**Week1**

**Data Structures And Algorithms**

**HandsOn**

**1.Search Products:**

**Code:**

import java.util.Arrays;

class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

}

public class SearchProducts {

public static int linearSearch(Product[] products, String key) {

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

if (products[i].productName.equalsIgnoreCase(key)) {

return i;

}

}

return -1;

}

public static int binarySearch(Product[] products, String key) {

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

while (low <= high) {

int mid = (low + high) / 2;

int cmp = products[mid].productName.compareToIgnoreCase(key);

if (cmp == 0) return mid;

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

else high = mid - 1;

}

return -1;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Book", "Education"),

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

new Product(104, "Phone", "Electronics")

};

String searchKey = "Shirt";

int index = linearSearch(products, searchKey);

System.out.println("[Linear] Found at index: " + index);

Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

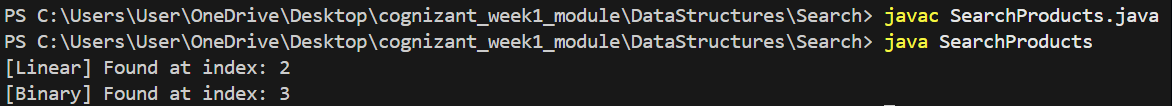
index = binarySearch(products, searchKey);

System.out.println("[Binary] Found at index: " + index);

}

}

**Output:**



**2.Recursion (Forecast):**

**Code:**

public class Forecast {

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

if (years == 0) return currentValue;

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

}

public static void main(String[] args) {

double currentValue = 10000;

double growthRate = 0.10;

int years = 5;

double futureValue = predictFutureValue(currentValue, growthRate, years);

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

}

}

**Output:**

