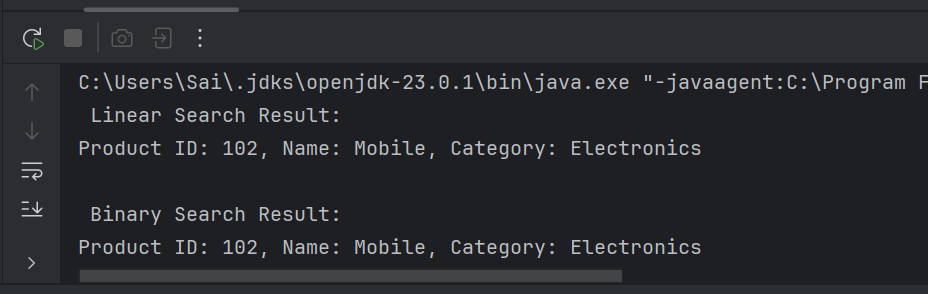
1 . E-commerce Platform Search Function

public class Week1 {  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(104, "Shoes", "Footwear"),  
 new Product(101, "Laptop", "Electronics"),  
 new Product(105, "Watch", "Accessories"),  
 new Product(102, "Mobile", "Electronics"),  
 new Product(103, "Shirt", "Apparel")  
 };  
  
 int searchId = 102;  
  
 // Linear Search  
 Product foundLinear = SearchEngine.*linearSearch*(products, searchId);  
 System.*out*.println(" Linear Search Result:");  
 if (foundLinear != null)  
 System.*out*.println(foundLinear);  
 else  
 System.*out*.println("Product not found.");  
  
 // Binary Search (after sorting)  
 SearchEngine.*sortByProductId*(products);  
 Product foundBinary = SearchEngine.*binarySearch*(products, searchId);  
 System.*out*.println("\n Binary Search Result:");  
 if (foundBinary != null)  
 System.*out*.println(foundBinary);  
 else  
 System.*out*.println("Product not found.");  
 }  
}

public class Product {  
 private int productId;  
 private String productName;  
 private String category;  
  
 public Product(int productId, String productName, String category) {  
 this.productId = productId;  
 this.productName = productName;  
 this.category = category;  
 }  
  
 public int getProductId() {  
 return productId;  
 }  
  
 public String getProductName() {  
 return productName;  
 }  
  
 public String getCategory() {  
 return category;  
 }  
  
 @Override  
 public String toString() {  
 return "Product ID: " + productId +  
 ", Name: " + productName +  
 ", Category: " + category;  
 }  
}

import java.util.Arrays;  
import java.util.Comparator;  
  
public class SearchEngine {  
  
 public static Product linearSearch(Product[] products, int searchId) {  
 for (Product product : products) {  
 if (product.getProductId() == searchId) {  
 return product;  
 }  
 }  
 return null;  
 }  
  
 public static void sortByProductId(Product[] products) {  
 Arrays.*sort*(products, Comparator.*comparingInt*(Product::getProductId));  
 }  
  
 public static Product binarySearch(Product[] products, int searchId) {  
 int low = 0, high = products.length - 1;  
 while (low <= high) {  
 int mid = (low + high) / 2;  
 int midId = products[mid].getProductId();  
  
 if (midId == searchId)  
 return products[mid];  
 else if (midId < searchId)  
 low = mid + 1;  
 else  
 high = mid - 1;  
 }  
 return null;  
 }  
}



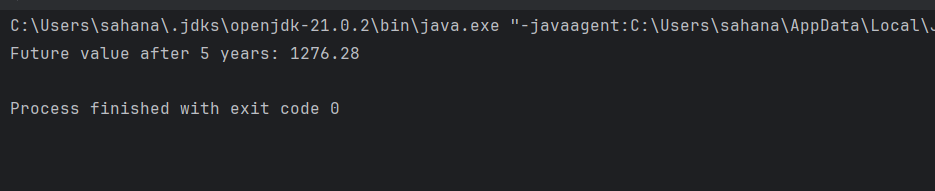
**Analysis**

**Time Complexity Comparison**

* **Linear Search**:
  + Time Complexity: **O(n)**
  + It checks each product one by one.
  + Slower when there are many products.
* **Binary Search**:
  + Time Complexity: **O(log n)**
  + It searches by dividing the sorted product list in half repeatedly.
  + Much faster than linear search.

2 . Financial Forecasting

public class FinancialForcasting {  
  
 // Recursive method to calculate future value  
 public static double calculateFutureValue(double presentValue, double growthRate, int years) {  
 if (years == 0) {  
 return presentValue;  
 }  
 return (1 + growthRate) \* *calculateFutureValue*(presentValue, growthRate, years - 1);  
 }  
  
 public static void main(String[] args) {  
 double presentValue = 1000; // Starting investment  
 double annualGrowthRate = 0.05; // 5% annual growth  
 int forecastYears = 5;  
  
 double futureValue = *calculateFutureValue*(presentValue, annualGrowthRate, forecastYears);  
  
 System.*out*.printf("Future value after %d years: %.2f\n", forecastYears, futureValue);  
 }  
}



**Analysis**

* Linear Search takes O(n) time.  
  It checks each item one by one. Slower for large data.
* Binary Search takes O(log n) time.  
  It works only on sorted data, but it's much faster.

Binary search is better for an e-commerce platform because it is faster and works well with large product lists.