# Week-1

## Data structures and Algorithms

## Exercise 2- E-commerce Platform Search Function

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

Code:

#### Product.java

public class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

#### SearchDemo.java

import java.util.Arrays;

import java.util.Comparator;

public class SearchDemo {

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

for (Product p : products) {

if (p.productId == id) return p;

}

return null;

}

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

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (products[mid].productId == id)

return products[mid];

else if (products[mid].productId < id)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] items = {

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

new Product(102, "Laptop", "Electronics"),

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

new Product(101, "Shoes", "Fashion"),

};

System.out.println("Linear search result:");

Product result1 = linearSearch(items, 104);

System.out.println(result1 != null ? result1 : "Not found");

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

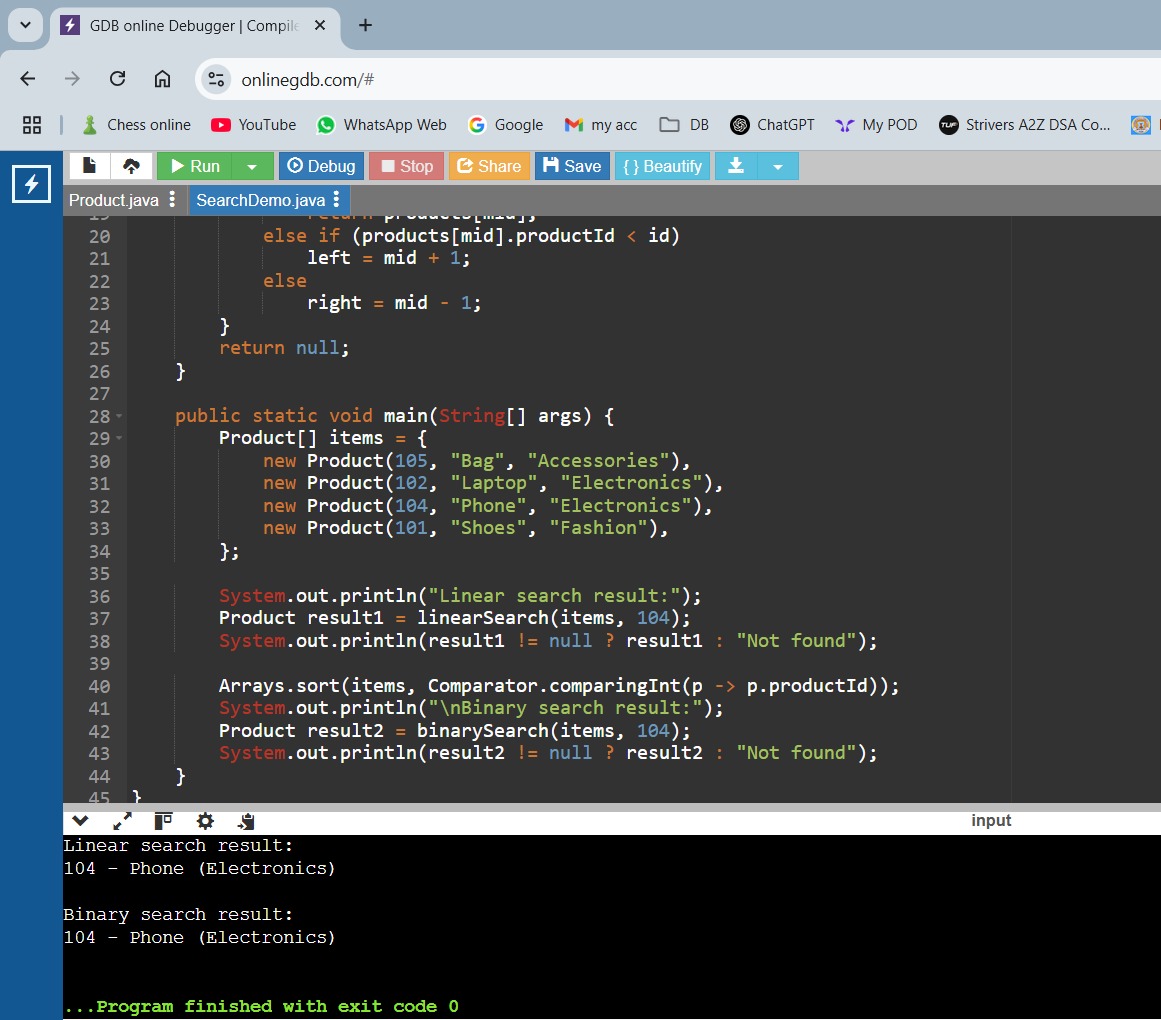
System.out.println("\nBinary search result:");

Product result2 = binarySearch(items, 104);

System.out.println(result2 != null ? result2 : "Not found");

}

}

Output Screenshot:

Output is:

Linear search result:

104 - Phone (Electronics)

Binary search result:

104 - Phone (Electronics)

#### Analysis:

* **Linear Search**-
  + Best: O(1) (first element)
  + Worst: O(n) (not found)
* **Binary Search**-
  + Always O(log n), but only works on sorted arrays.
* **Conclusion**-  
  Binary search is better for sorted and large data; linear is okay for small/unsorted lists.

### Exercise 7: Financial Forecasting

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

Code:

#### FinancialForecast.java

public class FinancialForecast {

public static double futureValue(double current, double rate, int years) {

if (years == 0) return current;

return futureValue(current, rate, years - 1) \* (1 + rate);

}

public static void main(String[] args) {

double value = 1000;

double growthRate = 0.05;

int duration = 6;

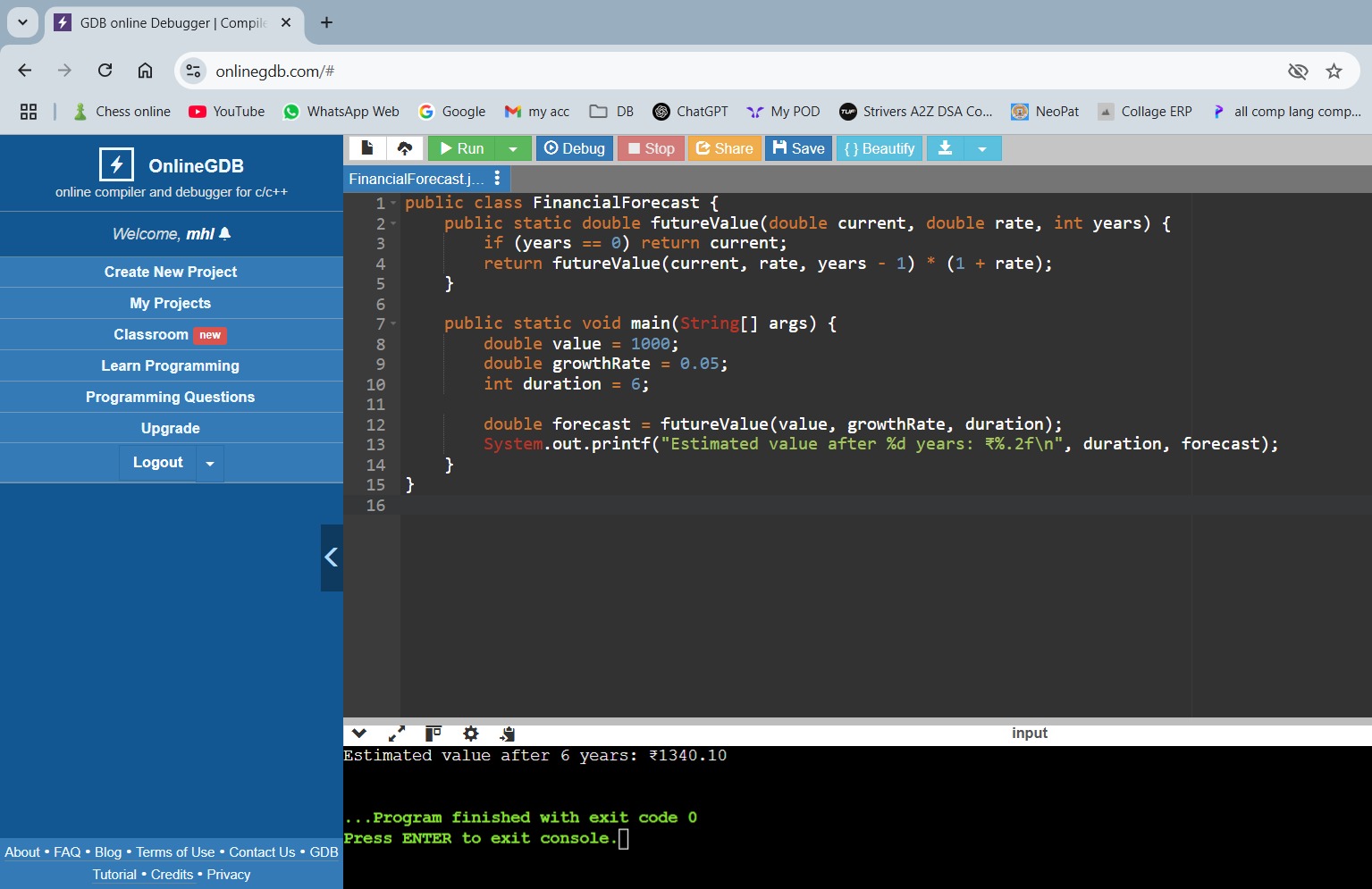
double forecast = futureValue(value, growthRate, duration);

System.out.printf("Estimated value after %d years: ₹%.2f\n", duration, forecast);

}

}

Output Screenshot:



Output is:

Estimated value after 6 years: ₹1340.10

#### Time Complexity:

* Recursive calls happen n times ⇒ **O(n)**.
* Each step is simple multiplication — no overlapping subproblems.