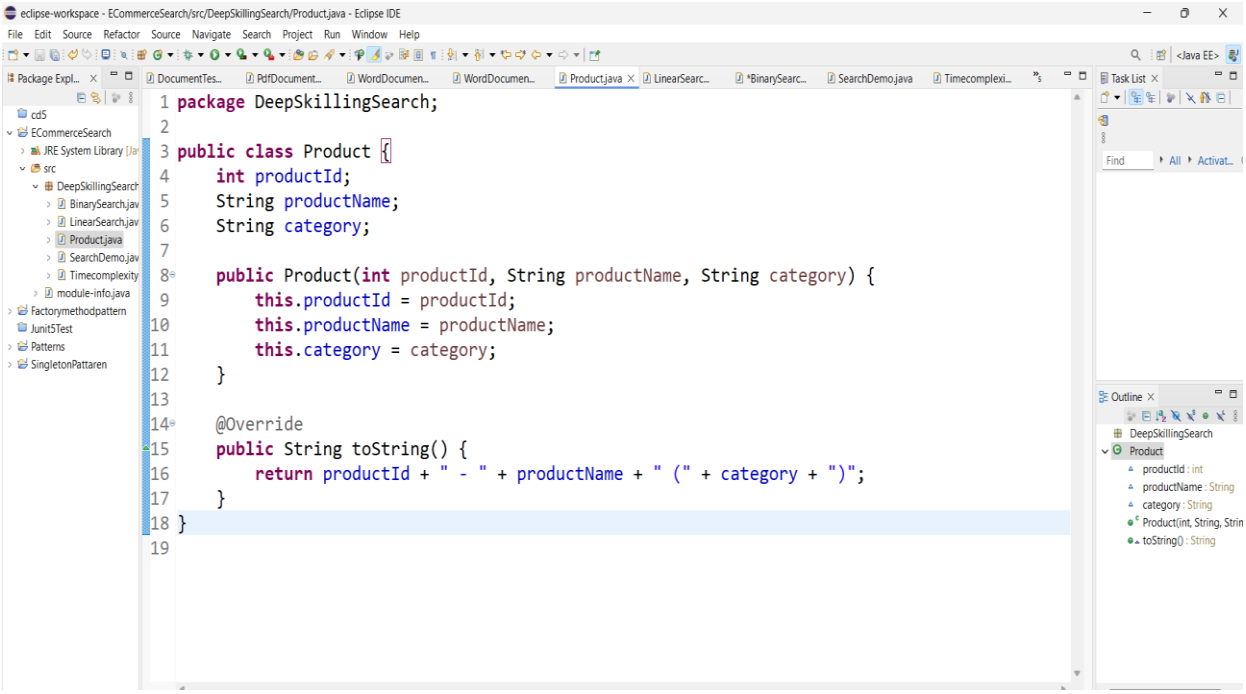
E-commerce Platform Search Function

1) Product.java : code

```
package DeepSkillingSearch;
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 + ")";
    }  }
```

Pic From Eclipse IDE: Java



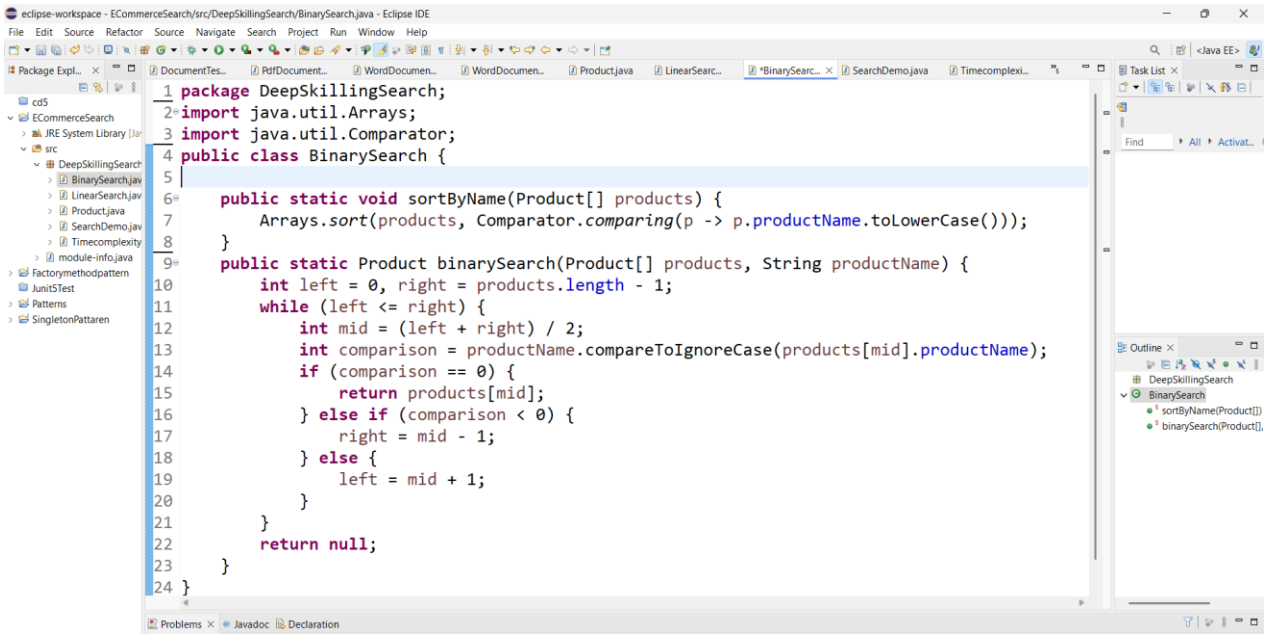
2) BinarSearch.java : Code

```
package DeepSkillingSearch;

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

    public static void sortByName(Product[] products) {

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

    }
```

```
    public static Product binarySearch(Product[]
products, String productName) {
        int left = 0, right = products.length - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            int comparison =
productName.compareToIgnoreCase(products[mid].pr
oductName);
            if (comparison == 0) {
                return products[mid];
            } else if (comparison < 0) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        return null;
    }
```

}

Pic From Eclipse IDE:

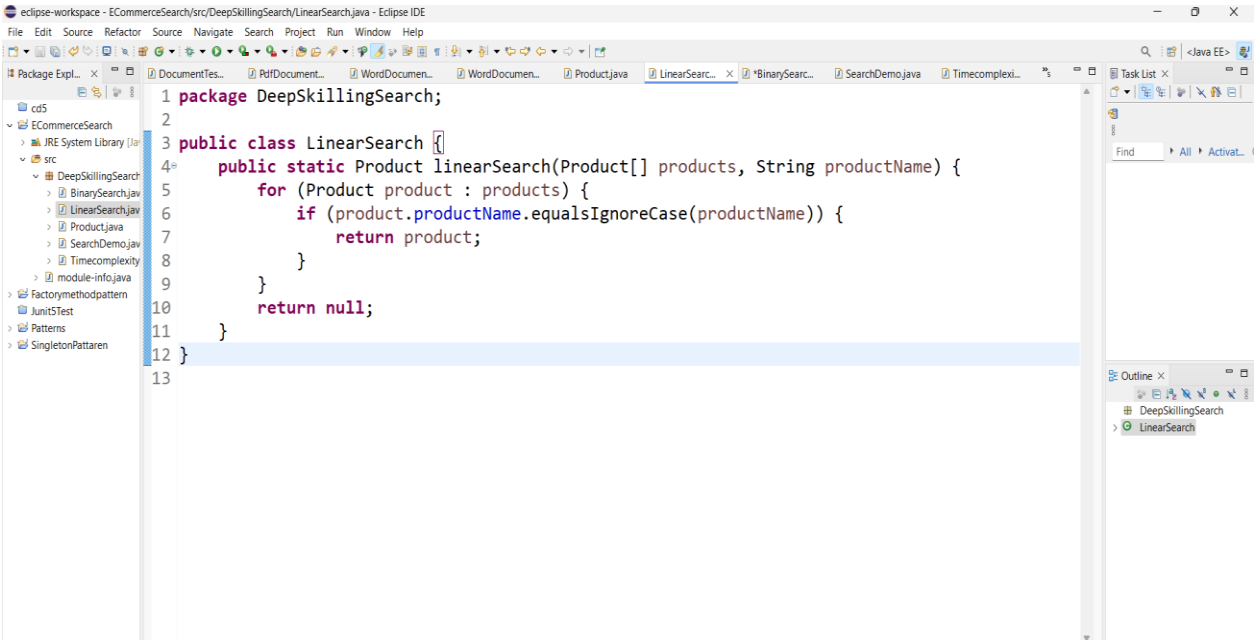


3)LinearSearch.java : Code

```
package DeepSkillingSearch;

public class LinearSearch {

    public static Product linearSearch(Product[]
products, String productName) {

        for (Product product : products) {

            if
(product.productName.equalsIgnoreCase(productNam
e)) {
```

```
            return product;
            }
        }
        return null;
    }
}
```

Pic From Eclipse IDE :

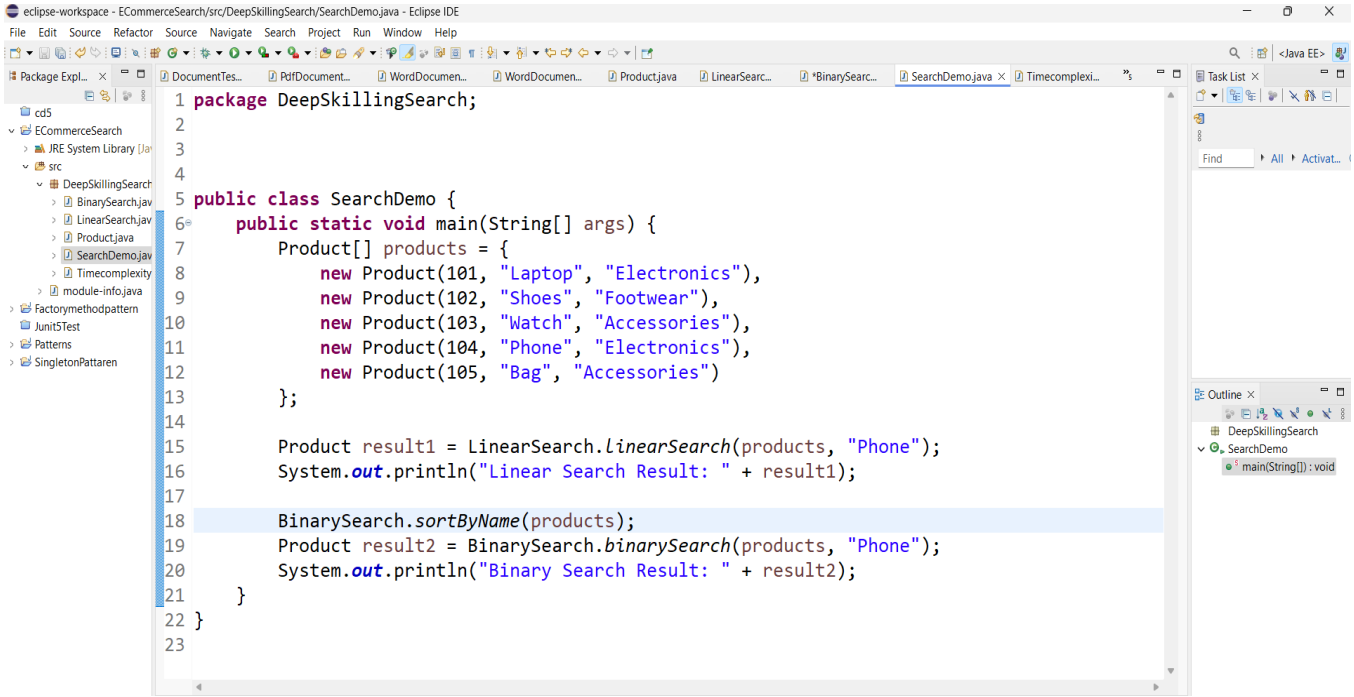


4)MainClass-SearchDemo.java : Code

```
package DeepSkillingSearch;
public class SearchDemo {
    public static void main(String[] args) {
        Product[] products = {
            new Product(101, "Laptop", "Electronics"),
            new Product(102, "Shoes", "Footwear"),
            new Product(103, "Watch", "Accessories"),
            new Product(104, "Phone", "Electronics"),
            new Product(105, "Bag", "Accessories")
        };
        Product result1 =
LinearSearch.linearSearch(products, "Phone");
        System.out.println("Linear Search Result: " +
result1);
        BinarySearch.sortByName(products);
        Product result2 =
BinarySearch.binarySearch(products, "Phone");
```

```
        System.out.println("Binary Search Result: " + result2);
    }
}
```

Pic From Eclipse IDE:

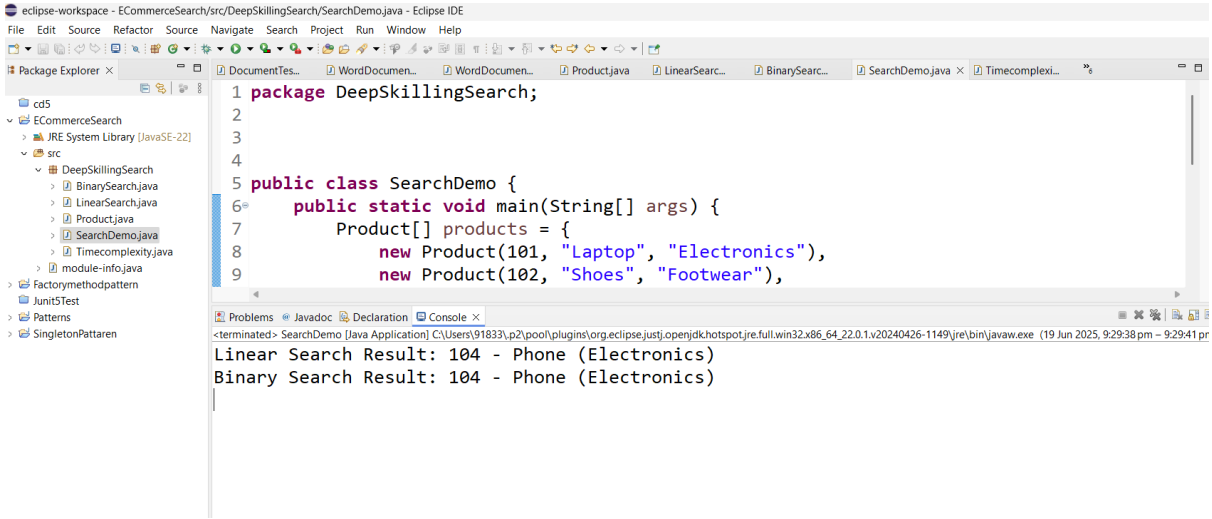


OUTPUT:

Linear Search Result: 104 - Phone (Electronics)

Binary Search Result: 104 - Phone (Electronics)

Pic From Eclipse IDE:



In this we can see the item(phone) is searched through the both the LineraSearch and the Binarysearch.
But to Know the which search is more efficient and takes less time we can know by the timeComplexity.
TimeComplexity:

LinearSearch:

Bestcase: O(1)
AverageCase : O(n)

BinarySearch:

BestCase: O(1)

AverageCase: O(log n)

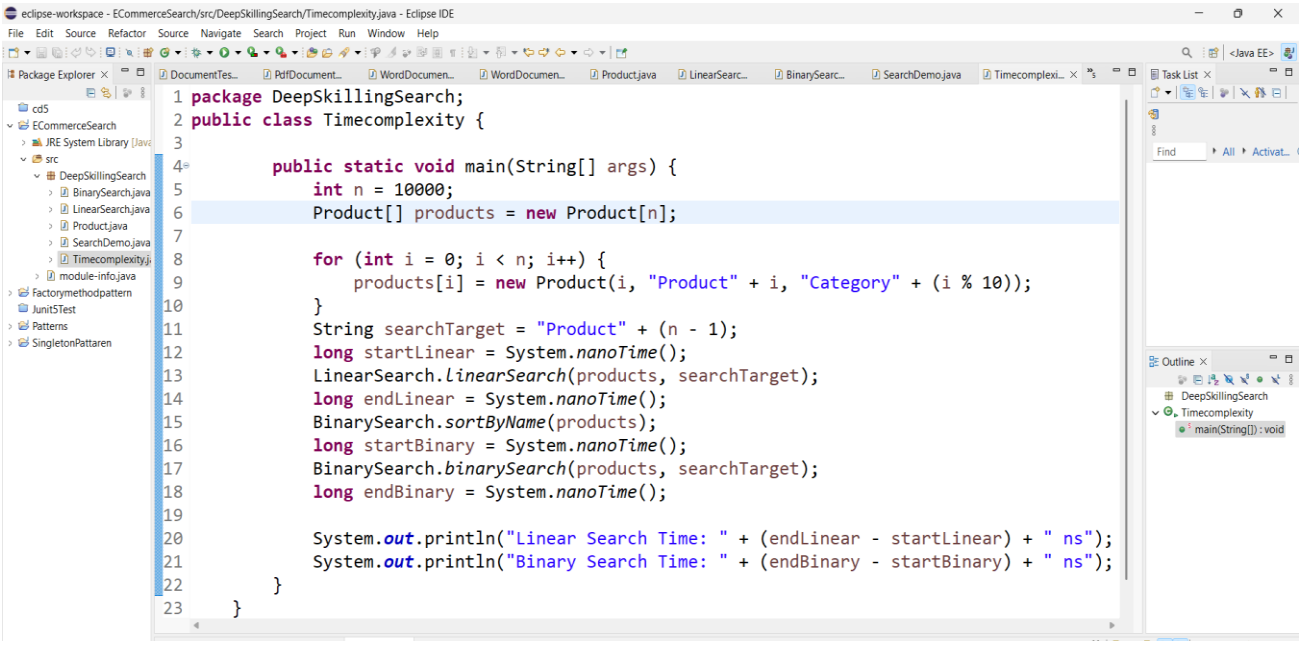
To directly see this I wrote another class TimeComplexity By which we can see the Time taken By them For searching.

6)TimeComplexity.Java : Code

```
package DeepSkillingSearch;

public class Timecomplexity {

        public static void main(String[] args) {

            int n = 10000;

            Product[] products = new Product[n];

            for (int i = 0; i < n; i++) {

                products[i] = new Product(i, "Product" + i,
"Category" + (i % 10));

            }

            String searchTarget = "Product" + (n - 1);

            long startLinear = System.nanoTime();

            LinearSearch.linearSearch(products,
searchTarget);

            long endLinear = System.nanoTime();

            BinarySearch.sortByName(products);
```

```
        long startBinary = System.nanoTime();
        BinarySearch.binarySearch(products, searchTarget);
        long endBinary = System.nanoTime();
        System.out.println("Linear Search Time: " + (endLinear - startLinear) + " ns");
        System.out.println("Binary Search Time: " + (endBinary - startBinary) + " ns");
    }
}
```

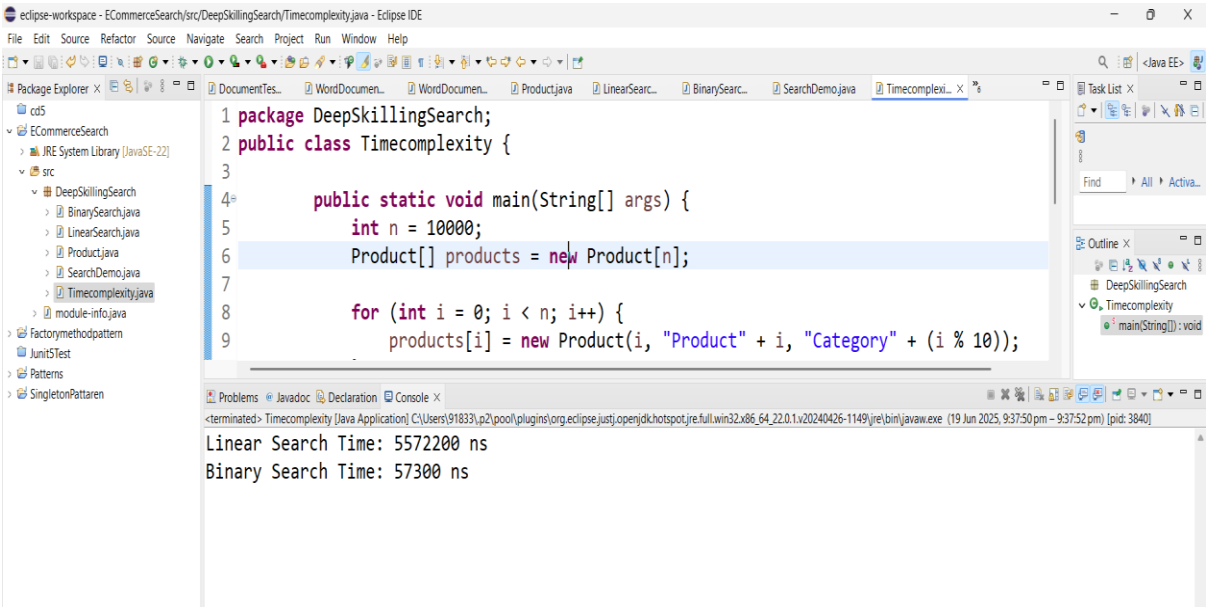
Pic From Eclipse IDE:



OUTPUT:

Linear Search Time: 5572200 ns

Binary Search Time: 57300 ns



By this we can tell that BinarySerach takes less time and more Efficient.