**Data structures and Algorithms**

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

**Program.cs:**

using System;

class Program

{

    static void Main()

    {

        Product[] unsortedProducts = {

            new Product(101, "Laptop", "Electronics"),

            new Product(102, "Shoes", "Fashion"),

            new Product(103, "Phone", "Electronics"),

            new Product(104, "Watch", "Accessories")

        };

        Product[] sortedProducts = {

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

            new Product(104, "Shoes", "Fashion"),

            new Product(101, "Watch", "Accessories"),

            new Product(103, "Zipper", "Clothing")

        };

        Array.Sort(sortedProducts, (a, b) => a.ProductName.CompareTo(b.ProductName));

        string searchTerm = "Watch";

        Console.WriteLine("=== Linear Search ===");

        var result1 = SearchAlgorithms.LinearSearch(unsortedProducts, searchTerm);

        Console.WriteLine(result1 != null ? $"Found: {result1}" : "Not found");

        Console.WriteLine("\n=== Binary Search ===");

        var result2 = SearchAlgorithms.BinarySearch(sortedProducts, searchTerm);

        Console.WriteLine(result2 != null ? $"Found: {result2}" : "Not found");

    }

}

**Product.cs**

public class Product

{

    public int ProductId { get; set; }

    public string ProductName { get; set; }

    public string Category { get; set; }

    public Product(int id, string name, string category)

    {

        ProductId = id;

        ProductName = name;

        Category = category;

    }

    public override string ToString()

    {

        return $"{ProductId} - {ProductName} ({Category})";

    }

}

**SearchAlgorithms.cs:**

using System;

public class SearchAlgorithms

{

    public static Product LinearSearch(Product[] products, string name)

    {

        foreach (var product in products)

        {

            if (product.ProductName.Equals(name, StringComparison.OrdinalIgnoreCase))

                return product;

        }

        return null;

    }

    public static Product BinarySearch(Product[] products, string name)

    {

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

        while (left <= right)

        {

            int mid = (left + right) / 2;

            int cmp = string.Compare(products[mid].ProductName, name, StringComparison.OrdinalIgnoreCase);

            if (cmp == 0) return products[mid];

            else if (cmp < 0) left = mid + 1;

            else right = mid - 1;

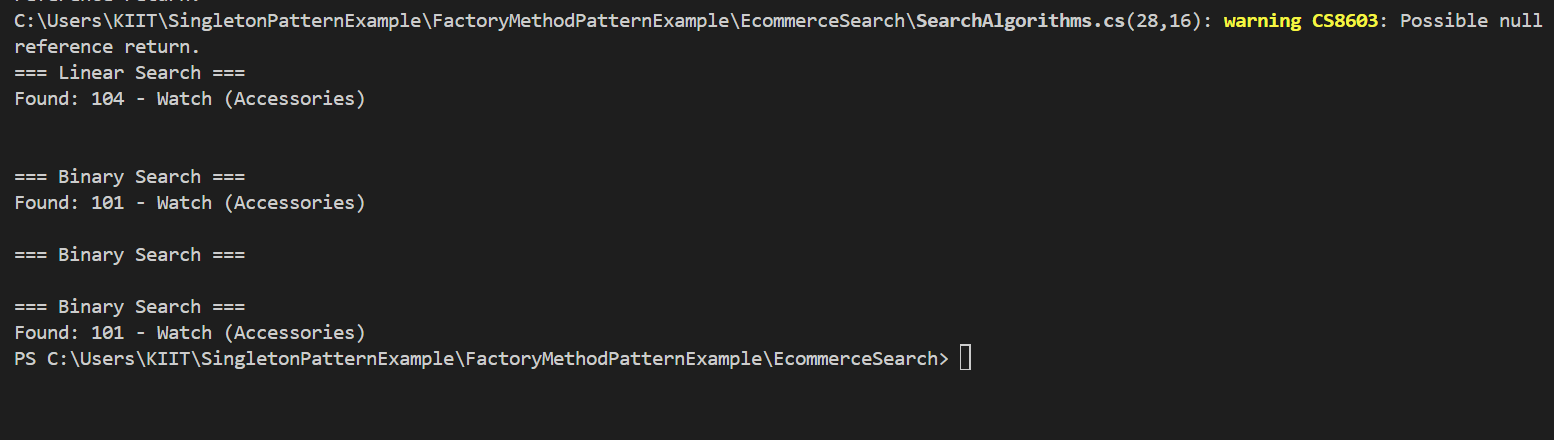
        }

        return null;

    }

}

Output:



**Exercise 7: Financial Forecasting**

**Program.cs**

using System;

class Program

{

    static void Main()

    {

        double initialValue = 1000;       // Starting investment

        double growthRate = 0.10;         // 10% growth per year

        int years = 5;

        double futureValue = CalculateFutureValue(initialValue, growthRate, years);

        Console.WriteLine($"Future Value after {years} years: ₹{futureValue:F2}");

    }

    static double CalculateFutureValue(double value, double rate, int years)

    {

        if (years == 0)

            return value;

        return CalculateFutureValue(value, rate, years - 1) \* (1 + rate);

    }

}

Output: 