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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**Code :   
  
BinarySearch.java :**package ecommerceplatformsearch;

public class BinarySearch {

public static void sortByName(Product[] products)

{

int n = products.length;

for(int i=0;i<n;i++)

{

for(int j=0;j<n-1-i;j++)

{

if(products[j].productName.compareToIgnoreCase(products[j+1].productName) > 0)

{

Product temp = products[j];

products[j] = products[j+1];

products[j+1] = temp;

}

}

}

}

public static Product binarySearch(Product[] products, String productName)

{

int start = 0;

int end = products.length - 1;

while(start <= end)

{

int mid = (start + end) / 2;

String midName = products[mid].productName;;

if(midName.equalsIgnoreCase(productName))

{

return products[mid];

}

else if(productName.compareToIgnoreCase(midName) > 0)

{

start = mid + 1;

}

else

{

end = mid - 1;

}

}

return null;

}

}

**LinearSearch.java :**

package ecommerceplatformsearch;

public class LinearSearch {

public static Product linearSearch(Product[] products, String productName)

{

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

{

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

{

return products[i];

}

}

return null;

}

}

**Main.java :**package ecommerceplatformsearch;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Total Number of Products to be added : ");

int numberOfProducts = sc.nextInt();

sc.nextLine();

Product[] products = new Product[numberOfProducts];

for (int i = 0; i < numberOfProducts; i++) {

System.***out***.println("\nEnter details for product " + (i + 1) + " : ");

System.***out***.print("Product ID: ");

int id = sc.nextInt();

sc.nextLine();

System.***out***.print("Product Name: ");

String name = sc.nextLine();

System.***out***.print("Category: ");

String category = sc.nextLine();

products[i] = new Product(id, name, category);

}

System.***out***.print("\nEnter the Product Name to be searched: ");

String searchName = sc.nextLine();

System.***out***.println("\nLinear Search : ");

Product linearSearchResult = LinearSearch.*linearSearch*(products, searchName);

if (linearSearchResult != null) {

System.***out***.println("Product found: " + linearSearchResult.printProduct());

} else {

System.***out***.println("Product not found.");

}

BinarySearch.*sortByName*(products);

System.***out***.println("\nBinary Search : ");

Product binarySearchResult = BinarySearch.*binarySearch*(products, searchName);

if (binarySearchResult != null) {

System.***out***.println("Product found: " + binarySearchResult.printProduct());

} else {

System.***out***.println("Product not found.");

}

}

}

**Product.java :**package ecommerceplatformsearch;

public class Product {

int productId;

String productName;

String category;

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

{

this.productId = id;

this.productName = name;

this.category = category;

}

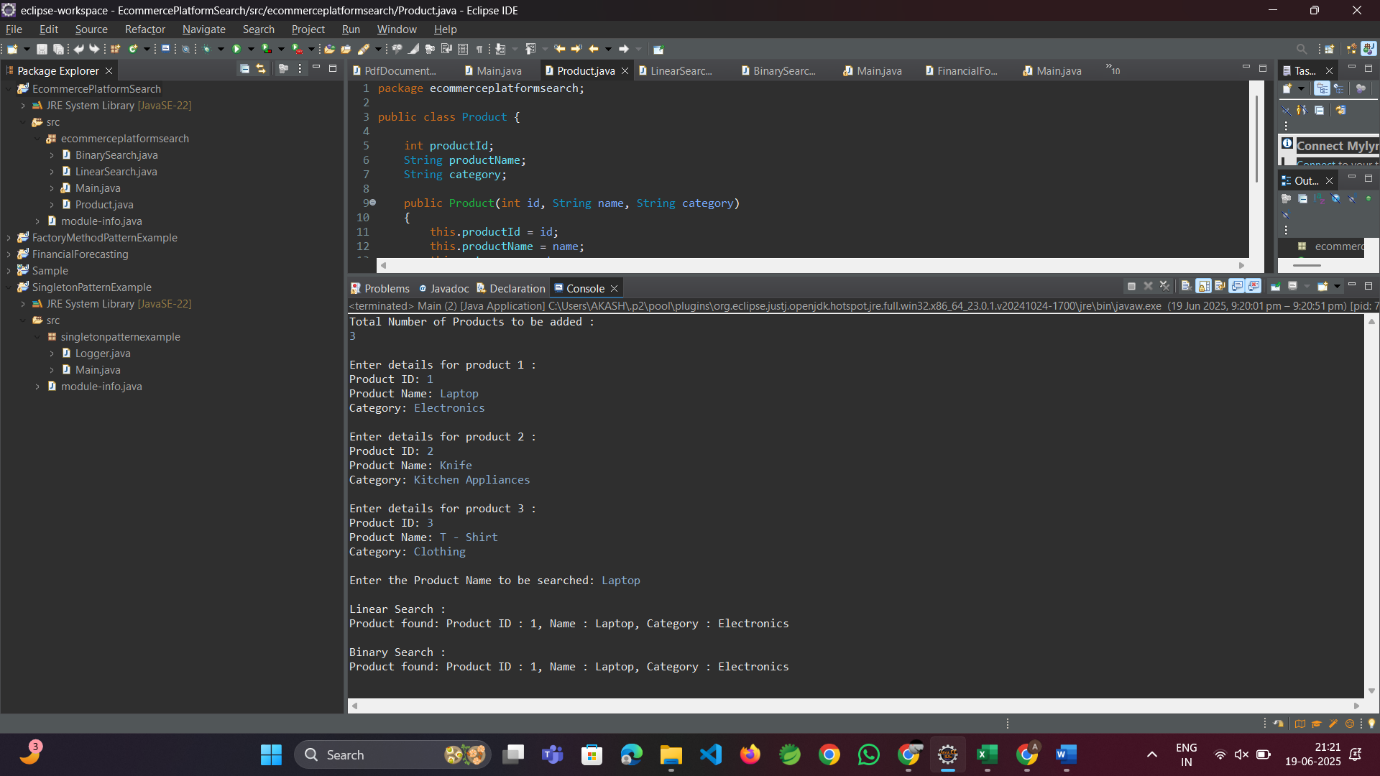
public String printProduct()

{

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

}

}

**Output Screenshots :**

**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

**Code :   
  
FinancialForecast.java :**package financialforecasting;

public class FinancialForecast {

public static double futureAmount(double amt, double rate, int year)

{

if(year == 0)

{

return amt;

}

else

{

return *futureAmount*(amt, rate, year - 1) \* (1 + rate);

}

}

}

**Main.java :**package financialforecasting;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter the Amount : ");

Double amt = sc.nextDouble();

System.***out***.println("Enter the growth rate : ");

Double rate = sc.nextDouble();

System.***out***.println("Enter the total number of years : ");

int year = sc.nextInt();

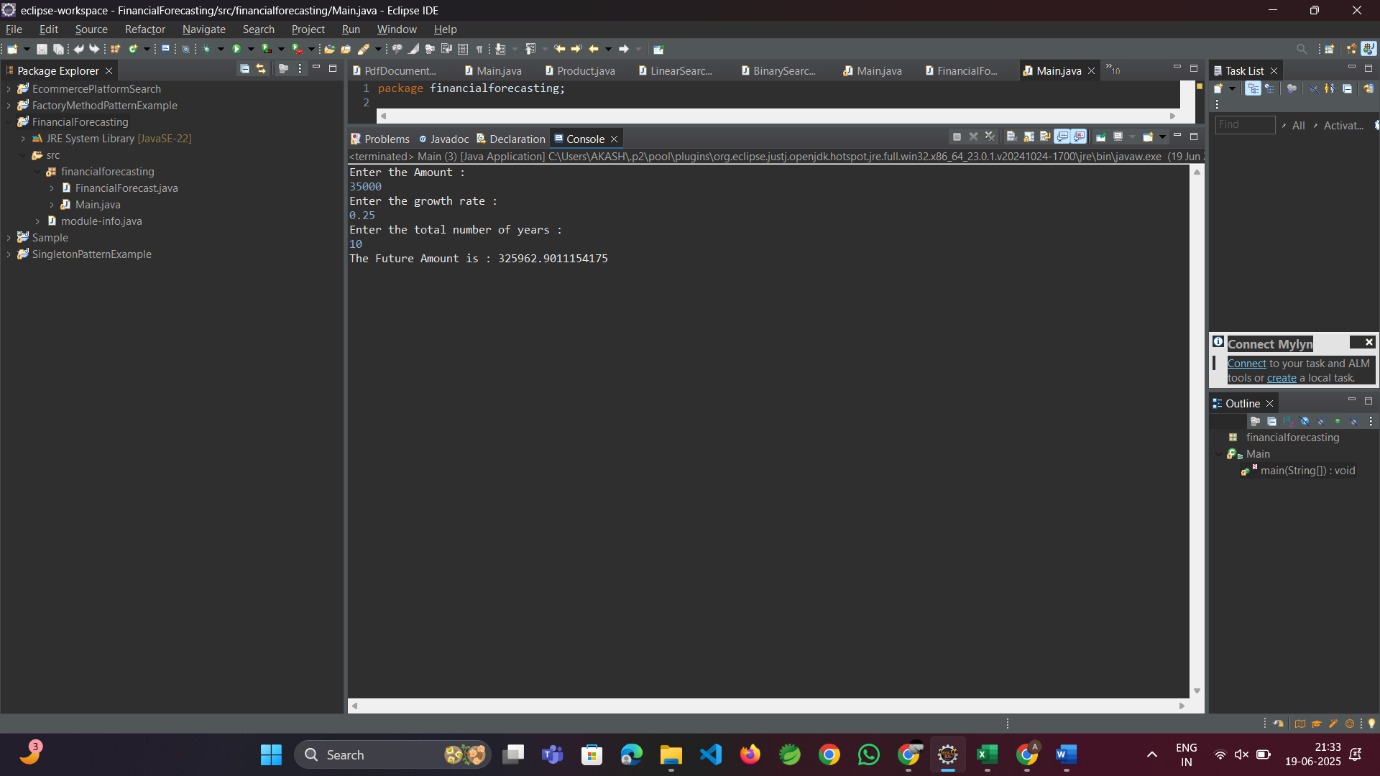
double totalAmount = FinancialForecast.*futureAmount*(amt, rate, year);

System.***out***.printf("The Future Amount is : " + totalAmount);

}

}

**Output Screenshots :**

****