**Algorithm and Data Structures :**

1. **E\_commerce:**

**1.1 Product.java:**

package week1.Data\_Structures\_and\_Algoriithms.E\_commerce;

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;

    }

    public String toString() {

        return productId + " - " + productName + " (" + category + ")";

    }

}

* 1. **Search.java**

package week1.Data\_Structures\_and\_Algoriithms.E\_commerce;

import java.util.\*;

public class Search{

    // Linear Search

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

        for (Product product : products) {

            if (product.productId == targetId) {

                return product;

            }

        }

        return null;

    }

    // Binary Search

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

        int low = 0;

        int high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

                return products[mid];

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

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

            new Product(3, "Shoes", "Footwear"),

            new Product(1, "Laptop", "Electronics"),

            new Product(5, "Watch", "Accessories"),

            new Product(2, "Phone", "Electronics"),

            new Product(4, "Bag", "Fashion")

        };

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

        Product result1 = linearSearch(products, 4);

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

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

        // Binary search requires sorted array

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

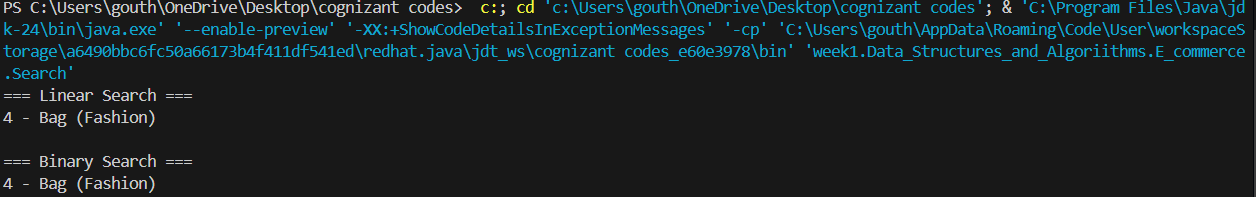
        Product result2 = binarySearch(products, 4);

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

    }

}

**Output:**

****

1. **Financial\_forecasting:**

package week1.Data\_Structures\_and\_Algoriithms.Financial\_forecasting.src;

public class FinancialForecast {

    // Recursive method

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

        if (years == 0) {

            return currentValue;

        }

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

    }

    // Iterative method

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

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

            currentValue \*= (1 + growthRate);

        }

        return currentValue;

    }

    public static void main(String[] args) {

        double initialValue = 10000;

        double growthRate = 0.08;

        int years = 5;

        // Recursive result

        double recursiveResult = calculateFutureValueRecursive(years, initialValue, growthRate);

        System.out.printf("Recursive : Future value after %d years: %.2f\n", years, recursiveResult);

        // Iterative result

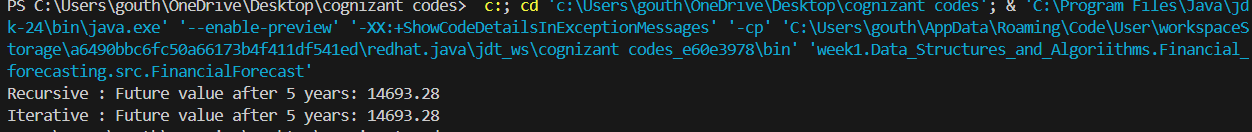
        double iterativeResult = calculateFutureValueIterative(years, initialValue, growthRate);

        System.out.printf("Iterative : Future value after %d years: %.2f\n", years, iterativeResult);

    }

}

**Output:**

****

**Design Principles:**

1. **Singleton\_example;**
   1. **Logger.java**

public class Logger {

    private static Logger instance = null;

    private Logger() {

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

    }

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger(); // Create instance if not created yet

        }

        return instance;

    }

    // A sample log method

    public void log(String message) {

        System.out.println("[LOG] " + message);

    }

}

* 1. **Testlogger.java:**

public class TestLogger {

    public static void main(String[] args) {

        // Fetch Logger instance multiple times

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        Logger logger3 = Logger.getInstance();

        // Log a message

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

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

        // Check if all instances are the same

        if (logger1 == logger2 && logger2 == logger3) {

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

        } else {

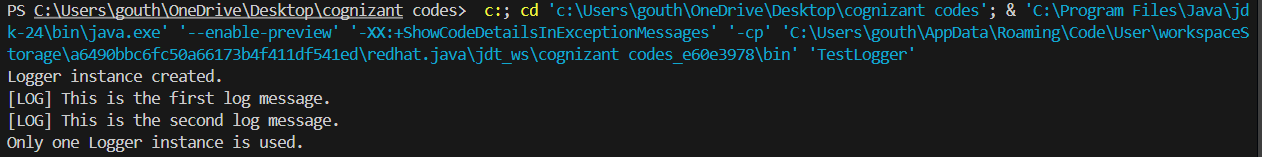
            System.out.println("Different Logger instances exist.");

        }

    }

}

**Output:**

****

1. **Factory\_Method\_example**
   1. **Document.java**

package week1.DesignPrinciples.FactoryMethod.src;

public interface Document {

void open();

}

* 1. **DocumentFactory.java**

package week1.DesignPrinciples.FactoryMethod.src;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

* 1. **ExcelDocument.java**

package week1.DesignPrinciples.FactoryMethod.src;

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

}

**2.4 ExcelDocumentFactory.java**

package week1.DesignPrinciples.FactoryMethod.src;

public class ExcelDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new ExcelDocument();

    }

}

* 1. **PdfDocument.java**

package week1.DesignPrinciples.FactoryMethod.src;

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}

* 1. **PdfDocumentFactory.java**

package week1.DesignPrinciples.FactoryMethod.src;

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

* 1. **WordDocument.java**

package week1.DesignPrinciples.FactoryMethod.src;

public class WordDocument implements Document {

@Override

public void open() {

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

}

}

* 1. **WordDocumentFactory.java**

package week1.DesignPrinciples.FactoryMethod.src;

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

* 1. **Test.java**

package week1.DesignPrinciples.FactoryMethod.src;

public class Test {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

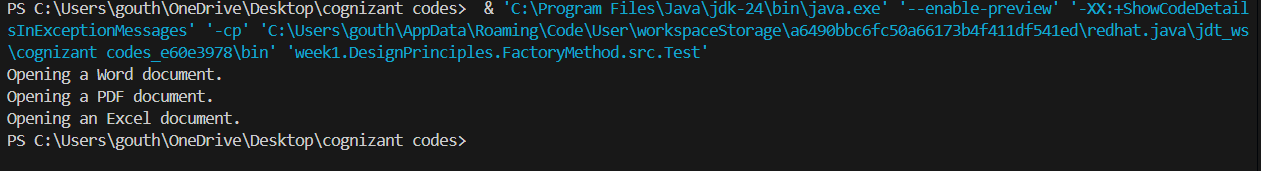
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:**

****