**E-commerce Platform Search Function**

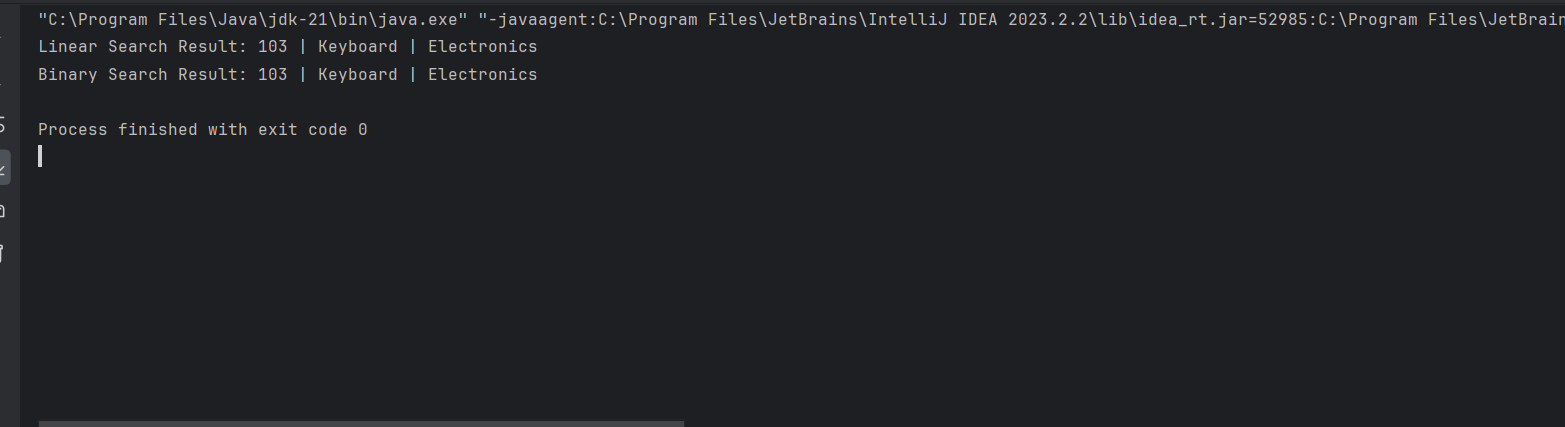
**Question 1:**

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

**Code:**

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;  
 }  
 @Override  
 public String toString() {  
 return productId + " | " + productName + " | " + category;  
 }  
}

import java.util.\*;  
public class Main {  
 public static Product binarySearch(Product[] products, String targetName) {  
 Arrays.*sort*(products, Comparator.*comparing*(p -> p.productName.toLowerCase()));  
  
 int left = 0, right = products.length - 1;  
 while (left <= right) {  
 int mid = (left + right) / 2;  
 int cmp = products[mid].productName.compareToIgnoreCase(targetName);  
 if (cmp == 0) return products[mid];  
 else if (cmp < 0) left = mid + 1;  
 else right = mid - 1;  
 }  
 return null;  
 }  
 public static Product linearSearch(Product[] products, String targetName) {  
 for (Product p : products) {  
 if (p.productName.equalsIgnoreCase(targetName)) {  
 return p;  
 }  
 }  
 return null;  
 }  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(101, "Mouse", "Electronics"),  
 new Product(102, "Shirt", "Clothing"),  
 new Product(103, "Keyboard", "Electronics"),  
 new Product(104, "Shoes", "Footwear"),  
 new Product(105, "Monitor", "Electronics")  
 };  
 Product found1 = *linearSearch*(products, "Keyboard");  
 System.*out*.println("Linear Search Result: " + (found1 != null ? found1 : "Not Found"));  
 Product found2 = *binarySearch*(products, "Keyboard");  
 System.*out*.println("Binary Search Result: " + (found2 != null ? found2 : "Not Found"));  
 }  
}



**Linear Search:**

* Time complexity: O(n)
* Space complexity: O(1)-No extra memory used.

**Binary Search:**

* **Time Complexity:**
  + Worst case: O(n)
  + Best case: O(1)
  + Average case: O(n)
* **Space Complexity:** O(1)- No extra memory used.

**Binary Search is efficient as it reduces Search Time from O(n) to O(log n).**

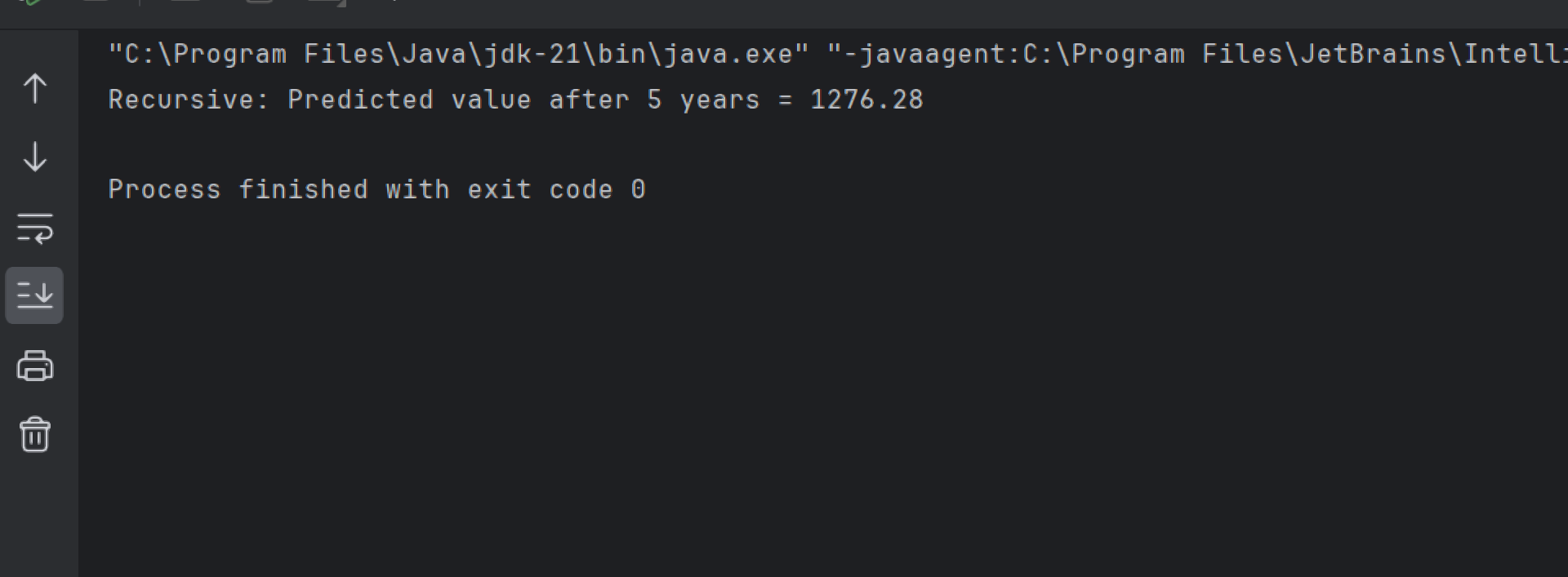
**Financial Forecasting**

**Question 2:**

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

**Code:**

public class Forecasting {  
 public static double Recursive(double presentValue, double growthRate, int years) {  
 if (years == 0) return presentValue;  
 return (1 + growthRate) \* *Recursive*(presentValue, growthRate, years - 1);  
 }  
 public static void main(String[] args) {  
 double present = 1000.0;  
 double rate = 0.05;  
 int years = 5;  
  
 double resultRecursive = *Recursive*(present, rate, years);  
  
 System.*out*.printf("Recursive: Predicted value after %d years = %.2f\n", years, resultRecursive);  
  
 }  
}



**Time Complexity: O(n)**

**Space Complexity: O(n)**

**To optimize the code we can use Iterative method**