**DESIGN PRINCIPLES :**

**1 . Singleton Pattern**

package singleton\_Pattern;

public class Logger {

private static Logger *instance*;

private Logger() {

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

}

public static Logger getInstance() {

if (*instance* == null) {

*instance* = new Logger();

}

return *instance*;

}

public void log(String message) {

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

}

}

package singleton\_Pattern;

public class TestLogger {

public static void main(String[] args) {

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

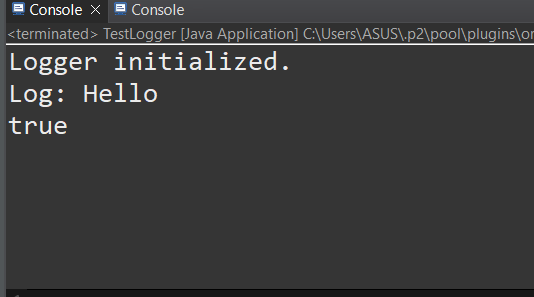
logger1.log("Hello");

System.***out***.println(logger1 == logger2);

}

}

**OUTPUT :**



1. **Factory Method Patterm**

package factory\_Method\_Pattern;

public interface Document {

void open();

}

package factory\_Method\_Pattern;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

package factory\_Method\_Pattern;

public class ExcelDocument implements Document {

public void open() {

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

}

}

package factory\_Method\_Pattern;

public class PdfDocument implements Document {

public void open() {

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

}

}

package factory\_Method\_Pattern;

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

package factory\_Method\_Pattern;

public class TestFactory {

public static void main(String[] args) {

DocumentFactory factory = new PdfFactory();

Document doc = factory.createDocument();

doc.open();

}

}

package factory\_Method\_Pattern;

public class WordDocument implements Document {

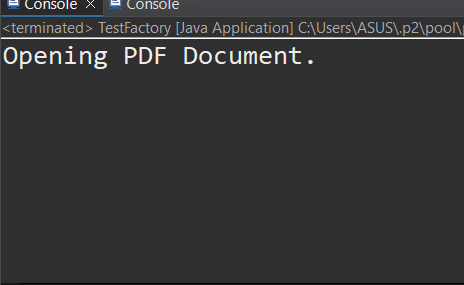
public void open() {

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

}

}

**OUTPUT :**



**DATA STRUCTURES AND ALGORITHMS :**

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

package ecommerce.search;

public 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;

}

*@Override*

public String toString() {

return "Product{" +

"ID=" + productId +

", Name='" + productName + '\'' +

", Category='" + category + '\'' +

'}';

}

}

package ecommerce.search;

import java.util.Arrays;

import java.util.Comparator;

public class ProductSearch{

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

for (Product product : products) {

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

return product;

}

}

return null;

}

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

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

while (left <= right) {

int mid = left + (right - left) / 2;

int cmp = products[mid].productName.compareToIgnoreCase(name);

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

else if (cmp < 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, "Shampoo", "Personal Care"),

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

new Product(104, "Chair", "Furniture"),

new Product(105, "Apple", "Grocery")

};

Product result1 = *linearSearch*(products, "Chair");

System.***out***.println("Linear Search Result: " + (result1 != null ? result1 : "Not Found"));

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

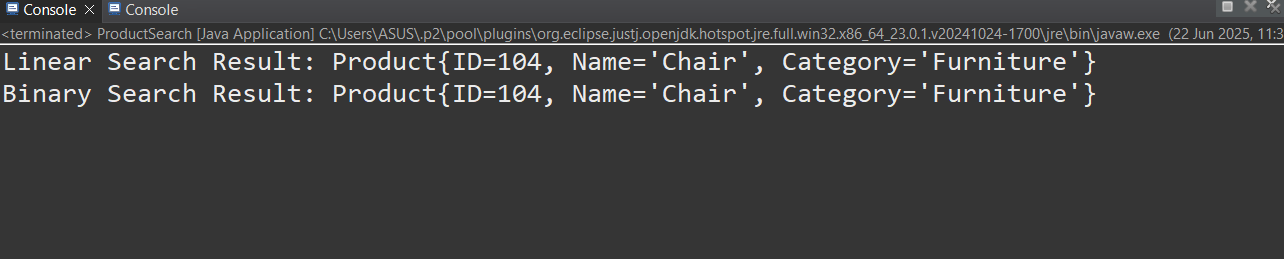
Product result2 = *binarySearch*(products, "Chair");

System.***out***.println("Binary Search Result: " + (result2 != null ? result2 : "Not Found"));

}

}

**OUTPUT :**



1. **Financial Forecasting**

package finance.forecast;

public class Forecast {

public static double forecastValue(double currentValue, double growthRate, int years) {

if (years == 0) return currentValue;

return *forecastValue*(currentValue \* (1 + growthRate), growthRate, years - 1);

}

public static void main(String[] args) {

double currentValue = 10000;

double growthRate = 0.05;

int years = 5;

double futureValue = *forecastValue*(currentValue, growthRate, years);

System.***out***.println("Future Value after " + years + " years: ₹" + futureValue);

}

}

**OUTPUT :**

