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

**Implementation:**

**CODE:**

**Product.java:**

public class Product implements Comparable<Product>{

    int productId;

    String productName;

    String category;

    public Product(){}

    public Product(int productId,String productName,String category)

    {

        this.productId=productId;

        this.productName=productName;

        this.category=category;

    }

      @Override

    public int compareTo(Product other) {

        return this.productName.compareToIgnoreCase(other.productName); // sort by name

    }

    public int getProductId() {

        return productId;

    }

    public void setProductId(int productId) {

        this.productId = productId;

    }

    public String getProductName() {

        return productName;

    }

    public void setProductName(String productName) {

        this.productName = productName;

    }

    public String getCategory() {

        return category;

    }

    public void setCategory(String category) {

        this.category = category;

    }

    @Override

    public String toString()

    {

        return "Product Id: "+productId+"\nProduct Name: "+productName+"\nCategory: "+category;

    }

}

**LinearSearch.java:**

public class LinearSearch {

    public  Product linearSearch(Product[] products,String target)

    {

        for(Product p:products)

        {

            if(p.getProductName().equalsIgnoreCase(target))

            {

                return p;

            }

        }

        return null;

    }

}

**BinarySearch.java:**

public class BinarySearch {

    public Product binarySearch(Product[] products,String target)

    {

        int l=0,r=products.length-1;

        while(l<=r)

        {

            int mid=(l+r)/2;

            String name=products[mid].getProductName();

            int com=name.compareToIgnoreCase(target);

            if(com==0)

            {

                return products[mid];

            }

            else if(com<0)

            {

                l=mid+1;

            }

            else

            {

                r=mid-1;

            }

        }

        return null;

    }

}

**Main.java:**

import java.util.Arrays;

public class Main {

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        Product[] products= {new Product(101,"Laptop","Electronics"),

                new Product(102,"Phone","Electronics"),

                new Product(103,"Chair","Furniture"),

                new Product(104,"Books","Education"),

                new Product(105,"Necklace","Accessories"),

                new Product(106,"Frock","Dress")};

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

        LinearSearch l=new LinearSearch();

        System.out.println(l.linearSearch(products,"Chair"));

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

        BinarySearch b=new BinarySearch();

        //Arrays.sort(products,Comparator.comparing(Product :: getProductName));

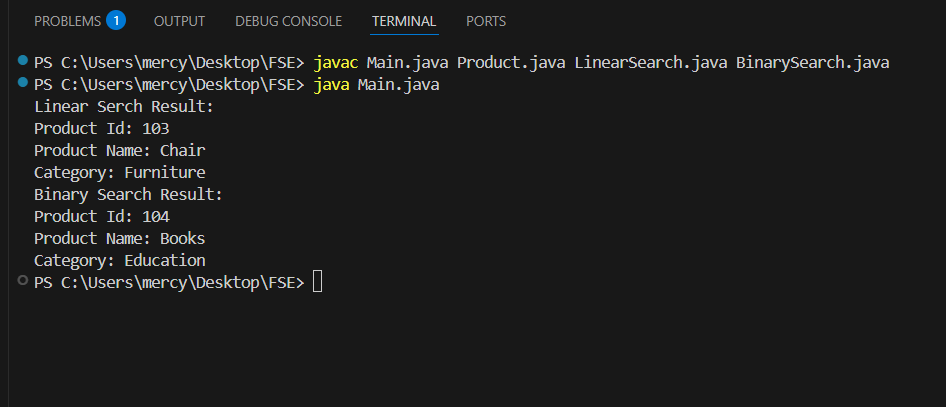
        Arrays.sort(products);

        System.out.println(b.binarySearch(products,"Books"));

    }

}

**OUTPUT:**

****

**Analysis:**

Linear Search has a time complexity of O(n) which is slower in case of large data set. It doesn’t require sorted order. So linear search can be used for small datasets. When it comes to Ecommerce, the dataset is large. So, we can use Binary Search of time complexity O(log n) which makes it to work faster. But the data need to be sorted.

**Exercise 7: Financial Forecasting**

**Implementation:**

**CODE:**

**FinancialForecast.java:**

public class FinancialForecast {

    public static double Recursion(double principal,double growth,int years)

    {

        if(years ==0) return principal;

        return Recursion(principal,growth,years-1)\*(1+growth);

    }

    public static double Iteration(double principal,double growth,int years)

    {

        double res=principal;

        for(int i=1;i<=years;i++)

        {

            res\*=(1+growth);

        }

        return res;

    }

    public static double Memoize(double principal,double growth,int years,Double[] memo)

    {

        if(years==0) return principal;

        if(memo[years]!=null) return memo[years];

        memo[years]=Memorize(principal, growth, years-1, memo)\*(1+growth);

        return memo[years];

    }

    public static void main(String args[])

    {

        double principal=1000.0;

        double growth=0.1;

        int years=6;

        System.out.println("Financial Forecast using Recursion:"+Recursion(principal, growth, years));

        System.out.println("Financial forecast using Iteration: "+Iteration(principal, growth, years));

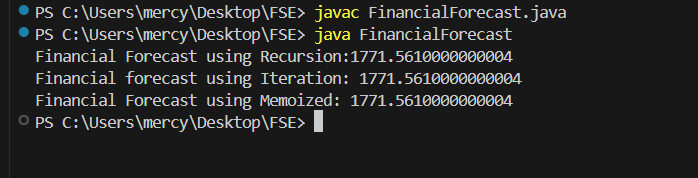
        Double memo[]=new Double[years+1];

        System.out.println("Financial Forecast using Memoized: "+Memoize(principal, growth, years, memo));

    }

}

**OUTPUT:**

****

**Analysis:**

The implementation of Recursion is simple when compared to Iteration, Memoized. But the Performance of Recursion is slow and it also has the risk of stack overflow. The performance of Iterative and Memoized is fast. They can be used for better results.

Time complexity of Recursion: O (2 n)

Time complexity of Iteration: O (n)

Time complexity of Memoization: O (n)