***Week1 Design Principles and Patterns HandsOn***

**Exercise 1: Implementing the Singleton Pattern**

**Code:**

Logger.java

public class Logger {

    /\*Singleton class enables a single instance of Logger to be created and used throughout the application.

    This can be done by creating a private static instance of the class and making the constructor private.\*/

    // 1-> Create a private static instance of Logger class

    private static Logger singleInstance;

    // 2-> Make the constructor private to prevent instantiation from outside of the class

    private Logger() {

        System.out.println("The Logger is Initialized");

    }

    //3-> Provide a public static method to return the single instance

    public static Logger getInstance() {

        if (singleInstance == null) {

            singleInstance = new Logger(); //here we will create the instance only if it is null

        }

        return singleInstance;

    }

    // Example logging method

    public void log(String message) {

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

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        /\*  Create instances of Logger using the singleton Pattern

        This will ensure that only one instance of Logger is created

        and used throughout the application.\*/

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        // Log messages

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

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

        // Check if both instances are the same

        if (logger1 == logger2) {

            System.out.println("Same Logger instance is used.Hence Singleton is Working!");

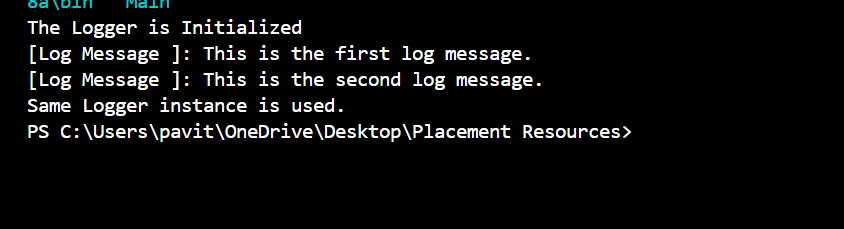
        } else {

            System.out.println("Different instances! Singleton failed.");

        }

    }

}

**OUTPUT** ****

**EXERCISE -2**

**Document.java**

public interface Document {

void open();

}

**WordDocument.java**

public class WordDocument implements Document {

public void open() {

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

}

}

**PdfDocument.java**

public class PdfDocument implements Document {

public void open() {

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

}

}

**ExcelDocument.java**

public class ExcelDocument implements Document {

public void open() {

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

}

}

**DocumentFactory.java**

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**WordFactory.java**

public class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

**PdfFactory.java**

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

**ExcelFactory.java**

public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelFactory();

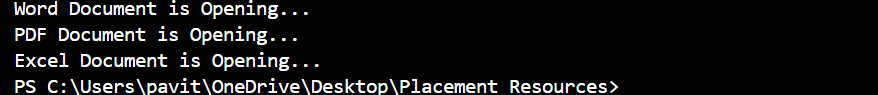
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:**



**EXERCISE – 3**

Main.java

  import java.util.\*;

 class Product{

    int productId;

    String productName,category;

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

    this.productId=productId;

    this.productName=productName;

    this.category=category;

    }

    void displayProduct(){

        System.out.print("Product ID: "+productId);

        System.out.print(" Product Name: "+productName);

        System.out.print(" Category: "+category);

    }

}

public class Main{

    public static void main(String args[])

    {

        Scanner sc=new Scanner(System.in);

        Product[] productArray={

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

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

        new Product(103,"Refrigerator","APPLIANCES"),

        new Product(104,"Washing Machine","APPLIANCES"),

        new Product(105,"Television","ELECTRONICS"),

        new Product(106,"Microwave","APPLIANCES"),

        new Product(107,"Air Conditioner","APPLIANCES"),

        new Product(108,"Vacuum Cleaner","APPLIANCES"),

        new Product(109,"Smartwatch","ELECTRONICS"),

        new Product(110,"Bluetooth Speaker","ELECTRONICS")

    };

    System.out.println("Displaying the Product details:");

    for(Product product:productArray){

        product.displayProduct();

        System.out.println();

        }

        System.out.println("Enter the product ID or The Name of the Product to Search \nPress 1 for ID and 2 for Name:");

        int choice = sc.nextInt();

        sc.nextLine(); // consume newline

        if(choice == 1)

        {

System.out.println("Enter Product ID to search:");

int searchId = sc.nextInt();

int left = 0, right = productArray.length - 1;

boolean found = false;

while (left <= right) {

    int mid = (left + right) / 2;

    if (productArray[mid].productId == searchId) {

        System.out.println("Product Found:");

        productArray[mid].displayProduct();

        found = true;

        break;

    } else if (productArray[mid].productId < searchId) {

        left = mid + 1;

    } else {

        right = mid - 1;

    }

}

if (!found) {

    System.out.println("Sorry, Product Not Found");

}

        }

        else if(choice == 2)

        {

            System.out.println("Enter Product Name to search:");

            String searchName = sc.nextLine();

            boolean found = false;

            for(Product product : productArray)

            {

                if(product.productName.equalsIgnoreCase(searchName))

                {

                    System.out.println("Product Found:");

                    product.displayProduct();

                    found = true;

                    break;

                }

            }

            if(!found)

            {

                System.out.print("Sorry, Product Not Found");

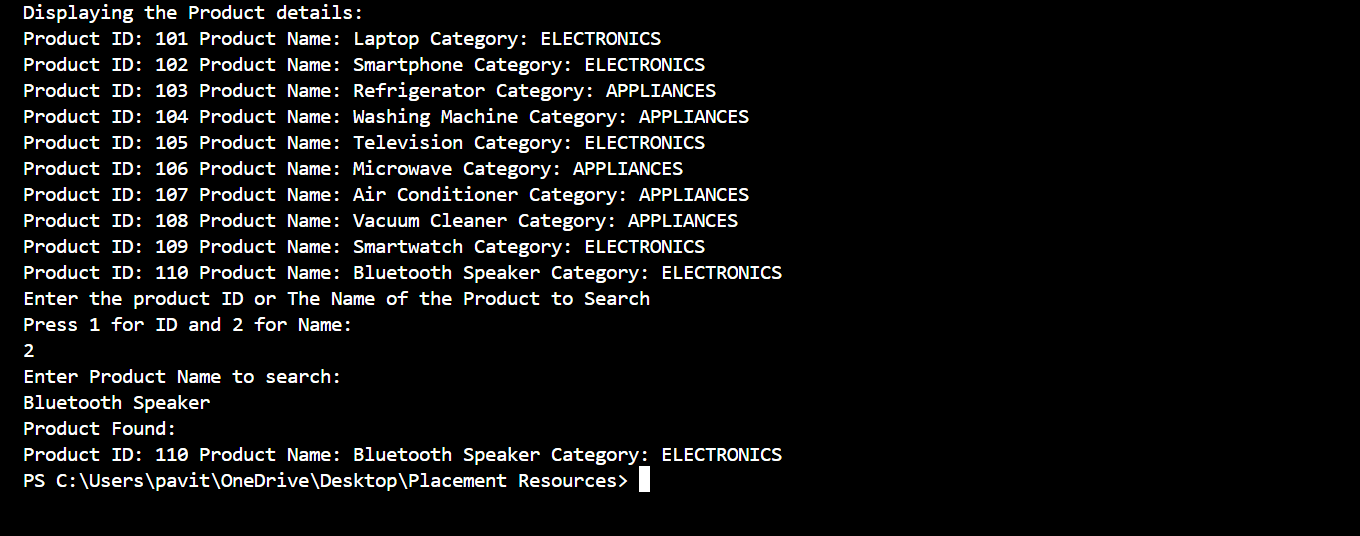
            }

        }

        sc.close();

    }

}

****

**EXERCISE -4**

**Forecast.java**

import java.util.Scanner;

public class Forecast {

    public static double futureValueRecursive(int years, double presentValue, double rate) {

        if (years == 0)

            return presentValue;

        return futureValueRecursive(years - 1, presentValue, rate) \* (1 + rate);

    }

    public static double futureValueIterative(int years, double presentValue, double rate) {

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

            presentValue \*= (1 + rate);

        }

        return presentValue;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        // Input

        System.out.print("Enter Present Value in Rupees: ");

        double presentValue = sc.nextDouble();

        System.out.print("Enter Annual Growth Rate in %: ");

        double rate = sc.nextDouble() / 100.0;

        System.out.print("Enter Number of Years: ");

        int years = sc.nextInt();

        // Recursive Forecast

        double resultRecursive = futureValueRecursive(years, presentValue, rate);

        System.out.printf(" Forecasted Future Value after %d years(Recurrsive): %.2f Ruppees\n", years, resultRecursive);

        double resultIterative = futureValueIterative(years, presentValue, rate);

        System.out.printf(" Forecasted Future Value after %d years(Iterative): ₹%.2f Rupees", years, resultIterative);

        sc.close();

    }

}

output

