

E-commerce Platform Search Function

1) Product.java : code

```
package DeepSkillSearch;

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;

    }

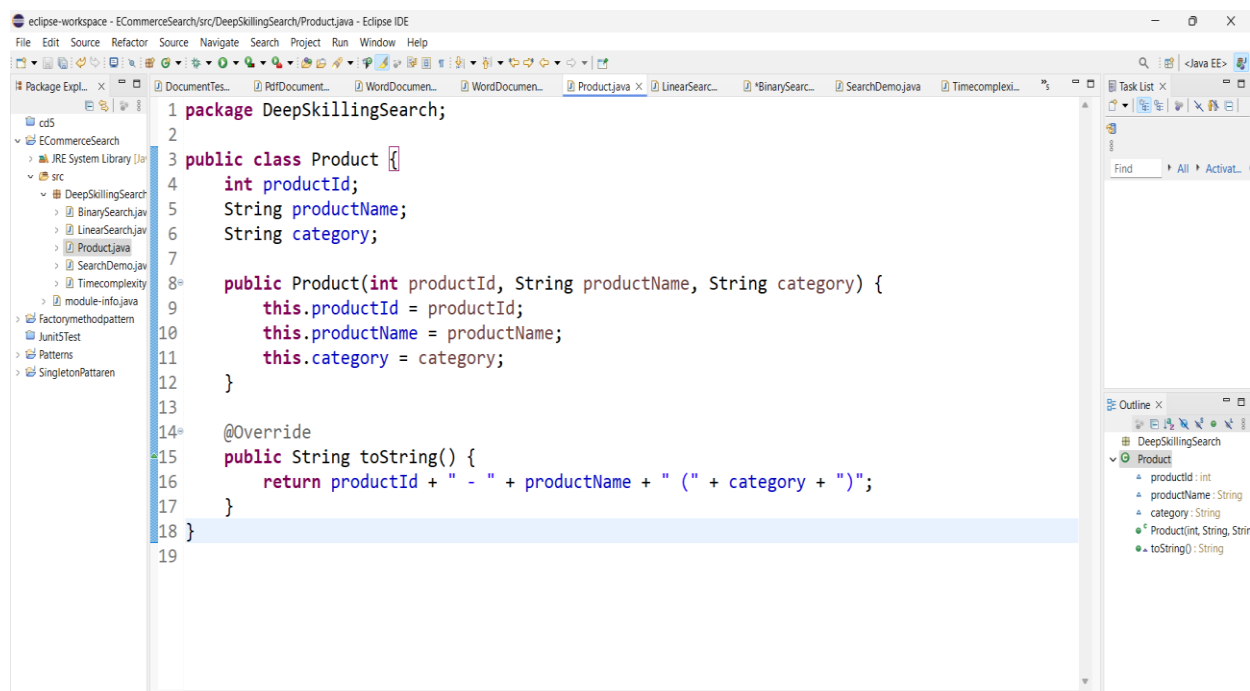
    @Override

    public String toString() {

        return productId + " - " + productName + " (" +
category + ")";

    } }
```

Pic From Eclipse IDE: Java



2) BinarSearch.java : Code

```
-----  
  
package DeepSkillSearch;  
  
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()));  
  
    }  
  
}
```

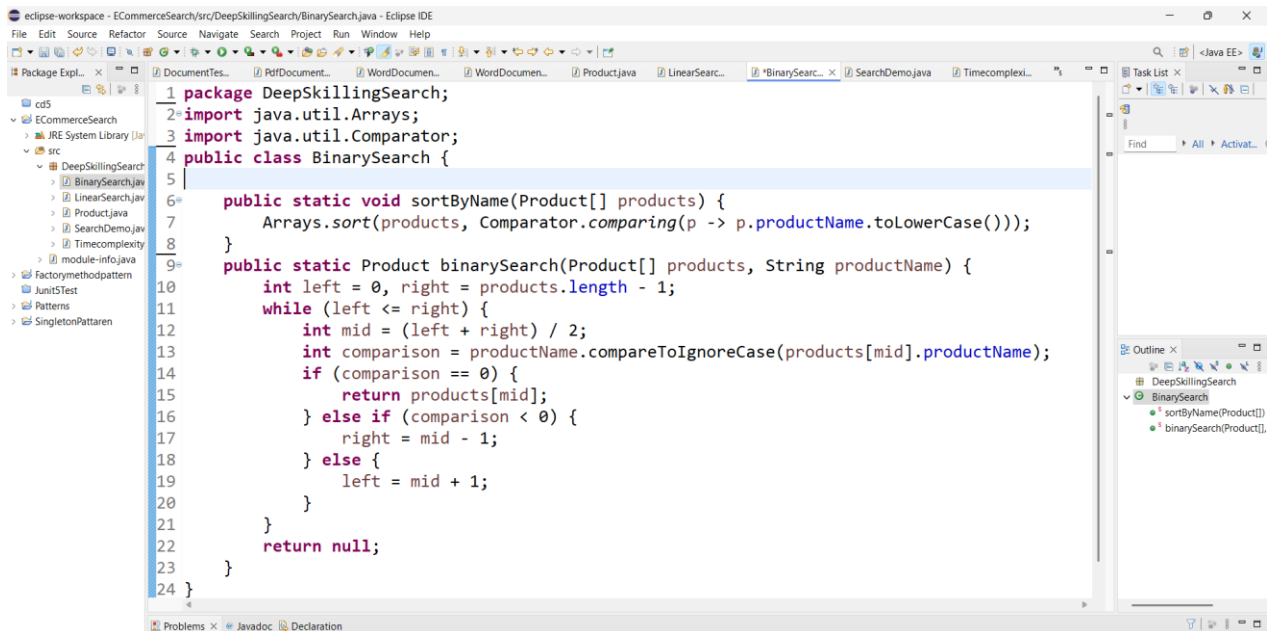
```
public static Product binarySearch(Product[]
products, String productName) {
    int left = 0, right = products.length - 1;
    while (left <= right) {
        int mid = (left + right) / 2;

        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 DeepSkilllingSearch;
```

```
public class LinearSearch {
```

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

```
        for (Product product : products) {
```

```
            if
```

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

```
        return product;
    }

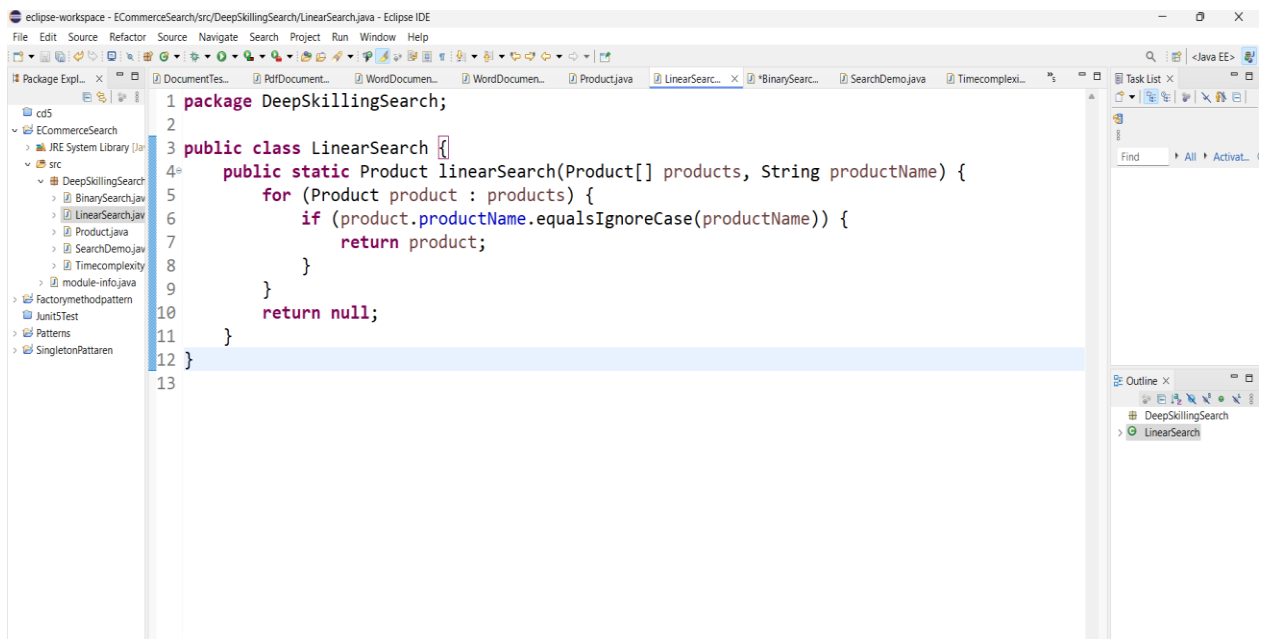
}

return null;

}

}
```

Pic From Eclipse IDE :

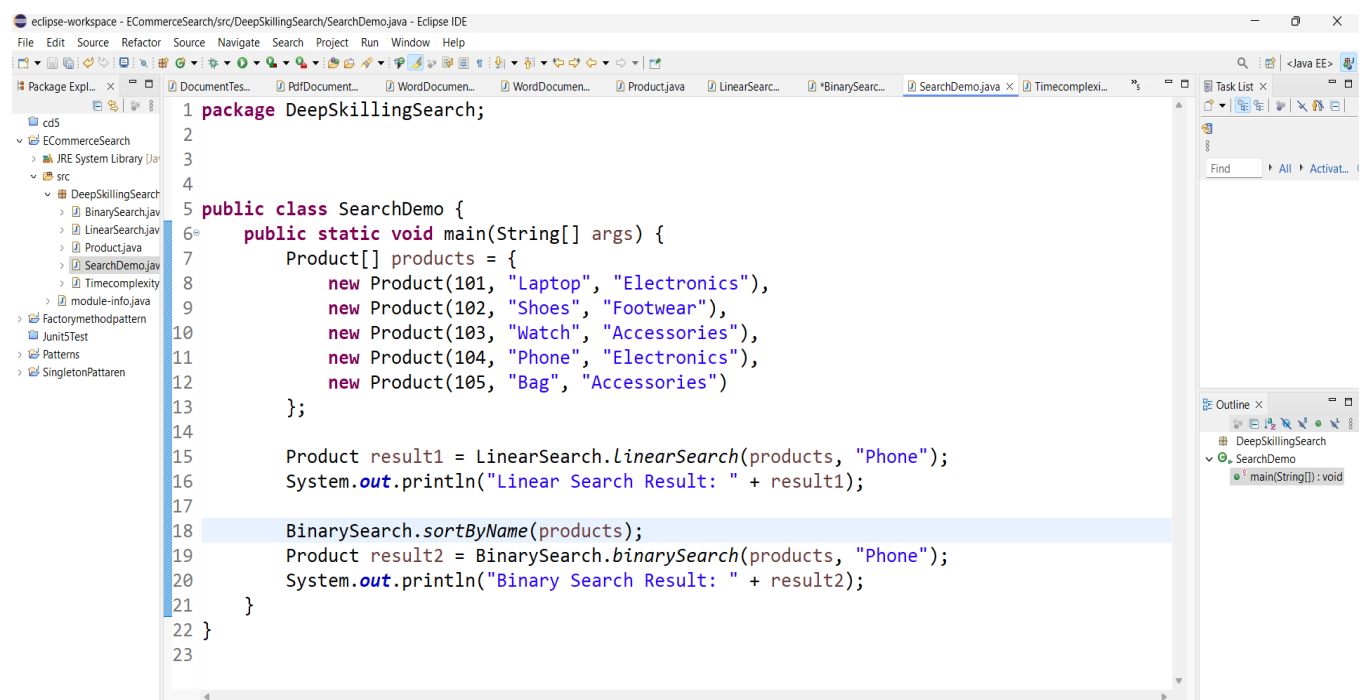


4)MainClass-SearchDemo.java : Code

```
-----  
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:



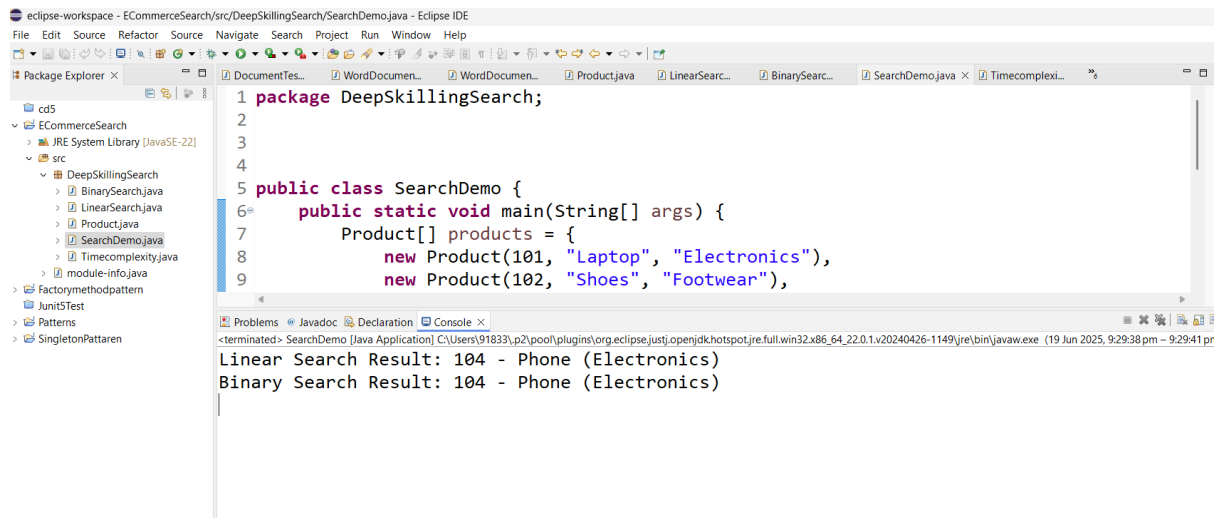
```
1 package DeepSkilllingSearch;  
2  
3  
4  
5 public class SearchDemo {  
6     public static void main(String[] args) {  
7         Product[] products = {  
8             new Product(101, "Laptop", "Electronics"),  
9             new Product(102, "Shoes", "Footwear"),  
10            new Product(103, "Watch", "Accessories"),  
11            new Product(104, "Phone", "Electronics"),  
12            new Product(105, "Bag", "Accessories")  
13        };  
14  
15        Product result1 = LinearSearch.linearSearch(products, "Phone");  
16        System.out.println("Linear Search Result: " + result1);  
17  
18        BinarySearch.sortByName(products);  
19        Product result2 = BinarySearch.binarySearch(products, "Phone");  
20        System.out.println("Binary Search Result: " + result2);  
21    }  
22 }  
23
```

OUTPUT:

Linear Search Result: 104 - Phone (Electronics)

Binary Search Result: 104 - Phone (Electronics)

Pic From Eclipse IDE:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'cd5' with a package 'ECommerceSearch' containing a 'src' folder. Inside 'src', there is a package 'DeepSkillSearch' with files 'BinarySearch.java', 'LinearSearch.java', 'Product.java', 'SearchDemo.java', and 'TimeComplexity.java'. The main editor window shows the code for 'SearchDemo.java'. The code defines a package 'DeepSkillSearch' and a public class 'SearchDemo' with a 'main' method. The 'main' method creates an array of 'Product' objects: 'new Product(101, "Laptop", "Electronics")' and 'new Product(102, "Shoes", "Footwear")'. The console at the bottom shows the output of the program: 'Linear Search Result: 104 - Phone (Electronics)' and 'Binary Search Result: 104 - Phone (Electronics)'.

```
1 package DeepSkillSearch;
2
3
4
5 public class SearchDemo {
6     public static void main(String[] args) {
7         Product[] products = {
8             new Product(101, "Laptop", "Electronics"),
9             new Product(102, "Shoes", "Footwear"),
10        }
```

Linear Search Result: 104 - Phone (Electronics)
Binary Search Result: 104 - Phone (Electronics)

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 DeepSkillSearch;  
  
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");

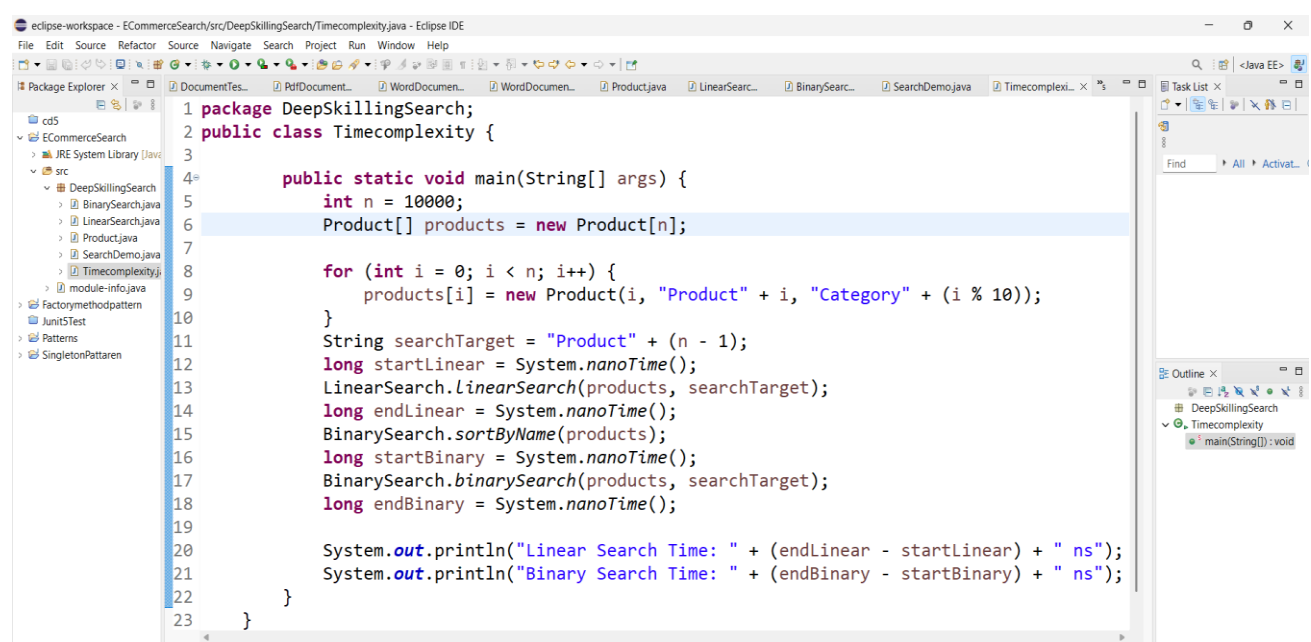
        System.out.println("Binary Search Time: " +
(endBinary - startBinary) + " ns");

    }

}

```

Pic From Eclipse IDE:



```

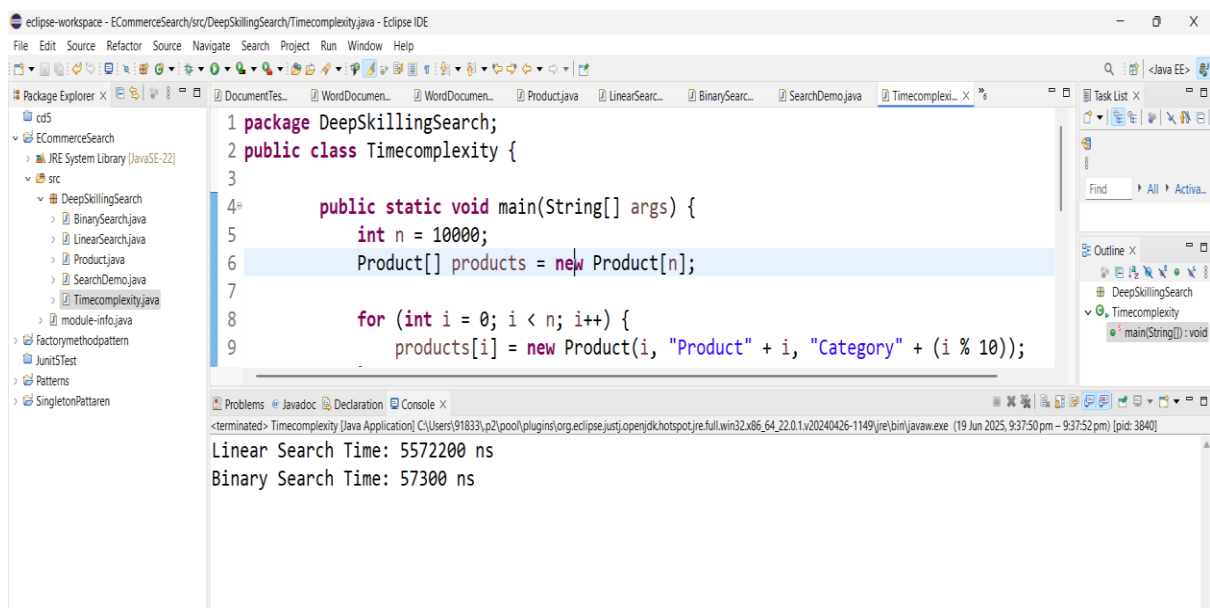
1 package DeepSkillSearch;
2 public class Timecomplexity {
3
4     public static void main(String[] args) {
5         int n = 10000;
6         Product[] products = new Product[n];
7
8         for (int i = 0; i < n; i++) {
9             products[i] = new Product(i, "Product" + i, "Category" + (i % 10));
10        }
11        String searchTarget = "Product" + (n - 1);
12        long startLinear = System.nanoTime();
13        LinearSearch.linearSearch(products, searchTarget);
14        long endLinear = System.nanoTime();
15        BinarySearch.sortByName(products);
16        long startBinary = System.nanoTime();
17        BinarySearch.binarySearch(products, searchTarget);
18        long endBinary = System.nanoTime();
19
20        System.out.println("Linear Search Time: " + (endLinear - startLinear) + " ns");
21        System.out.println("Binary Search Time: " + (endBinary - startBinary) + " ns");
22    }
23 }

```

OUTPUT:

Linear Search Time: 5572200 ns

Binary Search Time: 57300 ns



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with a package named 'DeepSkillSearch' containing several Java files. The main editor window shows the source code of 'Timecomplexity.java'. The code defines a package 'DeepSkillSearch', a public class 'Timecomplexity', and a public static void main method. Inside the main method, it initializes an array of 'Product' objects and iterates through them, printing their details. The console at the bottom shows the output of the program, displaying the linear search time as 5572200 ns and the binary search time as 57300 ns.

```
1 package DeepSkillSearch;
2 public class Timecomplexity {
3
4     public static void main(String[] args) {
5         int n = 10000;
6         Product[] products = new Product[n];
7
8         for (int i = 0; i < n; i++) {
9             products[i] = new Product(i, "Product" + i, "Category" + (i % 10));
10        }
11    }
12 }
```

<terminated> Timecomplexity [Java Application] C:\Users\91833\p2\pool\plugins\org.eclipse.justi.openjdk hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\java.exe (19 Jun 2025, 9:37:50 pm - 9:37:52 pm) [pid: 3840]
Linear Search Time: 5572200 ns
Binary Search Time: 57300 ns

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