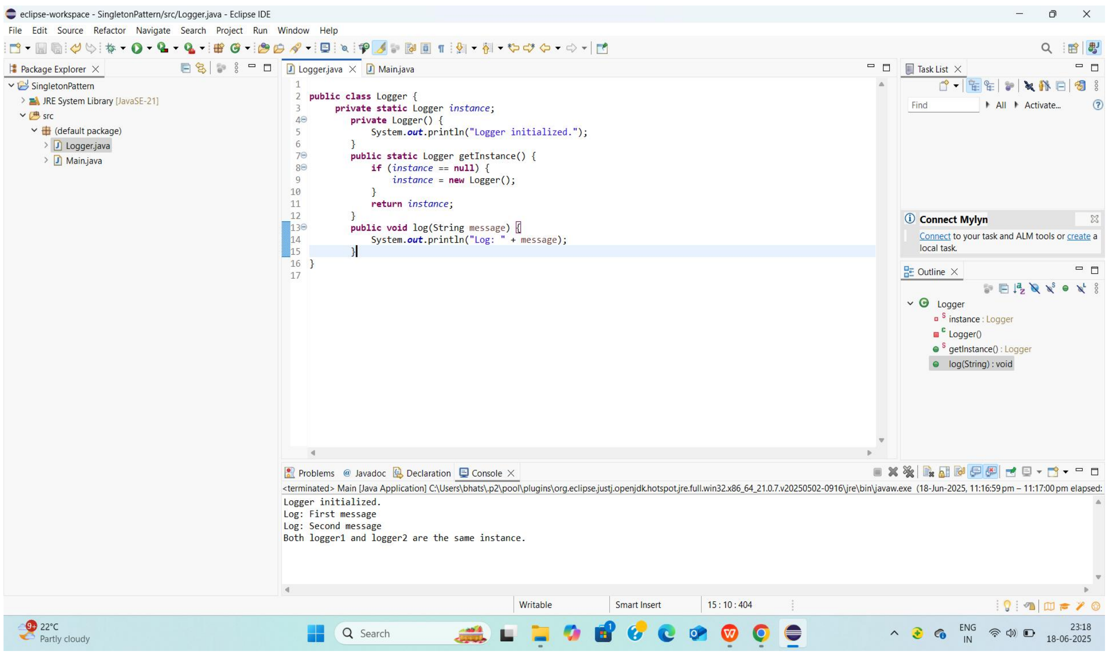
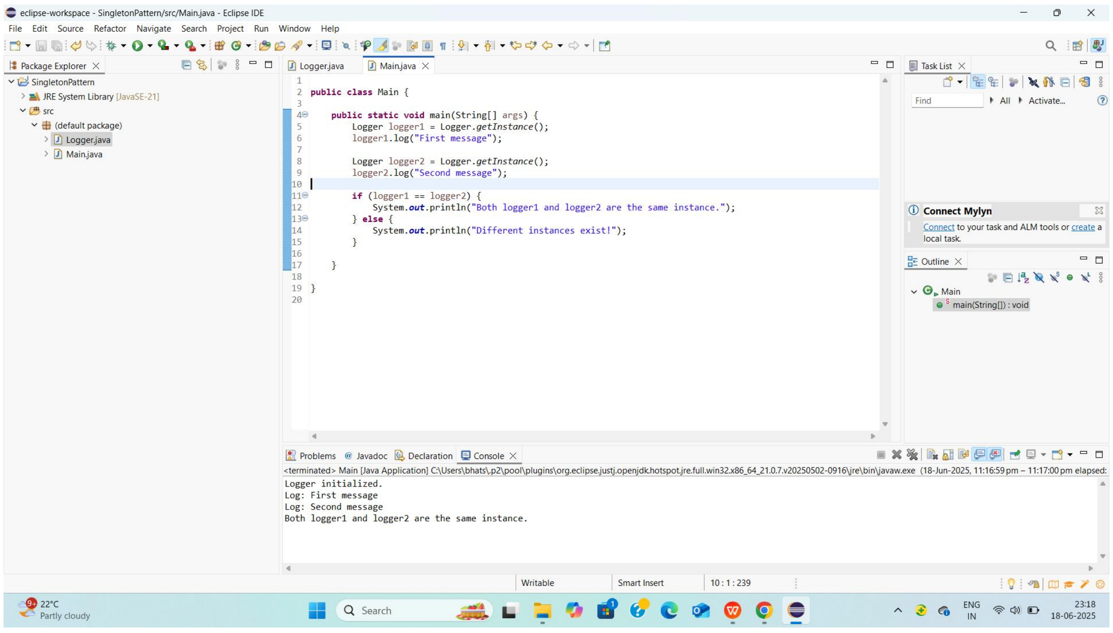
WEEK 1 MANDATORY HANDS ON

1.Implementing the Singleton Pattern



```java

public class Logger {
      private static Logger instance;
         private Logger() {
             System.out.println("Logger initialized.");
         }
         public static Logger getInstance() {
             if (instance == null) {
                 instance = new Logger();
             }
             return instance;
         }
         public void log(String message) {
             System.out.println("Log: " + message);
         }
}

```

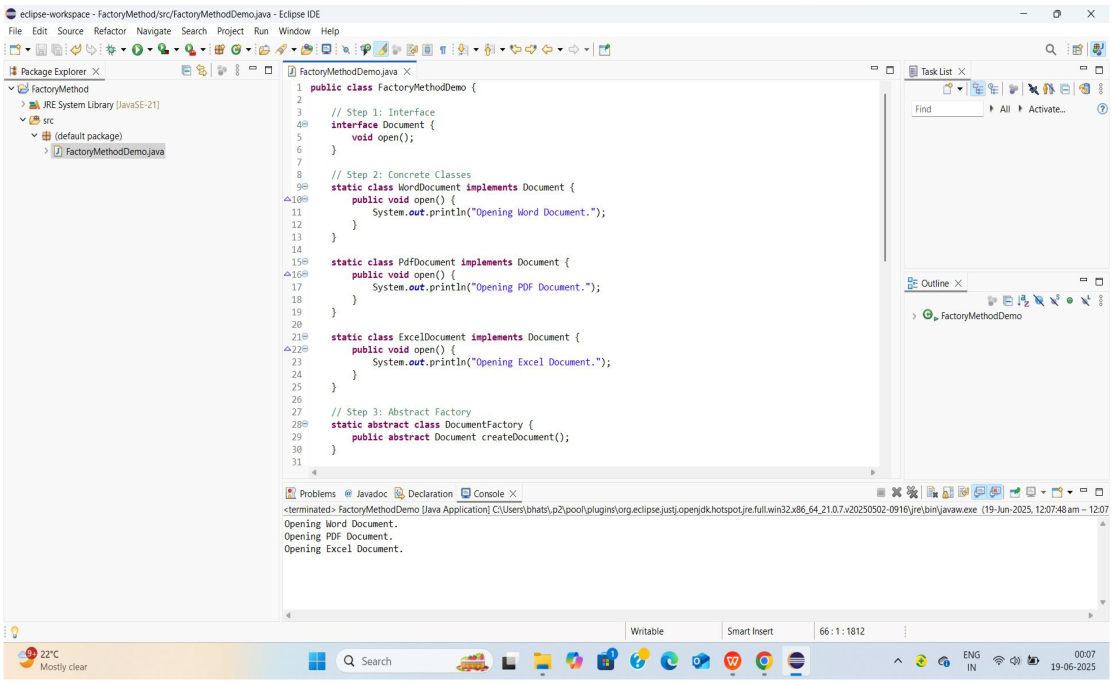
```
<terminated> Main [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (18-Jun-2025, 11:16:59 pm – 11:17:00 pm elapsed:
Logger initialized.
Log: First message
Log: Second message
Both logger1 and logger2 are the same instance.
```



```java

public class Main {

    public static void main(String[] args) {
        Logger logger1 = Logger.getInstance();
        logger1.log("First message");

        Logger logger2 = Logger.getInstance();
        logger2.log("Second message");

        if (logger1 == logger2) {
            System.out.println("Both logger1 and logger2 are the same instance.");
        } else {
            System.out.println("Different instances exist!");
        }

    }

}

```

```
<terminated> Main [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (18-Jun-2025, 11:16:59 pm – 11:17:00 pm elapsed:
Logger initialized.
Log: First message
Log: Second message
Both logger1 and logger2 are the same instance.
```

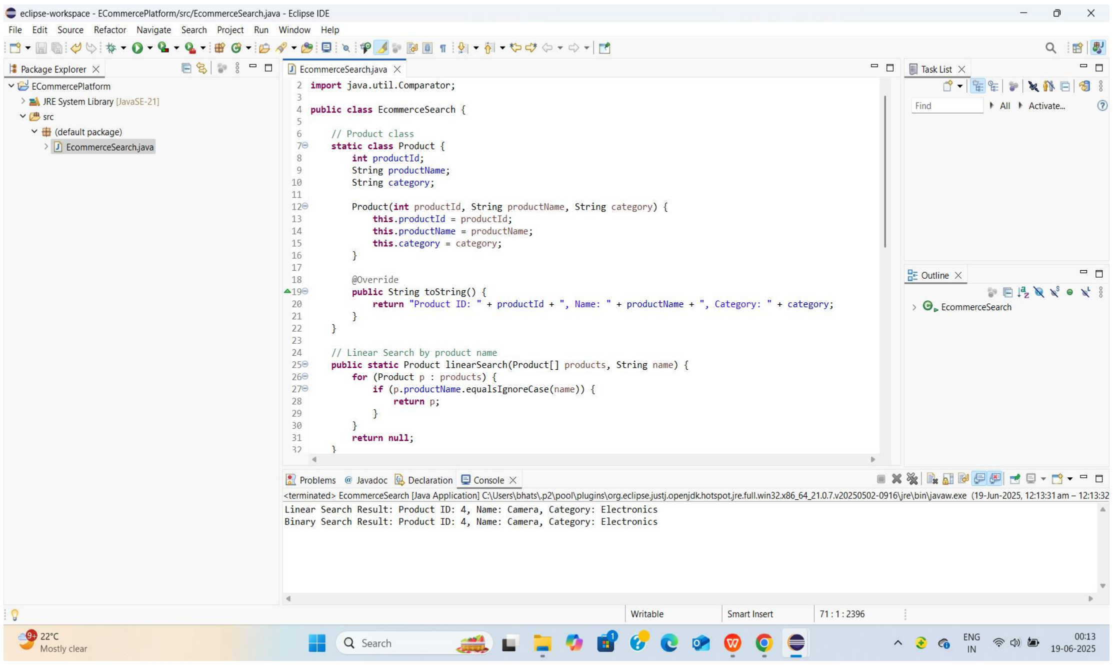
2.Factory Method Pattern



```java
public class FactoryMethodDemo {

    // Step 1: Interface
    interface Document {
        void open();
    }

    // Step 2: Concrete Classes
    static class WordDocument implements Document {
        public void open() {
            System.out.println("Opening Word Document.");
        }
    }

    static class PdfDocument implements Document {
        public void open() {
            System.out.println("Opening PDF Document.");
        }
    }

    static class ExcelDocument implements Document {
        public void open() {
            System.out.println("Opening Excel Document.");
        }
    }

    // Step 3: Abstract Factory
    static abstract class DocumentFactory {
        public abstract Document createDocument();
    }

```

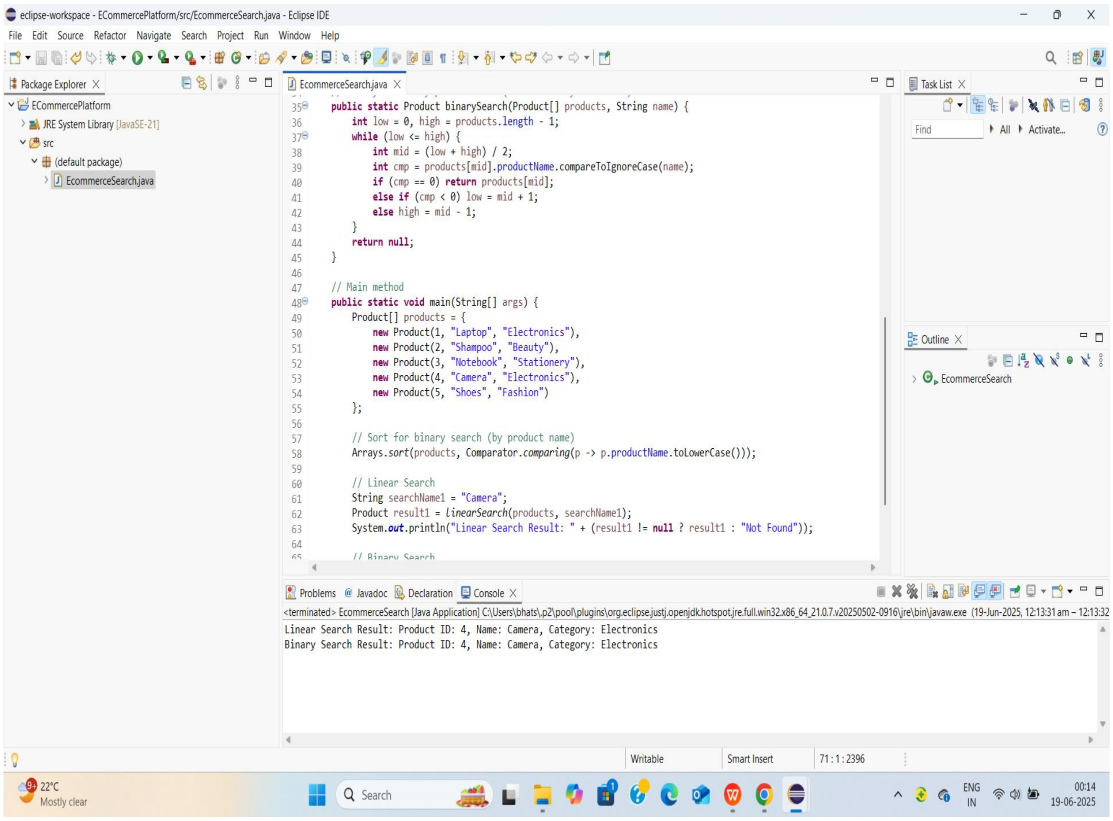
```
<terminated> FactoryMethodDemo [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (19-Jun-2025, 12:07:48 am – 12:07
Opening Word Document.
Opening PDF Document.
Opening Excel Document.
```

3.E-commerce Search Fuction



```java
import java.util.Comparator;

public class EcommerceSearch {

    // Product class
    static class Product {
        int productId;
        String productName;
        String category;

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

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

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

```
<terminated> EcommerceSearch [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (19-Jun-2025, 12:13:31 am – 12:13:32
Linear Search Result: Product ID: 4, Name: Camera, Category: Electronics
Binary Search Result: Product ID: 4, Name: Camera, Category: Electronics
```



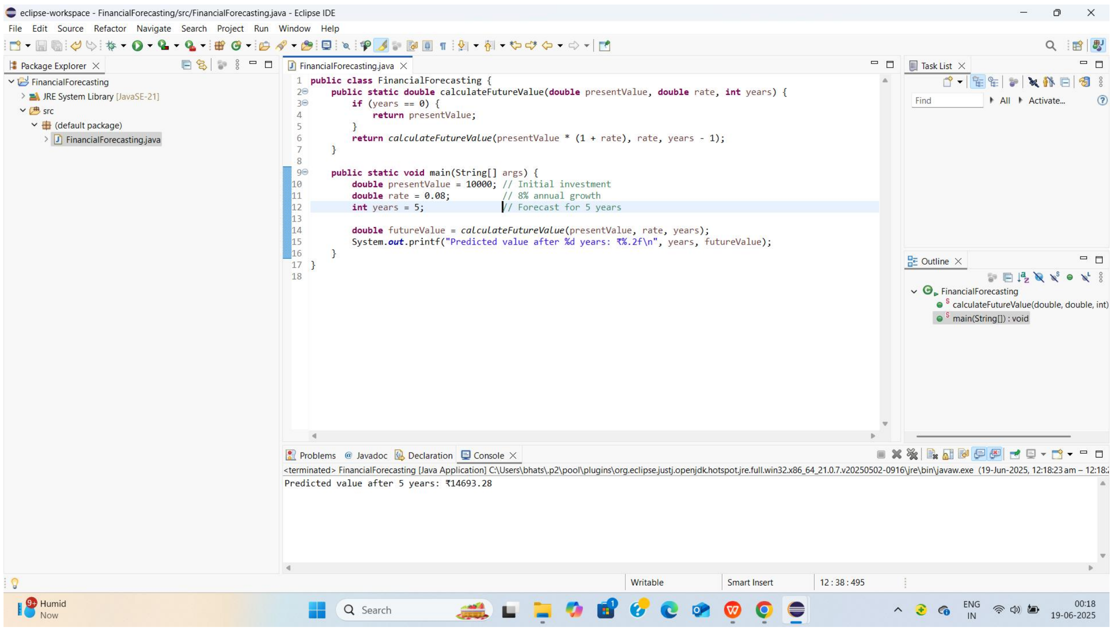
eclipse-workspace - ECommercePlatform/src/EcommerceSearch.java - Eclipse IDE

```java
    public static Product binarySearch(Product[] products, String name) {
        int low = 0, high = products.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            int cmp = products[mid].productName.compareToIgnoreCase(name);
            if (cmp == 0) return products[mid];
            else if (cmp < 0) low = mid + 1;
            else high = mid - 1;
        }
        return null;
    }

    // Main method
    public static void main(String[] args) {
        Product[] products = {
            new Product(1, "Laptop", "Electronics"),
            new Product(2, "Shampoo", "Beauty"),
            new Product(3, "Notebook", "Stationery"),
            new Product(4, "Camera", "Electronics"),
            new Product(5, "Shoes", "Fashion")
        };

        // Sort for binary search (by product name)
        Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

        // Linear Search
        String searchName1 = "Camera";
        Product result1 = linearSearch(products, searchName1);
        System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Not Found"));

        // Binary Search
```

<terminated> EcommerceSearch [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (19-Jun-2025, 12:13:31 am – 12:13:32

```
Linear Search Result: Product ID: 4, Name: Camera, Category: Electronics
Binary Search Result: Product ID: 4, Name: Camera, Category: Electronics
```

4.Financial Forecasting



eclipse-workspace - FinancialForecasting/src/FinancialForecasting.java - Eclipse IDE

```java
public class FinancialForecasting {
    public static double calculateFutureValue(double presentValue, double rate, int years) {
        if (years == 0) {
            return presentValue;
        }
        return calculateFutureValue(presentValue * (1 + rate), rate, years - 1);
    }

    public static void main(String[] args) {
        double presentValue = 10000; // Initial investment
        double rate = 0.08;          // 8% annual growth
        int years = 5;               // Forecast for 5 years

        double futureValue = calculateFutureValue(presentValue, rate, years);
        System.out.printf("Predicted value after %d years: ₹%.2f\n", years, futureValue);
    }
}
```

<terminated> FinancialForecasting [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (19-Jun-2025, 12:18:23 am – 12:18:

```
Predicted value after 5 years: ₹14693.28
```