**ALGORITHMS AND DATA STRUCTURES**

Exercise 2 : E-commerce Platform Search Function

1. **A. Big O Notation** :

* It is a mathematical notation used to describe the worst case scenario of an algorithm’s time complexity in terms of order of growth of input size.
* It marks the upper bound and helps us analyse the scalability of an algorithm without having to implement the code .
* It is independent of machine specific factors and implementation details like programming language ,etc.
* It is represented as O(f(n)).

**B. Best, Average , Worst Case in Search** :

* **Best :** Element found immediately ( at the first index ) .
* **Average** : Element is in the middle.
* **Worst :** Element is last or absent.

1. **Implementation :**

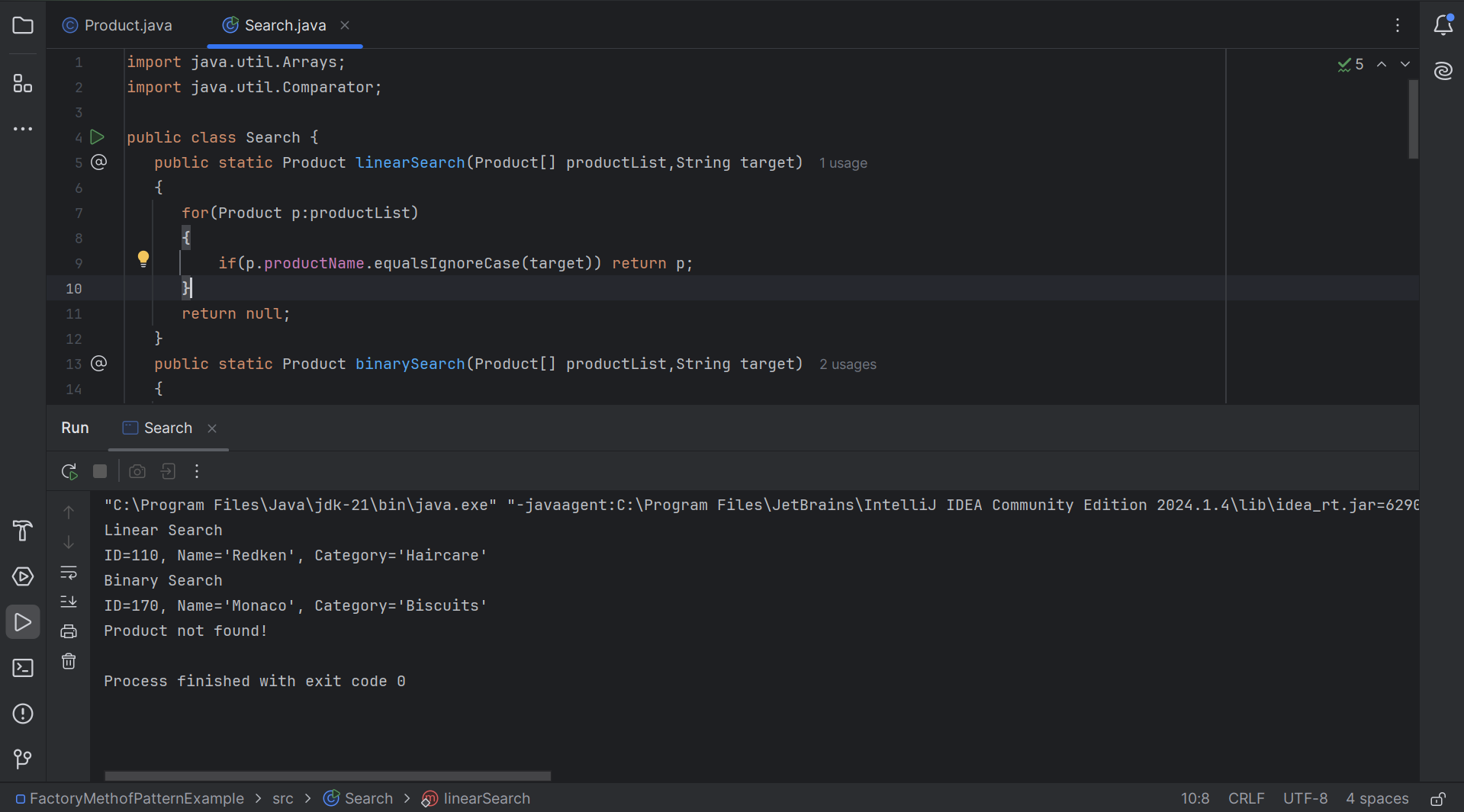
**Product.java**

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;  
 }  
 public String toString()  
 {  
 return  
 "ID=" + productId +  
 ", Name='" + productName + '\'' +  
 ", Category='" + category + '\'';  
 }  
}

**Search.java**

import java.util.Arrays;  
import java.util.Comparator;  
  
public class Search {  
 public static Product linearSearch(Product[] productList,String target)  
 {  
 for(Product p:productList)  
 {  
 if(p.productName.equalsIgnoreCase(target)) return p;  
 }  
 return null;  
 }  
 public static Product binarySearch(Product[] productList,String target)  
 {  
 int l=0;  
 int r= productList.length-1;  
 while(l<=r)  
 {  
 int mid=(l+r)/2;  
 int targetCompare=productList[mid].productName.compareToIgnoreCase(target);  
 if(targetCompare==0) return productList[mid];  
 else if(targetCompare<0) l=mid+1;  
 else r=mid-1;  
 }  
 return null;  
 }  
 public static void main(String[] args)  
 {  
 Product[] productList={  
 new Product(170,"Monaco","Biscuits"),  
 new Product(156,"Hide&Seek","Biscuits"),  
 new Product(139,"Loreal Paris","Haircare"),  
 new Product(100,"Neutrogena","Skincare"),  
 new Product(110,"Redken","Haircare"),  
 new Product(102,"COSRX","Skincare")  
 };  
 System.*out*.println("Linear Search ");  
 Product result1=*linearSearch*(productList,"Redken");  
 System.*out*.println(result1!=null ? result1 : "Product not found!");  
  
 System.*out*.println("Binary Search ");  
 Arrays.*sort*(productList, Comparator.*comparing*(p->p.productName.toLowerCase()));  
 Product result2=*binarySearch*(productList,"Monaco");  
 System.*out*.println(result2!=null ? result2 : "Product not found!");  
  
 Product result3=*binarySearch*(productList,"Laneige");  
 System.*out*.println(result3!=null ? result3 : "Product not found!");  
  
 }  
}

1. **Output :**

****

1. **Analysis:**

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

Which is better for e-commerce platform ?

In most ecommerce platforms , product catalogs are often sorted and is typically huge in size. Thus , binary search is more suited in such cases as it performs in logarithmic time.

However, if the dataset is small and unsorted (which is rare) , linear search may be used.