Exercise 2: E-commerce Platform Search Function

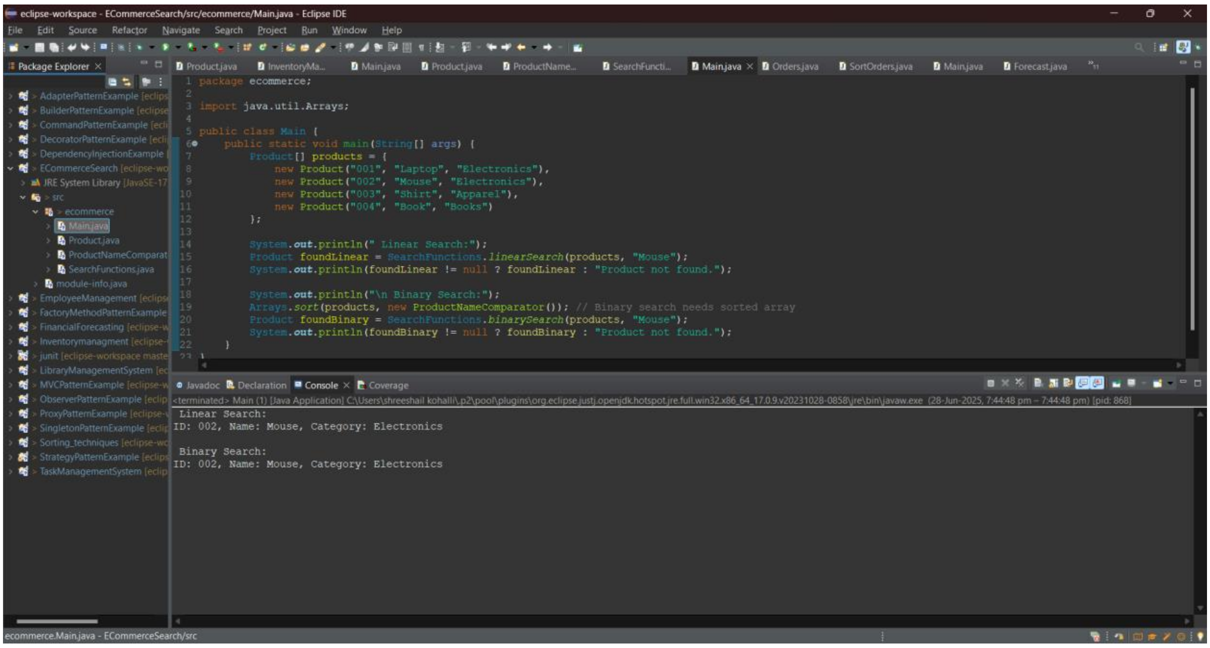
```
//Product.java


package ecommerce;

public class Product {
    String productId;
    String productName;
    String category;

    public Product(String productId, String productName, String category) {
        this.productId = productId;
        this.productName = productName;
        this.category = category;
    }

    @Override
    public String toString() {
        return "ID: " + productId + ", Name: " + productName + ", Category:
" + category;
    }
}
```

```
ProductNameComparator.java

package ecommerce;

import java.util.Comparator;

public class ProductNameComparator implements Comparator<Product> {
    @Override
    public int compare(Product a, Product b) {
        return a.productName.compareToIgnoreCase(b.productName);
    }
}
```

```
SearchFunctions.java


package ecommerce;

public class SearchFunctions {
    // Linear Search
    public static Product linearSearch(Product[] products, String name) {
        for (Product product : products) {
            if (product.productName.equalsIgnoreCase(name)) {
                return product;
            }
        }
        return null;
    }

    // Binary Search
    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;
    }
}
```

```
Main.java

package ecommerce;

import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        Product[] products = {
            new Product("001", "Laptop", "Electronics"),
            new Product("002", "Mouse", "Electronics"),
            new Product("003", "Shirt", "Apparel"),
            new Product("004", "Book", "Books")
        };

        System.out.println(" Linear Search:");
        Product foundLinear = SearchFunctions.linearSearch(products,
"Mouse");
        System.out.println(foundLinear != null ? foundLinear : "Product not
found.");

        System.out.println("\n Binary Search:");
        Arrays.sort(products, new ProductNameComparator()); // Binary
search needs sorted array
        Product foundBinary = SearchFunctions.binarySearch(products,
"Mouse");
        System.out.println(foundBinary != null ? foundBinary : "Product not
found.");
    }
}
```

Output:



```
package ecommerce;

import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        Product[] products = {
            new Product("001", "Laptop", "Electronics"),
            new Product("002", "Mouse", "Electronics"),
            new Product("003", "Shirt", "Apparel"),
            new Product("004", "Book", "Books")
        };

        System.out.println(" Linear Search:");
        Product foundLinear = SearchFunctions.linearSearch(products, "Mouse");
        System.out.println(foundLinear != null ? foundLinear : "Product not found.");

        System.out.println("\n Binary Search:");
        Arrays.sort(products, new ProductNameComparator()); // Binary search needs sorted array
        Product foundBinary = SearchFunctions.binarySearch(products, "Mouse");
        System.out.println(foundBinary != null ? foundBinary : "Product not found.");
    }
}
```

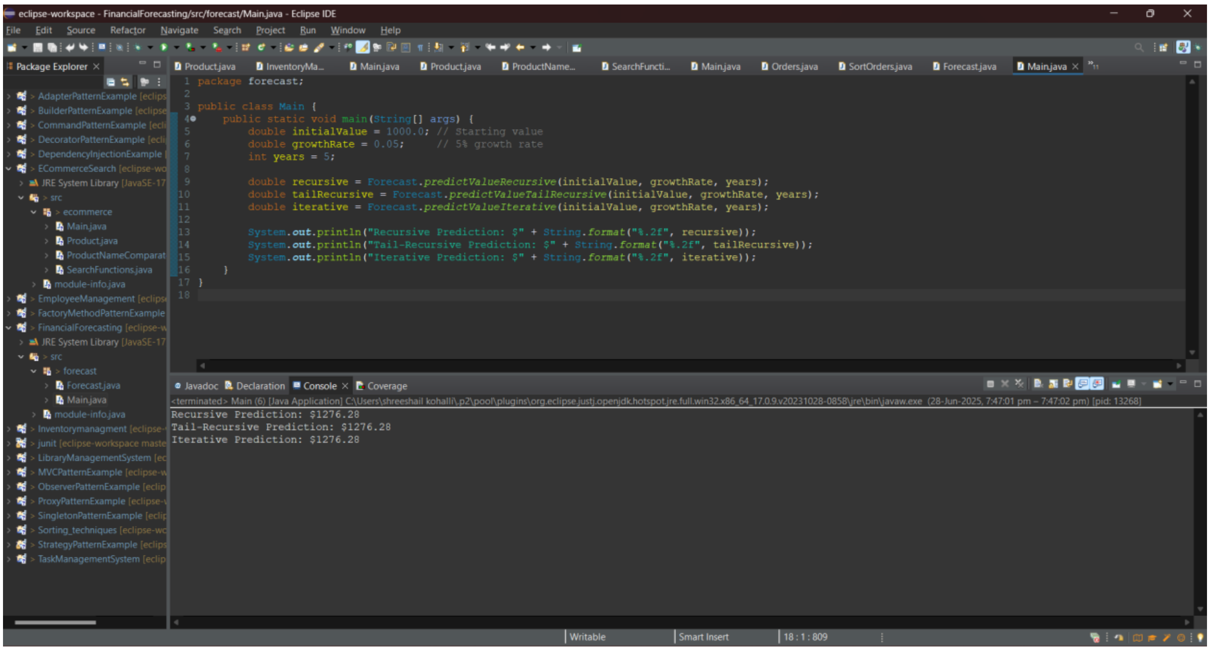
```
<terminated> Main (1) [Java Application] C:\Users\shreeshail kohalli\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe  (28-Jun-2025, 7:44:48 pm – 7:44:48 pm) [pid: 868]
 Linear Search:
ID: 002, Name: Mouse, Category: Electronics

 Binary Search:
ID: 002, Name: Mouse, Category: Electronics
```

Exercise 7: Financial Forecasting

```
package forecast;

public class Forecast {

    // Recursive method to calculate future value
    public static double predictValueRecursive(double initialValue, double
growthRate, int years) {
        if (years == 0) {
            return initialValue;
        }
        return predictValueRecursive(initialValue, growthRate, years - 1) *
(1 + growthRate);
    }

    // Optimized: Tail-recursive like method (uses accumulator)
    public static double predictValueTailRecursive(double initialValue,
double growthRate, int years) {
        return predictHelper(initialValue, growthRate, years);
    }

    private static double predictHelper(double value, double growthRate,
int years) {
        if (years == 0) return value;
        return predictHelper(value * (1 + growthRate), growthRate, years -
1);
    }

    // Iterative method (non-recursive alternative)
    public static double predictValueIterative(double initialValue, double
growthRate, int years) {
        double value = initialValue;
        for (int i = 0; i < years; i++) {
            value *= (1 + growthRate);
        }
        return value;
    }
}
```

```
Main.java

package forecast;

public class Main {
    public static void main(String[] args) {
        double initialValue = 1000.0; // Starting value
        double growthRate = 0.05;     // 5% growth rate
        int years = 5;

        double recursive = Forecast.predictValueRecursive(initialValue,
growthRate, years);
        double tailRecursive =
Forecast.predictValueTailRecursive(initialValue, growthRate, years);
        double iterative = Forecast.predictValueIterative(initialValue,
growthRate, years);

        System.out.println("Recursive Prediction: $" +
String.format("%.2f", recursive));
        System.out.println("Tail-Recursive Prediction: $" +
String.format("%.2f", tailRecursive));
        System.out.println("Iterative Prediction: $" +
String.format("%.2f", iterative));
    }
}
```

Output:



```java
package forecast;

public class Main {
    public static void main(String[] args) {
        double initialValue = 1000.0; // Starting value
        double growthRate = 0.05;     // 5% growth rate
        int years = 5;

        double recursive = Forecast.predictValueRecursive(initialValue, growthRate, years);
        double tailRecursive = Forecast.predictValueTailRecursive(initialValue, growthRate, years);
        double iterative = Forecast.predictValueIterative(initialValue, growthRate, years);

        System.out.println("Recursive Prediction: $" + String.format("%.2f", recursive));
        System.out.println("Tail-Recursive Prediction: $" + String.format("%.2f", tailRecursive));
        System.out.println("Iterative Prediction: $" + String.format("%.2f", iterative));
    }
}

```

```
<terminated> Main (6) [Java Application] C:\Users\shreeshail kohalli\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe  (28-Jun-2025, 7:47:01 pm – 7:47:02 pm) [pid: 13268]
Recursive Prediction: $1276.28
Tail-Recursive Prediction: $1276.28
Iterative Prediction: $1276.28
```