**Week-1**

**Algorithms Data Structures**

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

**Problem Statement:**

Designing a search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Product.java**

package com.search;

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;

}

}

**SearchAlgorithms.java**

package com.search;

public class SearchAlgorithms {

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

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

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

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

while (low <= high) {

int mid = (low + high) / 2;

int compare = products[mid].productName.compareToIgnoreCase(name);

if (compare == 0) {

return products[mid];

} else if (compare < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

}

**Main.java**

package com.search;

import java.util.Arrays;

import java.util.Comparator;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Footwear"),

new Product(3, "Watch", "Accessories"),

new Product(4, "Mobile", "Electronics")

};

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

Product result1 = SearchAlgorithms.linearSearch(products, "Watch");

if (result1 != null) {

System.out.println("Found: " + result1.productName);

} else {

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

}

Arrays.sort(products, Comparator.comparing(p -> p.productName));

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

Product result2 = SearchAlgorithms.binarySearch(products, "Watch");

if (result2 != null) {

System.out.println("Found: " + result2.productName);

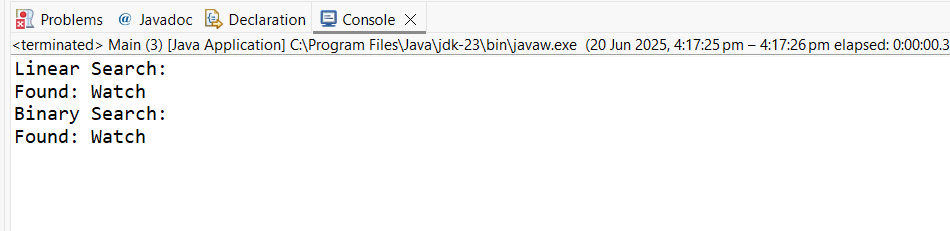
} else {

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

}

}

}

**OUTPUT:**

**Exercise 7: Financial Forecasting**

**Problem Statement:**

Developing a financial forecasting tool that predicts future values based on past data.

**Forecast.java**

package com.forecast;

public class Forecast {

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

if (years == 0) return currentAmount;

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

}

}

**Main.java**

package com.forecast;

public class Main {

public static void main(String[] args) {

double initialAmount = 10000;

double annualRate = 0.05;

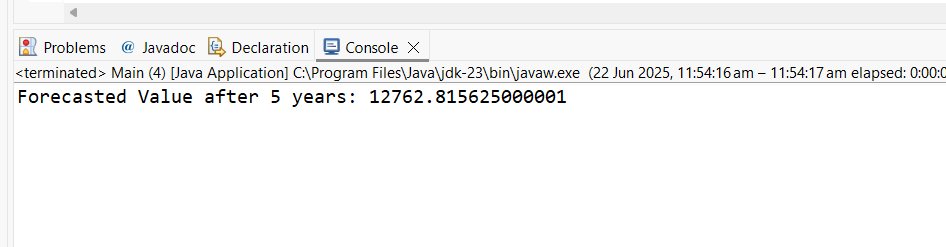
int years = 5;

double result = Forecast.*futureValue*(initialAmount, annualRate, years);

System.*out*.println("Forecasted Value after " + years + " years: " + result);

}

}

**OUTPUT**