (Product product : products) {

if (product.getProductName().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;

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

int compare = midName.compareToIgnoreCase(targetName);

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

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

else right = mid - 1;

}

return null;

}

public static void sortByName(Product[] products) {

Arrays.sort(products, Comparator.comparing(Product::getProductName, String.CASE\_INSENSITIVE\_ORDER));

}

}  
  
*Filename : Main.java*  
public class Main {

public static void main(String[] args) {

Product[] products = {

new Product("P1", "Desktop", "Electronics"),

new Product("P2", "Jeans", "Fashion"),

new Product("P3", "SmartPhone", "Electronics"),

new Product("P4", "Shoes", "Fashion"),

new Product("P5", "Camera", "Electronics")

};

Product result1 = SearchEngine.linearSearch(products, "Camera");

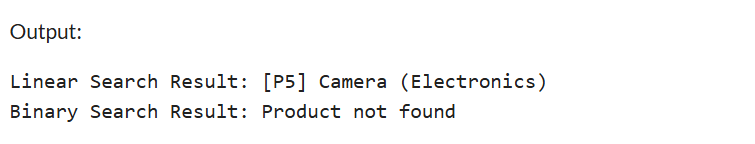
System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Product not found"));

SearchEngine.sortByName(products);

Product result2 = SearchEngine.binarySearch(products, "Air Cooller");

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

}

}  
  


**Exercise 7: Financial Forecasting**

*Filename: FinancialForecast.java*

public class FinancialForecast {

// Recursive method to calculate future value

public static double calculateFutureValue(double initialAmount, double growthRate, int years) {

if (years == 0) {

return initialAmount;

}

return calculateFutureValue(initialAmount, growthRate, years - 1) \* (1 + growthRate);

}

public static void main(String[] args) {

double initialAmount = 10000.0; // Starting investment

double growthRate = 0.05; // 5% annual growth

int years = 5; // Forecast for 5 years

double futureValue = calculateFutureValue(initialAmount, growthRate, years);

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

}

}

