WEEK-1 Algorithms\_Data\_Structures

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

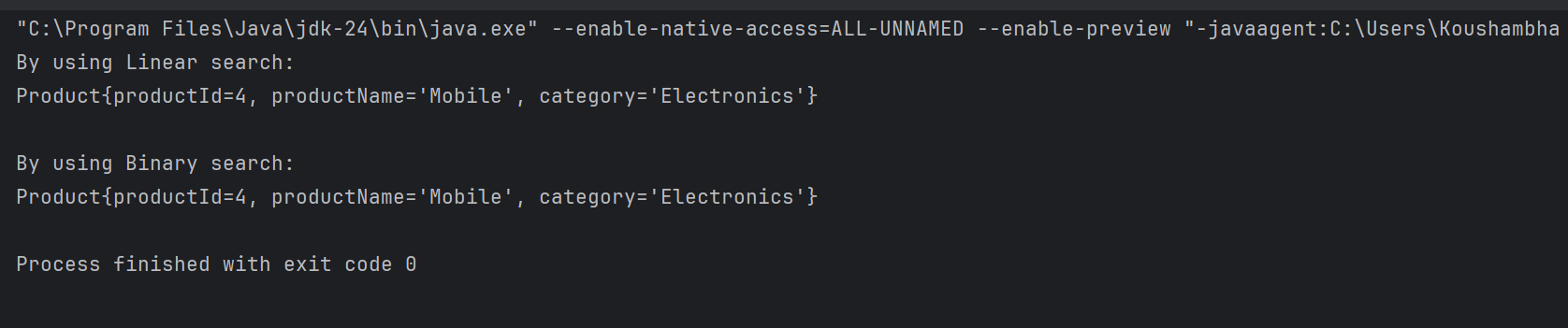
**CODE:**

**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 void setProductId(int productId) {  
 this.productId = productId;  
 }  
  
 public String getProductName() {  
 return productName;  
 }  
  
 public void setProductName(String productName) {  
 this.productName = productName;  
 }  
  
 public String getCategory() {  
 return category;  
 }  
  
 public void setCategory(String category) {  
 this.category = category;  
 }  
  
 @Override  
 public String toString() {  
 return "Product{" +  
 "productId=" + productId +  
 ", productName='" + productName + '\'' +  
 ", category='" + category + '\'' +  
 '}';  
 }  
}**

**import java.util.Arrays;  
import java.util.Comparator;  
  
public class SearchUtil {  
  
 public static Product linearSearch(Product[] products, String name) {  
 for (Product product : products) {  
 if (product.getProductName().equalsIgnoreCase(name)) {  
 return product;  
 }  
 }  
 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 comparison = products[mid].getProductName().compareToIgnoreCase(name);  
 if (comparison == 0) return products[mid];  
 else if (comparison < 0) low = mid + 1;  
 else high = mid - 1;  
 }  
 return null;  
 }  
  
 public static void sortByName(Product[] products) {  
 Arrays.*sort*(products, Comparator.*comparing*(Product::getProductName));  
 }  
}**

**public class Main {  
 public static void main(String[] args) {  
 Product[] products = new Product[]{  
 new Product(1, "Laptop", "Electronics"),  
 new Product(2, "Shoes", "Fashion"),  
 new Product(3, "Book", "Education"),  
 new Product(4, "Mobile", "Electronics"),  
 new Product(5, "Watch", "Accessories")  
 };  
  
 String targetName = "Mobile";  
  
 System.*out*.println("By using Linear search:");  
 Product foundLinear = SearchUtil.*linearSearch*(products, targetName);  
 System.*out*.println(foundLinear != null ? foundLinear : "Could Not Find Product");  
  
  
 SearchUtil.*sortByName*(products);  
 System.*out*.println();  
 System.*out*.println("By using Binary search:");  
 Product foundBinary = SearchUtil.*binarySearch*(products, targetName);  
 System.*out*.println(foundBinary != null ? foundBinary : "Could Not Find Product");  
 }  
}**

OUTPUT:

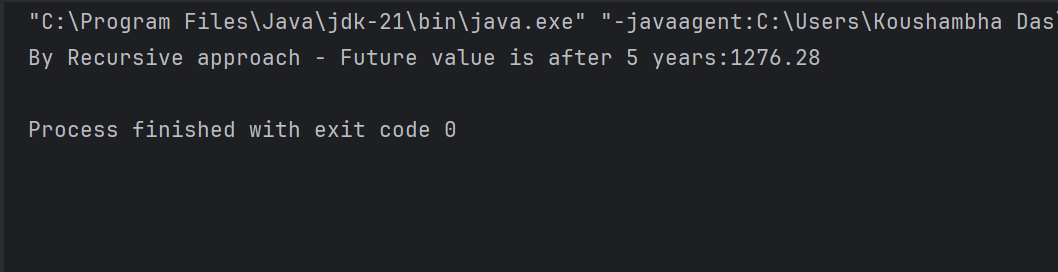


**Exercise 7: Financial Forecasting**

**CODE:**

**public class FinancialForecastRecursive {  
  
 public static double calculateFutureValue(double presentValue, double rate, int periods) {  
 if (periods == 0) {  
 return presentValue;  
 }  
  
 return (1 + rate) \* *calculateFutureValue*(presentValue, rate, periods - 1);  
 }  
  
 public static void main(String[] args) {  
 double presentValue = 1000.0;  
 double rate = 0.05;  
 int periods = 5;  
  
 double futureValue = *calculateFutureValue*(presentValue, rate, periods);  
 System.*out*.println("By Recursive approach - Future value is after " + periods + " years:" + String.*format*("%.2f", futureValue));  
 }  
}**

OUTPUT (Recursive Approach):



**CODE:**

**public class FinancialForecastOptimized {  
 public static double calculateFutureValue(double presentValue, double rate, int periods) {  
 double result = presentValue;  
 for (int i = 0; i < periods; i++) {  
 result \*= (1 + rate);  
 }  
 return result;  
 }  
  
 public static void main(String[] args) {  
 double presentValue = 1000.0;  
 double rate = 0.05;  
 int periods = 5;  
  
 double futureValue = *calculateFutureValue*(presentValue, rate, periods);  
 System.*out*.println("By Iterative approach - The Future Value after " + periods + " years: " + String.*format*("%.2f", futureValue));  
 }  
}**

OUTPUT (Iterative/Optimized Approach):

