**Digital Nurture 4.0**

Week 1-Data structures and Algorithms

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

**Program:**

Here, I created a project and class named EcommerceSearch. I defined a Product class and stored product data in an array for linear search and a sorted array for binary search. Then I implemented both search algorithms with a step count feature. Finally, the result and number of steps taken for each algorithm were displayed to compare their performance.

**EcommerceSearch.java**

import java.util.\*;

public class EcommerceSearch {

    static class Product {

        int productId;

        String productName;

        String category;

        Product(int id, String name, String cat) {

            productId = id;

            productName = name;

            category = cat;

        }

    }

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

    int steps = 0;

    for (Product p : products) {

        steps++;

        if (p.productId == targetId) {

            System.out.println(" Linear Search: Product Found;\n ID: " + p.productId + "; Category: " + p.category + "; Steps: " + steps);

            return p;

        }

    }

    System.out.println(" Linear Search: Product not found steps: " + steps);

    return null;

}

static Product binarySearch(Product[] products, int targetId) {

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

    while (low <= high) {

        steps++;

        int mid = (low + high) / 2;

        if (products[mid].productId == targetId) {

            System.out.println(" Binary Search: Product Found;\n ID: " + products[mid].productId + "; Category: " + products[mid].category + "; Steps: " + steps);

            return products[mid];

        } else if (products[mid].productId < targetId) {

            low = mid + 1;

        } else {

            high = mid - 1;

        }

    }

    System.out.println(" Binary Search: Product not found steps: " + steps);

    return null;

}

    public static void main(String[] args) {

        Product[] products = {

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

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

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

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

            new Product(105, "Bag", "Travel")

        };

        Product[] sortedProducts = Arrays.copyOf(products, products.length);

        Arrays.sort(sortedProducts, Comparator.comparingInt(p -> p.productId));

        int targetId = 103;

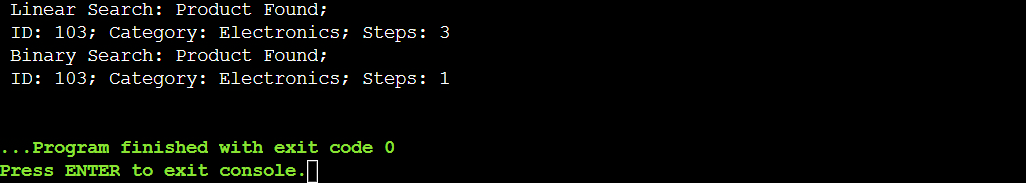
        Product linearResult = linearSearch(products, targetId);

        Product binaryResult = binarySearch(sortedProducts, targetId);

        }

    }

**Output:**

****

**Exercise 7: Financial Forecasting**

**Program:**

In this project, I created Forecast class, calculates the future value of a fixed deposit using recursion. It takes the principal amount, interest rate, and years as input, then recursively computes the compounded value and displays the final amount.

**Forecast.java:**

import java.util.\*;

public class Forecast {

static double calculateFD(double principal, double rate, int years) {

if (years == 0) return principal;

return calculateFD(principal, rate, years - 1) \* (1 + rate);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print(" Enter initial fixed deposit amount: ");

double principal = sc.nextDouble();

System.out.print(" Enter annual interest rate (e.g., 0.07 for 7%): ");

double rate = sc.nextDouble();

System.out.print(" Enter number of years: ");

int years = sc.nextInt();

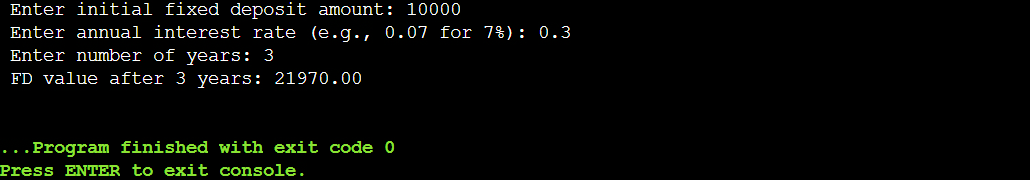
double result = calculateFD(principal, rate, years);

System.out.printf(" FD value after %d years: %.2f\n", years, result);

}

}

**Output:**

****