**Week 1**

**Module 1 : Design Patterns and Principles**

**1.Implemeting the Singleton Pattern**

**Program :**

public 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 class TestSingleton {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        logger1.log("First log message");

        logger2.log("Second log message");

        if (logger1 == logger2) {

            System.out.println("Both logger instances are the same (singleton works).");

        }

else {

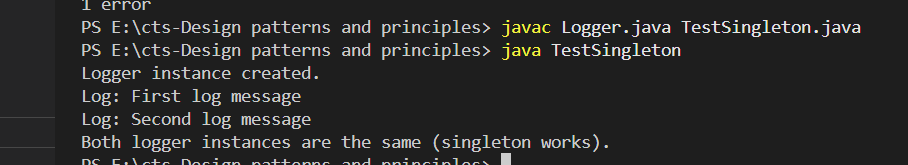
            System.out.println("Logger instances are different (singleton failed).");

        }

    }

}

**Output:**



**2.Implemeting the Factory Method Pattern**

**Program :**

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 WordDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new WordDocument();

    }

}

class PdfDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new PdfDocument();

    }

}

class ExcelDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new ExcelDocument();

    }

}

public class FactoryMethodExample {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordDocumentFactory();

        Document word = wordFactory.createDocument();

        word.open();

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        Document pdf = pdfFactory.createDocument();

        pdf.open();

      DocumentFactory excelFactory = new ExcelDocumentFactory();

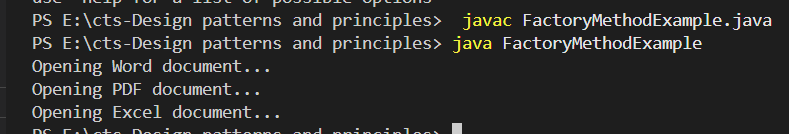
        Document excel = excelFactory.createDocument();

        excel.open();

    }

}

**Output:**



**Module 2 : Data Structures and Algorithms**

**1: Inventory Management System**

**Program:**

import java.util.\*;

public class Inventorymanagement {

    public static void main(String[] args) {

        Inventory inventory = new Inventory();

        inventory.addProduct(new Product(101, "Mouse", 50, 250.0));

        inventory.addProduct(new Product(102, "Keyboard", 30, 550.0));

        inventory.addProduct(new Product(103, "Monitor", 20, 7000.0));

        inventory.updateProduct(102, 40, 525.0);

        inventory.deleteProduct(101);

        inventory.displayAllProducts();

    }

}

class Product {

    int productId;

    String productName;

    int quantity;

    double price;

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

        this.productId = productId;

        this.productName = productName;

        this.quantity = quantity;

        this.price = price;

    }

    public String toString() {

        return "ID: " + productId + ", Name: " + productName + ", Quantity: " + quantity + ", Price: ₹" + price;

    }

}

class Inventory {

    private Map<Integer, Product> products = new HashMap<>();

    public void addProduct(Product product) {

        if (products.containsKey(product.productId)) {

            System.out.println("Product already exists.");

        } else {

            products.put(product.productId, product);

            System.out.println("Product added.");

        }

    }

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

        if (products.containsKey(productId)) {

            Product p = products.get(productId);

            p.quantity = quantity;

            p.price = price;

            System.out.println("Product updated.");

        } else {

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

        }

    }

    public void deleteProduct(int productId) {

        if (products.remove(productId) != null) {

            System.out.println("Product deleted.");

        } else {

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

        }

    }

  public void displayAllProducts() {

        System.out.println("\n--- Current Inventory ---");

        for (Product p : products.values()) {

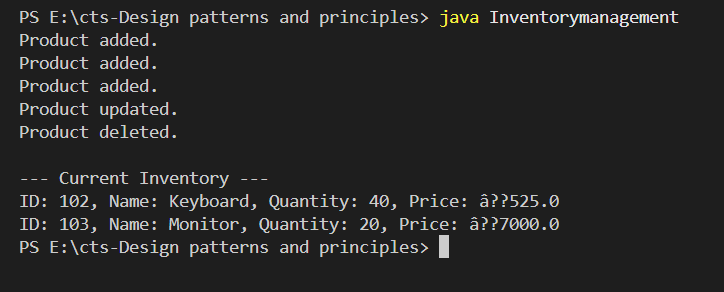
            System.out.println(p);

        }

    }

}

**Output:**

****

**1: E-commerce Platform Search Function**

**Program:**

import java.util.\*;

public class Ecommerce {

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(203, "Shirt", "Clothing"),

            new Product(150, "Laptop", "Electronics"),

            new Product(180, "Book", "Stationery")

        };

Scanner sc = new Scanner(System.in);

        System.out.print("Enter product ID to search: ");

        int id = sc.nextInt();

        System.out.println("\nLinear Search:");

        Product found1 = linearSearch(products, id);

        if (found1 != null) {

            System.out.println("Product found: " + found1);

        } else {

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

        }

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

        System.out.println("\nBinary Search:");

        Product found2 = binarySearch(products, id);

        if (found2 != null) {

            System.out.println("Product found: " + found2);

        } else {

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

        } }

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

        for (Product p : products) {

            if (p.productId == id) {

                return p;

            } }

        return null;

    }

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

        int low = 0, high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

                return products[mid];

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

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

}

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 "ID: " + productId + ", Name: " + productName + ", Category: " + category;

    }

}

**Output:**

