Exercise 2: E-commerce Platform Search Function

Product.java

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

LinearSearch.java

public class LinearSearch {

public static Product linearSearch(Product[] products, String targetName) {  
 for (Product p : products) {  
 if (p.productName.equalsIgnoreCase(targetName)) {  
 return p;  
 }  
 }  
 return null;  
 }  
}

BinarySearch.java

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

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

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

while (left <= right) {

int mid = left + (right - left) / 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 sortProducts(Product[] products) {

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

}

}

Main.java

public class Main {

public static void main(String[] args) {

Product[] products = {

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

new Product(2, "Shampoo", "Personal Care"),

new Product(3, "Notebook", "Stationery"),

new Product(4, "Chair", "Furniture")

};

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

Product result1 = LinearSearch.linearSearch(products, "Chair");

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

BinarySearch.sortProducts(products);

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

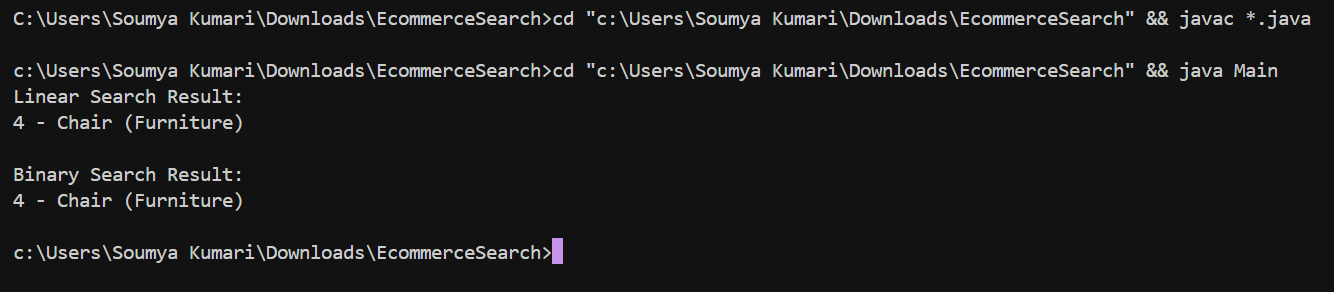
Product result2 = BinarySearch.binarySearch(products, "Chair");

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

}

}

Output :



### Analysis

#### Time Complexity

|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Linear Search | O(n) | O(1) |
| Binary Search | O(log n) | O(1) |

#### Which is More Suitable?

Binary Search is better for performance in large datasets.

Exercise 7: Financial Forecasting

FinancialForecast.java

public class FinancialForecast {

public static double calculateFutureValue(double initialValue, double growthRate, int years) {

if (years == 0) {

return initialValue;

}

return calculateFutureValue(initialValue, growthRate, years - 1) \* (1 + growthRate);

}

public static void main(String[] args) {

double initialValue = 10000;

double annualGrowthRate = 0.05;

int years = 10;

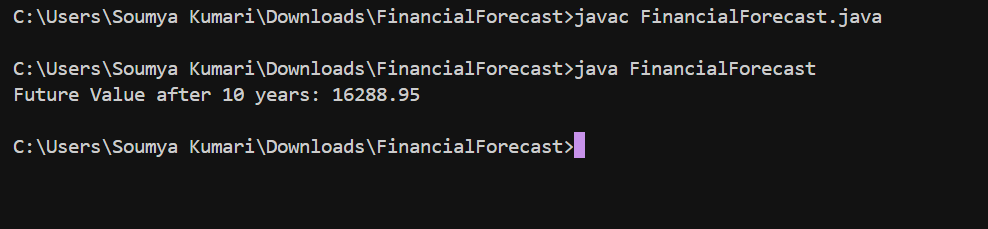
double futureValue = calculateFutureValue(initialValue, annualGrowthRate, years);

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

}

}

Output :



Time Complexity :

1. The time complexity is O(n), where n is the number of years.
2. For each year, the function makes one recursive call.

How to Optimize It :

Iterative Version

public static double iterativeFutureValue(double initialValue, double growthRate, int years) {

double value = initialValue;

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

value \*= (1 + growthRate);

}

return value;

}