**CTS – DIGITAL NUTURE – 4.0 - DEEPSKILLING PROGRAM**

**WEEK – 1 🡪 DATA STRUCTURES AND ALGORITHMS**

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

**Code:**

***Product.java***

package com.cts.dp.handson.week1;

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 [productId=" + productId + ", productName=" + productName + ", category=" + category + "]";

}

}

***ProductLinearSearch.java***

package com.cts.dp.handson.week1;

public class ProductLinearSearch

{

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

{

for (Product p : products) {

if (p.productName.equalsIgnoreCase(targetName)) {

return p;

}

}

return null;

}

}

***ProductBinarySearch.java***

package com.cts.dp.handson.week1;

import java.util.Arrays;

import java.util.Comparator;

public class ProductBinarySearch

{

public static Product binarySearch(Product[] products, String targetName)

{

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

public static void sortByProductName(Product[] products)

{

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

}

}

***Ecommerece\_Main.java***

package com.cts.dp.handson.week1;

public class Ecommerce\_Main

{

public static void main(String[] args)

{

Product[] products = {

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

new Product(102, "Shampoo", "Personal Care"),

new Product(103, "Notebook", "Stationery"),

new Product(104, "Smartphone", "Electronics"),

new Product(105, "Pen", "Stationery")

};

// Linear Search Test

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

Product foundLinear = ProductLinearSearch.linearSearch(products, "Pen");

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

// Binary Search Test

System.out.println("Binary Search:");

ProductBinarySearch.sortByProductName(products);

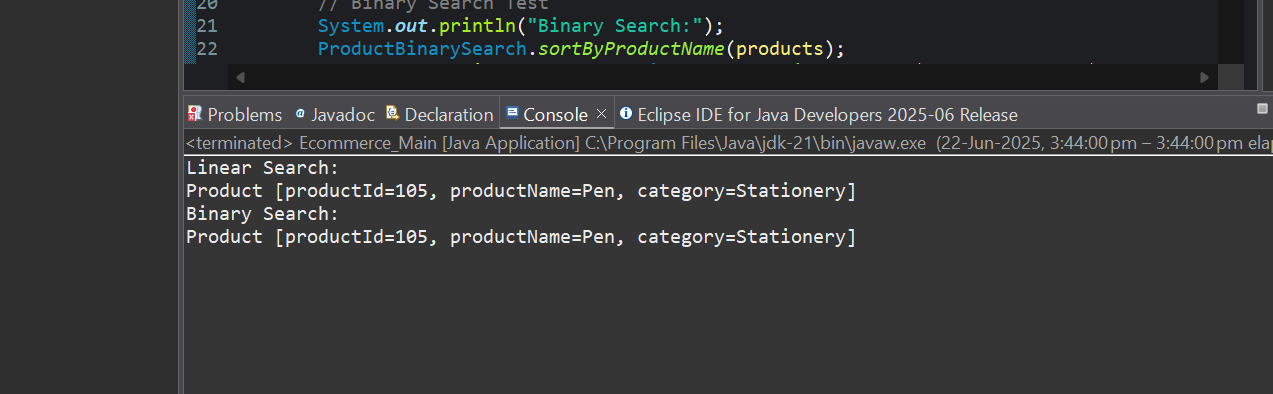
Product foundBinary = ProductBinarySearch.binarySearch(products, "Pen");

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

}

}

**Output:**

****

**Analysis:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| | **Algorithm** |  |  |  | | --- | --- | --- | --- | | **Time Complexity** | **Space Complexity** | **Suitable For** |
| **Linear Search** | O(n) | O(1) | Small unsorted datasets |
| **Binary Search** | O(log n) | O(1) | Large sorted datasets |

**Exercise 7: Financial Forecasting**

**Code:**

***FinancialForecasting.java***

package com.cts.dp.week1;

public class FinancialForecasting

{

public static double futureValue(double currentValue, double growthRate, int years)

{

if (years == 0)

{

return currentValue;

}

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

}

public static void main(String[] args)

{

double initialValue = 1000.0;

double rate = 0.05;

int years = 5;

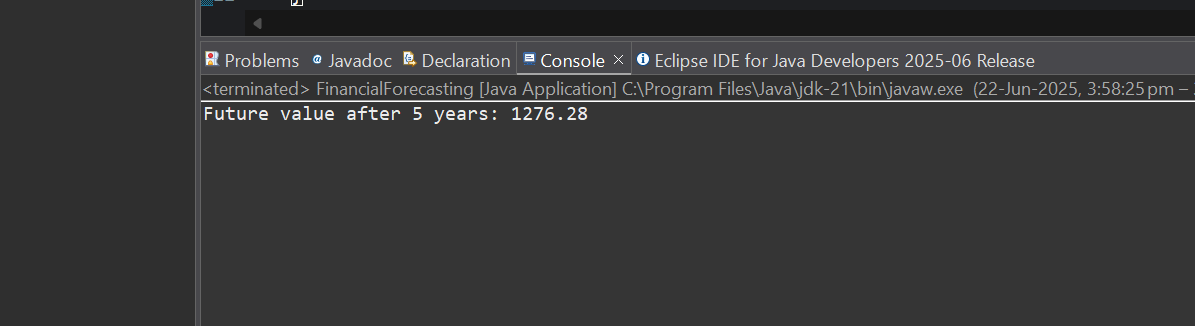
double predictedValue = *futureValue*(initialValue, rate, years);

System.*out*.printf("Future value after %d years: %.2f", years, predictedValue);

}

}

**Output:**

****

**Analysis:**

|  |  |  |
| --- | --- | --- |
| **Method** | **Time Complexity** | **Space Complexity** |
| Recursive | O(n) | O(n) |
| Iterative | O(n) | O(1) |