

DATA STRUCTURES AND ALGORITHMS

MANDATORY HANDS-ON

Exercise 2: E-commerce Platform Search Function

LINEAR SEARCH

Main.java:

```
import java.util.Scanner;

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+']';
    }
}

class LinearSearch{
    public static Product search(Product[] products,int targetId){
        for(Product p:products){
            if(p.productId==targetId){
                return p;
            }
        }
        return null;
    }
}
```

```
}  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        //System.out.println("Try programiz.pro");  
        Product[] products = {  
            new Product(101, "T-Shirt", "Fashion"),  
            new Product(105, "Kurta Set", "Clothing"),  
            new Product(103, "Watch", "Accessories"),  
            new Product(104, "Laptop", "Electronics"),  
            new Product(106, "Bangles", "Jewellery"),  
            new Product(102, "Shoes", "Footwear"),  
        };  
        Scanner s = new Scanner(System.in);  
        int searchId;  
        System.out.println("Enter the search ID of the product : ");  
        searchId=s.nextInt();  
        Product result = LinearSearch.search(products,searchId);  
        if(result!=null){  
            System.out.println("Product Found : "+result);  
        }  
        else{  
            System.out.println("The Product with id "+searchId+" not found");  
        }  
    }  
}
```

OUTPUT:

The screenshot shows the Programiz Online Java Compiler interface. The code editor on the left contains a Java program with a `Product` class and a `LinearSearch` class. The `LinearSearch` class has a static method `search` that iterates through an array of `Product` objects and returns the first one with a matching `productId`. The `Product` class has attributes `productId`, `productName`, and `category`, along with a constructor and a `toString` method. The `main` method in the `LinearSearch` class prompts the user to enter a search ID. The output pane on the right shows the user input '103' and the resulting output 'Product Found : [103,Watch,Accessories]', followed by a success message '=== Code Execution Successful ==='.

```
1 // Online Java Compiler
2 // Use this editor to write, compile and run your Java code online
3 import java.util.Scanner;
4 class Product{
5     int productId;
6     String productName;
7     String category;
8     public Product(int productId,String productName, String category
9     ){
10         this.productId=productId;
11         this.productName=productName;
12         this.category=category;
13     }
14     @Override
15     public String toString(){
16         return "["+productId+","+productName+","+category+"]";
17     }
18 }
19 class LinearSearch{
20     public static Product search(Product[] products,int targetId){
21         for(Product p:products){
22             if(p.productId==targetId){
23                 return p;
24             }
25         }
26     }
27 }
```

Enter the search ID of the product :
103
Product Found : [103,Watch,Accessories]
=== Code Execution Successful ===

The screenshot shows the same Programiz Online Java Compiler interface, but with a different input. The user has entered '107' in the search ID field. The output pane now displays 'The Product with id 107 not found', indicating that the search method did not find a matching product in the array. The success message '=== Code Execution Successful ===' is still present.

```
1 // Online Java Compiler
2 // Use this editor to write, compile and run your Java code online
3 import java.util.Scanner;
4 class Product{
5     int productId;
6     String productName;
7     String category;
8     public Product(int productId,String productName, String category
9     ){
10         this.productId=productId;
11         this.productName=productName;
12         this.category=category;
13     }
14     @Override
15     public String toString(){
16         return "["+productId+","+productName+","+category+"]";
17     }
18 }
19 class LinearSearch{
20     public static Product search(Product[] products,int targetId){
21         for(Product p:products){
22             if(p.productId==targetId){
23                 return p;
24             }
25         }
26     }
27 }
```

Enter the search ID of the product :
107
The Product with id 107 not found
=== Code Execution Successful ===

Binary Search:

Main.java

```
import java.util.Scanner;

import java.util.Arrays;

import java.util.Comparator;

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 + "]";

    }

}

class BinarySearch {

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

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

        while (left <= right) {

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

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

                return products[mid];

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

                left = mid + 1;

            } else {

                right = mid - 1;

            }

        }

    }

}
```

```

    }
}
return null;
}
}

```

```

public class Main {
    public static void main(String[] args) {
        Product[] products = {
            new Product(101, "T-Shirt", "Fashion"),
            new Product(105, "Kurta Set", "Clothing"),
            new Product(102, "Shoes", "Footwear"),
            new Product(106, "Bangles", "Jewellery"),
            new Product(103, "Watch", "Accessories"),
            new Product(104, "Laptop", "Electronics"),
        };

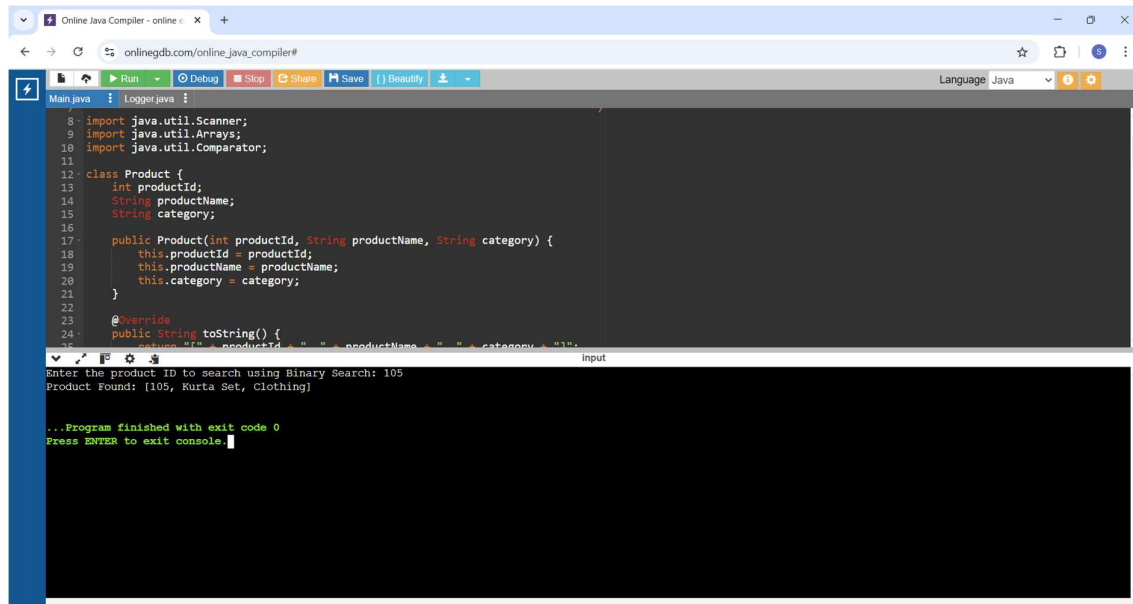
        Arrays.sort(products, Comparator.comparingInt(p -> p.productId)); //sorting
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the product ID to search using Binary Search: ");
        int searchId = scanner.nextInt();

        Product result = BinarySearch.search(products, searchId);
        if (result != null) {
            System.out.println("Product Found: " + result);
        } else {
            System.out.println("Product with ID " + searchId + " not found.");
        }
    }
}

```

OUTPUT :



```
8 import java.util.Scanner;
9 import java.util.Arrays;
10 import java.util.Comparator;
11
12 class Product {
13     int productId;
14     String productName;
15     String category;
16
17     public Product(int productId, String productName, String category) {
18         this.productId = productId;
19         this.productName = productName;
20         this.category = category;
21     }
22
23     @Override
24     public String toString() {
25         return "Product ID: " + productId + ", Product Name: " + productName + ", Category: " + category;
26     }
27 }
28
29 public class Main {
30     public static void main(String[] args) {
31         Scanner scanner = new Scanner(System.in);
32         System.out.println("Enter the product ID to search using Binary Search:");
33         int productId = Integer.parseInt(scanner.nextLine());
34
35         Product[] products = {
36             new Product(101, "T-shirt", "Clothing"),
37             new Product(102, "Jeans", "Clothing"),
38             new Product(103, "Sneakers", "Footwear"),
39             new Product(104, "Backpack", "Accessories"),
40             new Product(105, "Kurta Set", "Clothing"),
41             new Product(106, "Sunglasses", "Accessories"),
42             new Product(107, "Smartwatch", "Wearables"),
43             new Product(108, "Wallet", "Accessories"),
44             new Product(109, "Cap", "Accessories"),
45             new Product(110, "Shower Cap", "Accessories")
46         };
47
48         Arrays.sort(products, Comparator.comparingInt(Product::productId));
49
50         Product foundProduct = binarySearch(products, productId);
51
52         if (foundProduct != null) {
53             System.out.println("Product Found: " + foundProduct.toString());
54         } else {
55             System.out.println("Product not found.");
56         }
57     }
58
59     private static Product binarySearch(Product[] products, int productId) {
60         int left = 0;
61         int right = products.length - 1;
62
63         while (left <= right) {
64             int mid = (left + right) / 2;
65             Product midProduct = products[mid];
66
67             if (midProduct.productId == productId) {
68                 return midProduct;
69             } else if (midProduct.productId < productId) {
69                 left = mid + 1;
70             } else {
71                 right = mid - 1;
72             }
73         }
74
75         return null;
76     }
77 }
```

Enter the product ID to search using Binary Search: 105
Product Found: [105, Kurta Set, Clothing]

...Program finished with exit code 0
Press ENTER to exit console

Exercise 7: Financial Forecasting

Forecasting.java

//to calculate future value using $FV = PV \cdot (1+r)^n$

```
import java.util.*;
```

```
public class Forecasting
```

```
{
```

```
    public static double forecast(double presentValue, double rate, int years){
```

```
        if(years==0){
```

```
            return presentValue;
```

```
        }
```

```
        return forecast(presentValue,rate,years-1)*(1+rate);
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        System.out.println("Enter the present Value : ");
```

```

Scanner s = new Scanner(System.in);

double presentValue=s.nextDouble();

System.out.println("Enter the annual growth rate : ");

double rate = s.nextDouble(); //annual growth rate

System.out.print("Enter the no. of years : ");

int years=s.nextInt();

double futureValue=forecast(presentValue,rate,years);

System.out.printf("Future value after %d years : $%.3f\n", years,
futureValue);

//System.out.println("Future value after "+years+" years : $" +futureValue);

}

}

```

OUTPUT:

The screenshot shows a web browser window with the URL 'onlinegdb.com/online_java_compiler#'. The browser's address bar and tabs are visible at the top. Below the browser window is a code editor with a dark background and light-colored text. The code is a Java program named 'Forecasting.java'. It defines a static method 'forecast' that takes three parameters: 'double presentValue', 'double rate', and 'int years'. The method uses a recursive-like approach to calculate the future value. The 'main' method uses a 'Scanner' to take input from the user for the present value, annual growth rate, and number of years. It then calls the 'forecast' method and prints the result using 'printf'. The output window at the bottom shows the program's execution with the following input and output:

```

Enter the present Value :
10506
Enter the annual growth rate :
0.17
Enter the no. of years : 7
Future value after 7 years : $31531.050

...Program finished with exit code 0
Press ENTER to exit console.

```

OUTPUT

Enter the present Value :

10506

Enter the annual growth rate :

0.17

Enter the no. of years : 7

Future value after 7 years : \$31531.050