1. **E-commerce Platform Search Function**

**import java.util.\*;**

**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;**

**}**

**@Override**

**public int compareTo(Product other) {**

**return this.productId - other.productId;**

**}**

**@Override**

**public String toString() {**

**return "Product ID: " + productId + ", Name: " + productName + ", Category: " + category;**

**}**

**}**

**public class EcommerceSearchFunction {**

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

**for (Product product : products) {**

**if (product.productId == targetId) {**

**return product;**

**}**

**}**

**return null;**

**}**

**public static Product binarySearch(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];**

**if (products[mid].productId < targetId) left = mid + 1;**

**else right = mid - 1;**

**}**

**return null;**

**}**

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

**Product[] productList = new Product[] {**

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

**new Product(205, "Shoes", "Fashion"),**

**new Product(150, "Phone", "Electronics"),**

**new Product(112, "Book", "Education"),**

**new Product(299, "Watch", "Accessories")**

**};**

**Product[] sortedList = productList.clone();**

**Arrays.sort(sortedList);**

**int searchId = 150;**

**System.out.println("Searching for product with ID: " + searchId);**

**long startTime = System.nanoTime();**

**Product resultLinear = linearSearch(productList, searchId);**

**long endTime = System.nanoTime();**

**System.out.println("Linear Search Result: " + resultLinear);**

**System.out.println("Time taken (ns): " + (endTime - startTime));**

**startTime = System.nanoTime();**

**Product resultBinary = binarySearch(sortedList, searchId);**

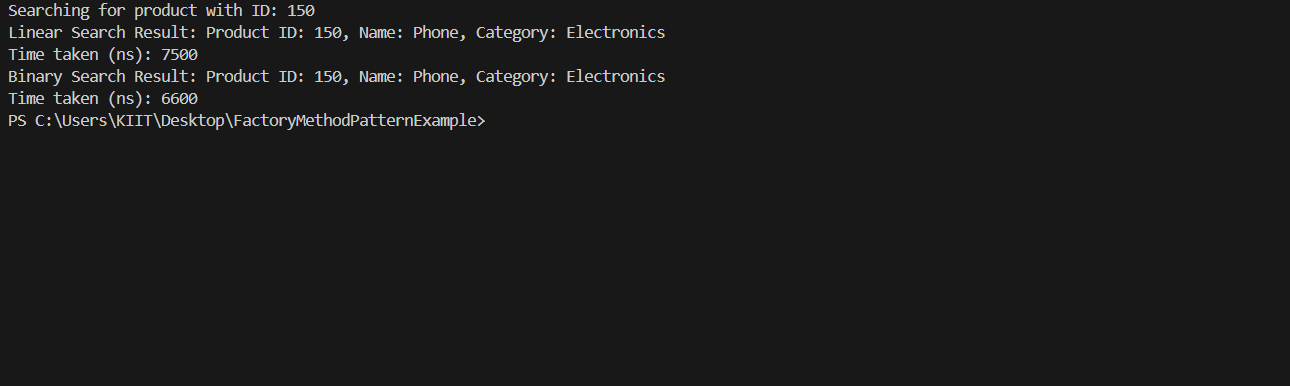
**endTime = System.nanoTime();**

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

**System.out.println("Time taken (ns): " + (endTime - startTime));**

**}**

**}**

****

1. **Financial Forecasting**

**public class FinancialForecasting {**

**public static double calculateFutureValue(double presentValue, double rate, int periods) {**

**if (periods == 0) return presentValue;**

**return (1 + rate) \* calculateFutureValue(presentValue, rate, periods - 1);**

**}**

**public static double calculateFutureValueMemo(double presentValue, double rate, int periods, Double[] memo) {**

**if (periods == 0) return presentValue;**

**if (memo[periods] != null) return memo[periods];**

**memo[periods] = (1 + rate) \* calculateFutureValueMemo(presentValue, rate, periods - 1, memo);**

**return memo[periods];**

**}**

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

**double presentValue = 1000.0;**

**double annualGrowthRate = 0.05;**

**int years = 10;**

**System.out.println("--- Financial Forecasting ---");**

**double futureValue = calculateFutureValue(presentValue, annualGrowthRate, years);**

**System.out.printf("Future Value (Recursive): $%.2f\n", futureValue);**

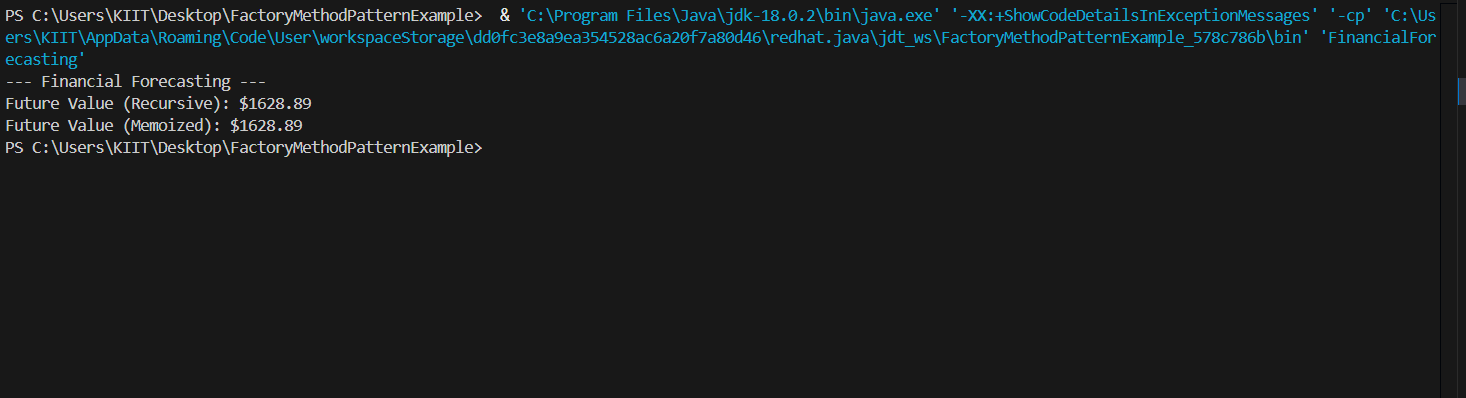
**Double[] memo = new Double[years + 1];**

**double futureValueMemo = calculateFutureValueMemo(presentValue, annualGrowthRate, years, memo);**

**System.out.printf("Future Value (Memoized): $%.2f\n", futureValueMemo);**

**}**

**}**

****