**Week 1- Data Structures and Alogorithms- Hands-On Exercises**

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

**Code:**

EcommerceSearch.java

import java.util.Arrays;

public class ECommerceSearch {

    private Product[] products;

    public ECommerceSearch(Product[] products) {

        this.products = products;

    }

    public Product linearSearch(String productId) {

        for (Product product : products) {

            if (product.getProductId().equals(productId)) {

                return product;

            }

        }

        return null;

    }

    public Product binarySearch(String productId) {

        int left = 0;

        int right = products.length - 1;

        while (left <= right) {

            int mid = left + (right - left) / 2;

            if (products[mid].getProductId().equals(productId)) {

                return products[mid];

            } else if (products[mid].getProductId().compareTo(productId) < 0) {

                left = mid + 1;

            } else {

                right = mid - 1;

            }

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

            new Product("P100", "Laptop", "Electronics"),

            new Product("P200", "Smartphone", "Electronics"),

            new Product("P300", "Tablet", "Electronics"),

            new Product("P400", "Headphones", "Accessories"),

            new Product("P500", "Charger", "Accessories")

        };

        Arrays.sort(products, (a, b) -> a.getProductId().compareTo(b.getProductId()));

        ECommerceSearch search = new ECommerceSearch(products);

        System.out.println("=== Linear Search ===");

        Product foundProduct = search.linearSearch("P200");

        System.out.println(foundProduct != null ? foundProduct : "Product not found");

        System.out.println("=== Binary Search ===");

        foundProduct = search.binarySearch("P200");

        System.out.println(foundProduct != null ? foundProduct : "Product not found");

    }

}

Product.java

public class Product {

    private String productId;

    private String productName;

    private String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    public String getProductId() {

        return productId;

    }

    public String getProductName() {

        return productName;

    }

    public String getCategory() {

        return category;

    }

    @Override

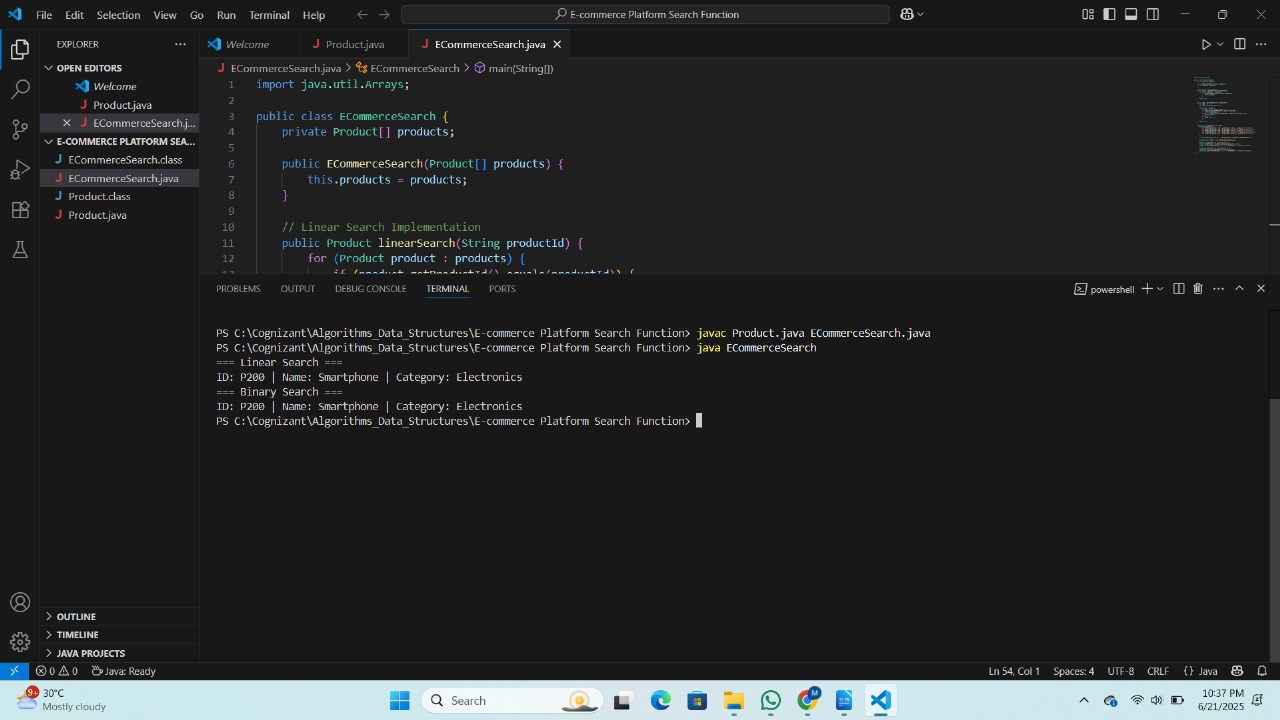
    public String toString() {

        return String.format("ID: %s | Name: %s | Category: %s", productId, productName, category);

    }

}

Output:

  
  
**Exercise 7: Financial Forecasting**

**Code:**

FinancialForecasting.java

public class FinancialForecasting {

    public static double calculateFutureValue(double principal, double growthRate, int years) {

        if (years == 0) {

            return principal;

        }

        return calculateFutureValue(principal \* (1 + growthRate), growthRate, years - 1);

    }

    public static void main(String[] args) {

        double principal = 1000.0;

        double growthRate = 0.05;

        int years = 10;

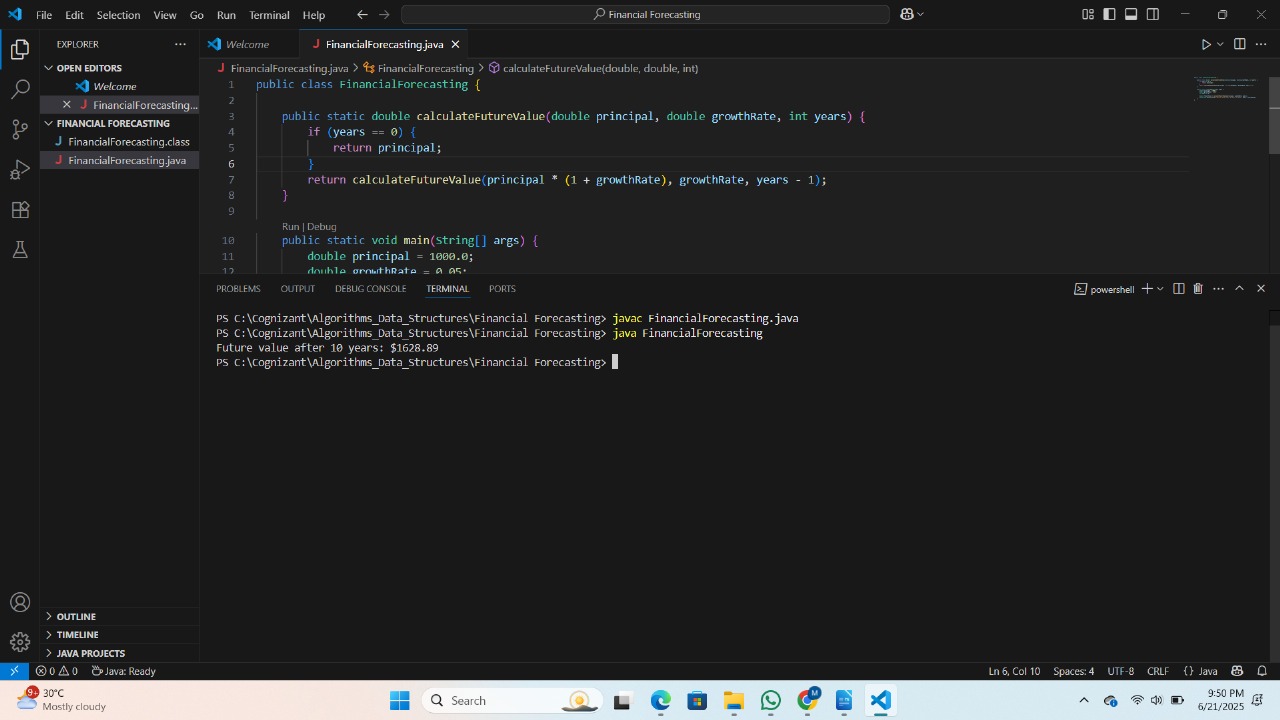
        double futureValue = calculateFutureValue(principal, growthRate, years);

        System.out.printf("Future value after %d years: $%.2f%n", years, futureValue);

    }

}

**Output:**

****