Module2-DSA

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

Product.java

package ecommerce;

public class product implements Comparable<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 int compareTo(product other){

        return this.productname.compareToIgnoreCase(other.productname);

    }

    public String toString(){

        return productid+ ": "+ productname +" ("+ category +")";

    }

}

Linearseacrh.java

package ecommerce;

public class linearsearch {

    public static product search(product[] products, String targetname) {

        for (product p : products) {

            if (p.productname.equalsIgnoreCase(targetname)) {

                return p;

            }

        }

        return null;

    }

}

Binarysearch.java

package ecommerce;

public class binarysearch {

    public static product search(product[] products, String targetname) {

        int low = 0, high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

            int cmp = products[mid].productname.compareToIgnoreCase(targetname);

            if (cmp == 0)

                return products[mid];

            else if (cmp < 0)

                low = mid + 1;

            else

                high = mid - 1;

        }

        return null;

    }

}

Ecommercesearch.java

package ecommerce;

import java.util.Arrays;

public class ecommercesearch {

     public static void main(String[] args) {

        product[] products = {

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

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

            new product(3, "Keyboard", "Electronics"),

            new product(4, "Book", "Education")

        };

        product result1 = linearsearch.search(products, "Keyboard");

        System.out.println("Linear Search Result: " + (result1 != null ? result1 : "product not found"));

        Arrays.sort(products);

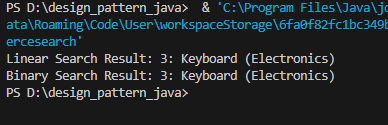
        product result2 = binarysearch.search(products, "Keyboard");

        System.out.println("Binary Search Result: " + (result2 != null ? result2 : "product not found"));

    }

}

Output:



If data is dynamic and frequently updated -Linear Search is simpler.

If data is mostly static or pre-processed - Binary Search provides faster lookups, especially with large product catalogs.

**Exercise 7: Financial Forecasting**

Financial.java

package forecasting;

public class financial {

    public static double forecastRecursive(double currentValue, double growthRate, int years) {

        if (years == 0) {

            return currentValue;

        }

        return forecastRecursive(currentValue, growthRate, years - 1) \* (1 + growthRate);

    }

    public static double forecastMemo(double currentValue, double growthRate, int years, double[] memo) {

        if (years == 0) return currentValue;

        if (memo[years] != 0) return memo[years];

        memo[years] = forecastMemo(currentValue, growthRate, years - 1, memo) \* (1 + growthRate);

        return memo[years];

    }

    public static void main(String[] args) {

        double currentValue = 10000;

        double growthRate = 0.08;

        int years = 5;

        double forecast = forecastRecursive(currentValue, growthRate, years);

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

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

        double forecastMemoized = forecastMemo(currentValue, growthRate, years, memo);

        System.out.printf("Memoized Prediction after %d years: %.2f\n", years, forecastMemoized);

    }

}

Output:

