# **EXERCISE 2: E-COMMERCE PLATFORM SEARCH FUNCTION**

**Main.java**

**package** ecom;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Product[] products = {

**new** Product(101, "Laptop", "Electronics"),

**new** Product(102, "Shoes", "Fashion"),

**new** Product(103, "Camera", "Electronics"),

**new** Product(104, "Watch", "Accessories"),

**new** Product(105, "Mobile", "Electronics")

};

System.***out***.println("🔍 Linear Search for 'Camera':");

Product foundLinear = SearchEngine.*linearSearch*(products, "Camera");

System.***out***.println(foundLinear != **null** ? foundLinear : "Product not found");

SearchEngine.*sortProductsByName*(products);

System.***out***.println("🔍 Binary Search for 'Camera':");

Product foundBinary = SearchEngine.*binarySearch*(products, "Camera");

System.***out***.println(foundBinary != **null** ? foundBinary : "Product not found");

}

}

**Product.java**  
  
**package** ecom;

**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 + "]";

}

}

**SearchEngine.java**

**package** ecom;

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** SearchEngine {

**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** left = 0;

**int** right = products.length - 1;

**while** (left <= right) {

**int** mid = (left + right) / 2;

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

**if** (cmp == 0) **return** products[mid];

**else** **if** (cmp < 0) left = mid + 1;

**else** right = mid - 1;

}

**return** **null**;

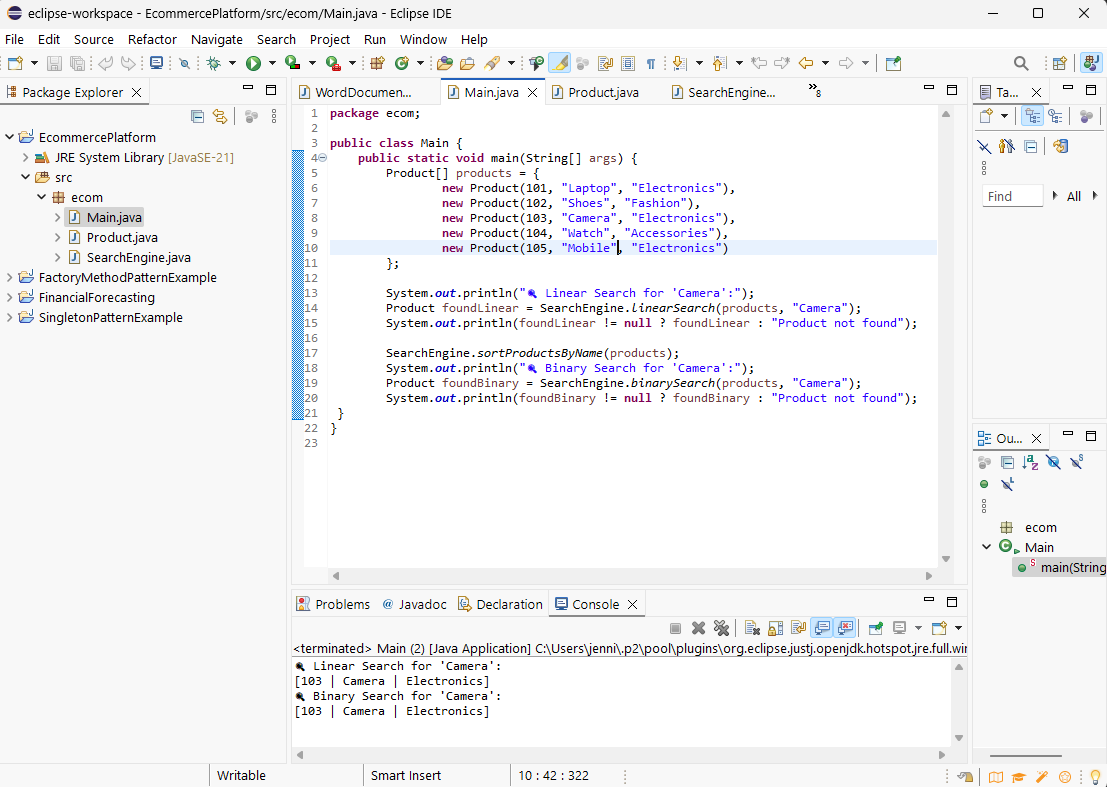
}

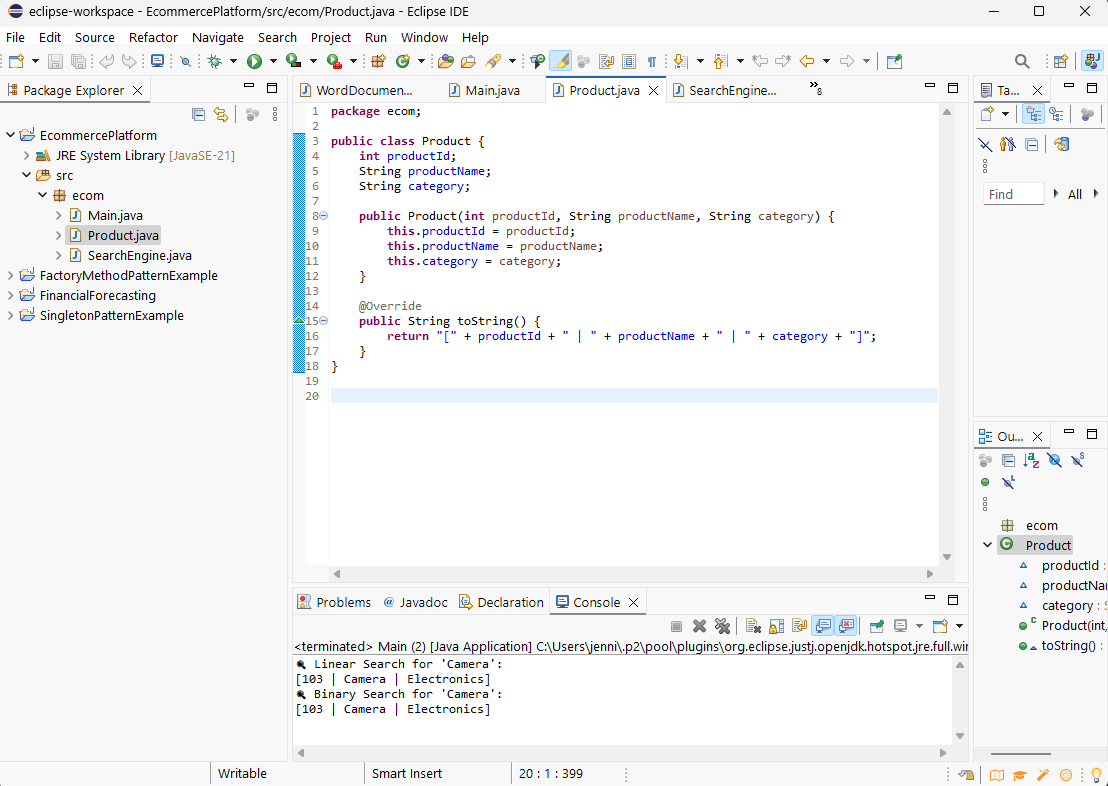
**public** **static** **void** sortProductsByName(Product[] products) {

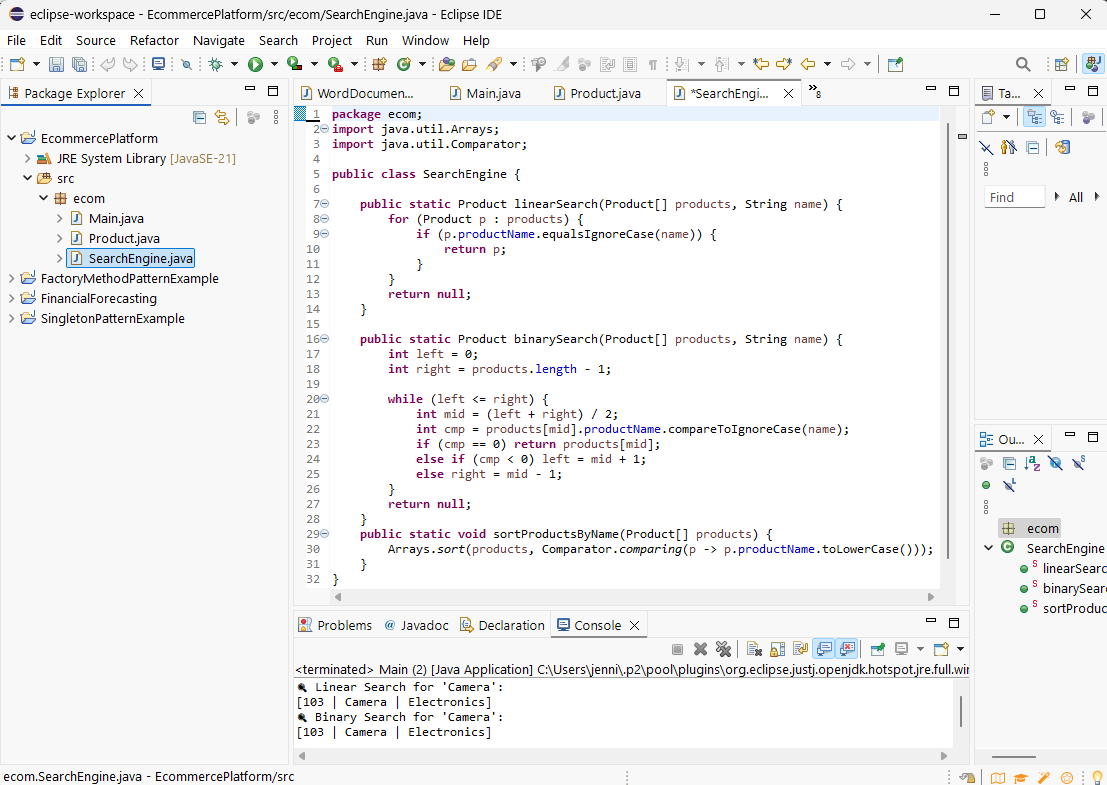
Arrays.*sort*(products, Comparator.*comparing*(p -> p.productName.toLowerCase()));

}

}







# **EXERCISE 7: FINANCIAL FORECASTING**

**Main.java**

**package** fincForecast;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**double** currentValue = 10000;

**double** growthRate = 0.05;

**int** years = 5;

**double** predictedValue = Forecasting.*futureValueRecursive*(currentValue, growthRate, years);

System.***out***.printf("Future Value after %d years: ₹%.2f%n", years, predictedValue);

}

}

**Forecasting.java**

**package** fincForecast;

**public** **class** Forecasting {

**public** **static** **double** futureValueRecursive(**double** currentValue, **double** growthRate, **int** years) {

**if** (years == 0) {

**return** currentValue;

}

**return** *futureValueRecursive*(currentValue, growthRate, years - 1) \* (1 + growthRate);

}

**public** **static** **double** futureValueTail(**double** currentValue, **double** growthRate, **int** years) {

**return** *futureValueHelper*(currentValue, growthRate, years);

}

**private** **static** **double** futureValueHelper(**double** value, **double** growthRate, **int** years) {

**if** (years == 0) {

**return** value;

}

**return** *futureValueHelper*(value \* (1 + growthRate), growthRate, years - 1);

}

}

