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

**Program :**

import java.util.\*;

class Product{

    public int productID;

    public String productName;

    public String category;

    Product(int id,String name,String category){

        this.productID = id;

        this.productName = name.toLowerCase();

        this.category = category.toLowerCase();

    }

    public String toString(){

         return productID + " - " + productName + " - " + category ;

    }

}

class LinearSearchEngine{

    public Product LinearSearchID(Product []prod,int id){

        for(int i=0;i<prod.length;i++){

            if(prod[i].productID==id){

                return prod[i];

            }

        }

        return null;

    }

    public Product LinearSearchName(Product []prod,String name){

        for(int i=0;i<prod.length;i++){

            if(prod[i].productName.equals(name)){

                return prod[i];

            }

        }

        return null;

    }

    public Product LinearSearchCat(Product []prod,String cat){

        for(int i=0;i<prod.length;i++){

            if(prod[i].category.equals(cat)){

                return prod[i];

            }

        }

        return null;

    }

}

class BinarySearchEngine{

    public Product BinarySearchID(Product []prod,int id){

        Arrays.sort(prod,(a,b)->{

            return a.productID - b.productID;

        });

        int i=0,j=prod.length-1;

        while(i<=j){

            int mid = (i+j)/2;

            if(prod[mid].productID==id){

                return prod[mid];

            }

            else if(prod[mid].productID < id){

                i = mid+1;

            }

            else{

                j = mid-1;

            }

        }

        return null;

    }

    public Product BinarySearchName(Product []prod,String name){

        Arrays.sort(prod,(a,b)->{

            return a.productName.compareTo(b.productName);

        });

        int i=0,j=prod.length-1;

        while(i<=j){

            int mid = (i+j)/2;

            int cmp = prod[mid].productName.compareTo(name);

            if(cmp==0){

                return prod[mid];

            }

            else if(cmp<0){

                i = mid+1;

            }

            else{

                j = mid-1;

            }

        }

        return null;

    }

    public Product BinarySearchCat(Product []prod,String cat){

        Arrays.sort(prod,(a,b)->{

            return a.category.compareTo(b.category);

        });

        int i=0,j=prod.length-1;

        while(i<=j){

            int mid = (i+j)/2;

            int cmp = prod[mid].category.compareTo(cat);

            if(cmp==0){

                return prod[mid];

            }

            else if(cmp<0){

                i = mid+1;

            }

            else{

                j = mid-1;

            }

        }

        return null;

    }

}

public class Main {

    public static void main(String[] args) {

        Product []prod = {

            new Product(1,"Sparx", "Fashion"),

            new Product(2,"Adidas", "Fashion"),

            new Product(3,"Jordan", "Fashion"),

            new Product(4,"Laptop", "Electronics"),

            new Product(5,"Phone", "Electronics"),

            new Product(6,"Shirt", "Casuals"),

            new Product(7,"Jeans", "Casuals"),

        };

        LinearSearchEngine linear = new LinearSearchEngine();

        BinarySearchEngine binary = new BinarySearchEngine();

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

        System.out.println("By id :"+linear.LinearSearchID(prod, 5));

        System.out.println("By name :"+linear.LinearSearchName(prod, "jeans"));

        System.out.println("By cat :"+linear.LinearSearchCat(prod, "fashion"));

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

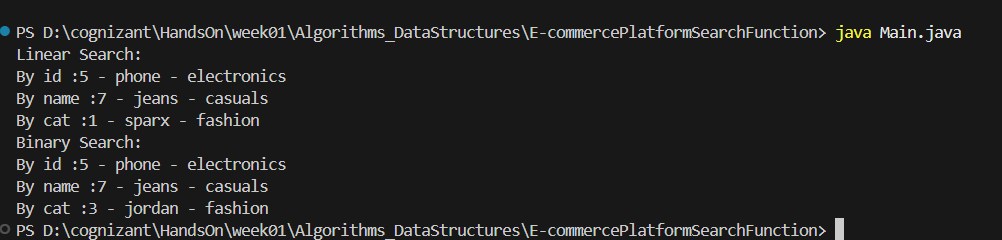
        System.out.println("By id :"+binary.BinarySearchID(prod, 5));

        System.out.println("By name :"+binary.BinarySearchName(prod, "jeans"));

        System.out.println("By cat :"+binary.BinarySearchCat(prod, "fashion"));

    }

}



Big O notation is an asymptotic notation which is used for space or time complexity. It describes upper limit of the function . It is generally used in worst case scenarios.

Ex : f(n) = n^2 + 2n + 2

O(f(n)) = n^2

It helps to

Compare algorithms (which one is faster or more efficient)

Predict performance for large inputs

Identify bottlenecks

Optimize code