Cognizant Digital Nurture 4.0

Engineering concepts

DSA

Exercise 1: Inventory Management System

CODE:

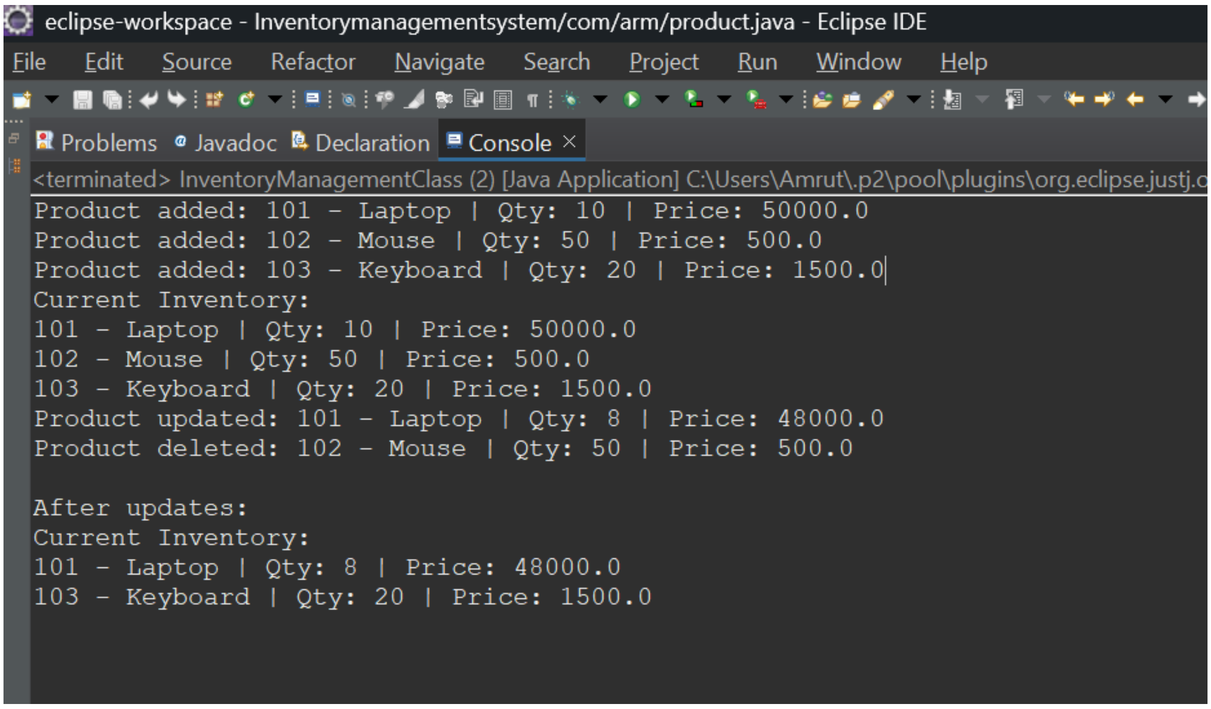
InventoryManagementClass.java

```
package com.arm;
import java.util.HashMap;

public class InventoryManagementClass {

    private HashMap<Integer, Product> inventory;

    public InventoryManagementClass() {
        inventory = new HashMap<>();
    }

    // Add a product
    public void addProduct(Product product) {
        inventory.put(product.productId, product);
        System.out.println("Product added: " + product);
    }

    // Update a product
    public void updateProduct(int productId, int quantity, double price) {
        Product p = inventory.get(productId);
        if (p != null) {
            p.quantity = quantity;
            p.price = price;
            System.out.println("Product updated: " + p);
        } else {
            System.out.println("Product ID " + productId + " not found!");
        }
    }

    // Delete a product
    public void deleteProduct(int productId) {
        Product removed = inventory.remove(productId);
        if (removed != null) {
            System.out.println("Product deleted: " + removed);
        } else {
            System.out.println("Product ID " + productId + " not found!");
        }
    }

    // Display all products
    public void displayInventory() {
        if (inventory.isEmpty()) {
            System.out.println("Inventory is empty.");
        } else {
            System.out.println("Current Inventory:");
            for (Product p : inventory.values()) {
                System.out.println(p);
            }
        }
    }

    // Main method for testing
    public static void main(String[] args) {
        InventoryManagementClass inv = new InventoryManagementClass();

        inv.addProduct(new Product(101, "Laptop", 10, 50000));
        inv.addProduct(new Product(102, "Mouse", 50, 500));
        inv.addProduct(new Product(103, "Keyboard", 20, 1500));

        inv.displayInventory();

        inv.updateProduct(101, 8, 48000);
        inv.deleteProduct(102);

        System.out.println("\nAfter updates:");
        inv.displayInventory();
    }
}
```

Product.java

```
package com.arm;
class Product {
    int productId;
    String productName;
    int quantity;
    double price;

    public Product(int productId, String productName, int quantity, double price) {
        this.productId = productId;
        this.productName = productName;
        this.quantity = quantity;
        this.price = price;
    }


    public String toString() {
        return productId + " - " + productName + " | Qty: " + quantity + " | Price: " + price;
    }
}
```

OUTPUT:


eclipse-workspace - Inventorymanagementsystem/com/arm/product.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console

```
<terminated> InventoryManagementClass (2) [Java Application] C:\Users\Amrut\.p2\pool\plugins\org.eclipse.justj.c
Product added: 101 - Laptop | Qty: 10 | Price: 50000.0
Product added: 102 - Mouse | Qty: 50 | Price: 500.0
Product added: 103 - Keyboard | Qty: 20 | Price: 1500.0
Current Inventory:
101 - Laptop | Qty: 10 | Price: 50000.0
102 - Mouse | Qty: 50 | Price: 500.0
103 - Keyboard | Qty: 20 | Price: 1500.0
Product updated: 101 - Laptop | Qty: 8 | Price: 48000.0
Product deleted: 102 - Mouse | Qty: 50 | Price: 500.0

After updates:
Current Inventory:
101 - Laptop | Qty: 8 | Price: 48000.0
103 - Keyboard | Qty: 20 | Price: 1500.0
```

Exercise 2: E-commerce Platform Search Function

Code:

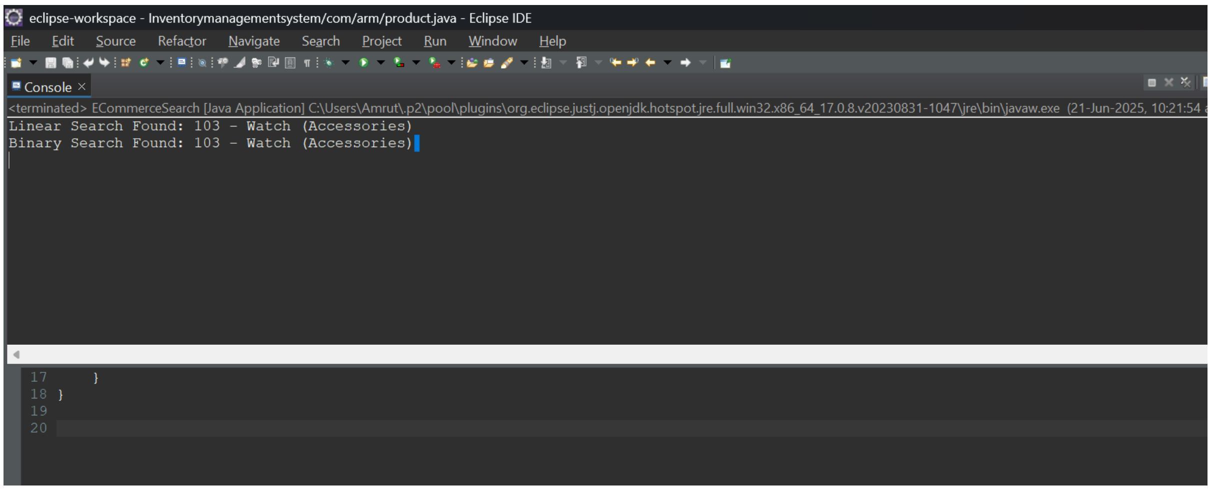
Product.java

```
package com.arm;
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;
    }
    public String toString() {
        return productId + " - " + productName + " (" + category + ")";
    }
}
```

EcommerceSearch.java

```
package com.arm;
import java.util.Arrays;

public class ECommerceSearch {
    public static void main(String[] args) {
        Product[] products = {
            new Product(105, "Shoes", "Fashion"),
            new Product(101, "Smartphone", "Electronics"),
            new Product(104, "Bag", "Fashion"),
            new Product(103, "Watch", "Accessories"),
            new Product(102, "Headphones", "Electronics")
        };

        int searchId = 103;

        // Linear search on unsorted array
        Product resultLinear = linearSearch(products, searchId);
        if (resultLinear != null) {
            System.out.println("Linear Search Found: " + resultLinear);
        } else {
            System.out.println("Linear Search: Product not found");
        }

        // Sort array for binary search
        Arrays.sort(products, (a, b) -> Integer.compare(a.productId, b.productId));
        // Binary search on sorted array
        Product resultBinary = binarySearch(products, searchId);
        if (resultBinary != null) {
            System.out.println("Binary Search Found: " + resultBinary);
        } else {
            System.out.println("Binary Search: Product not found");
        }
    }

    public static Product linearSearch(Product[] products, int productId) {
        for (Product p : products) {
            if (p.productId == productId) {
                return p;
            }
        }
        return null;
    }

    public static Product binarySearch(Product[] products, int productId) {
        int low = 0;
        int high = products.length - 1;

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (products[mid].productId == productId) {
                return products[mid];
            } else if (products[mid].productId < productId) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return null;
    }
}
```

Output:



```
<terminated> ECommerceSearch [Java Application] C:\Users\Amrut\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (21-Jun-2025, 10:21:54
Linear Search Found: 103 - Watch (Accessories)
Binary Search Found: 103 - Watch (Accessories)
```

```
    }
}
```

Design Patterns and Principles

Exercise 1: Implementing the Singleton Pattern

CODE:

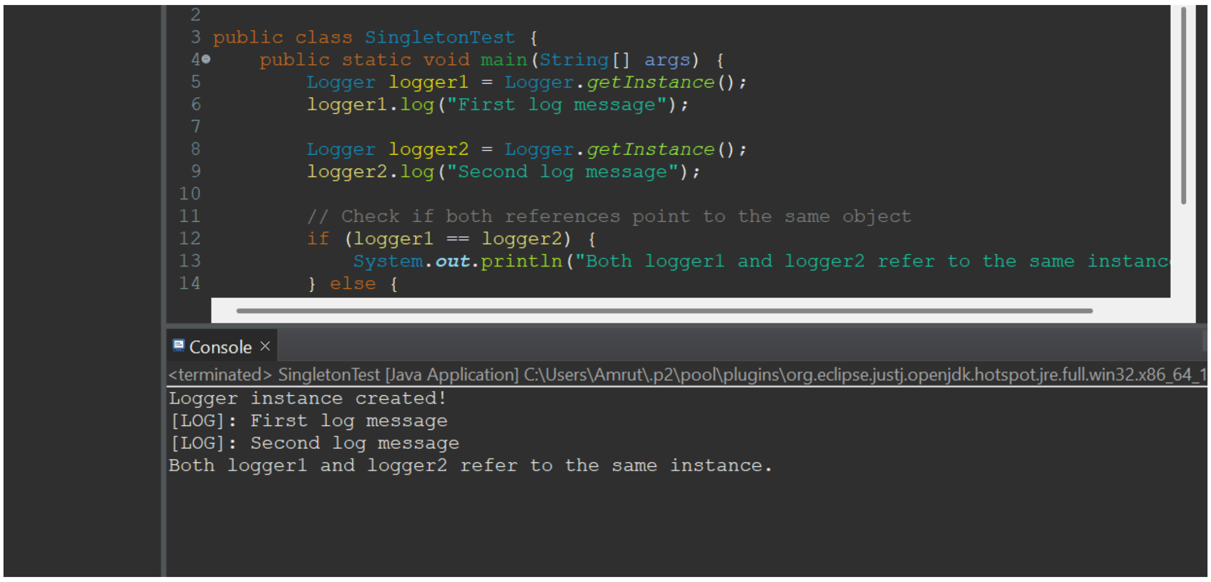
Logger.java

```
package singleton;


public class Logger {
    // Private static instance
    private static Logger instance;


    // Private constructor
    private Logger() {
        System.out.println("Logger instance created!");
    }

    // Public static method to get the instance
    public static Logger getInstance() {
        if (instance == null) {
            instance = new Logger();
        }
        return instance;
    }

    // Logging method
    public void log(String message) {
        System.out.println("[LOG]: " + message);
    }
}
```

SingletonTest.java

```
package singleton;

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

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

        // Check if both references point to the same object
        if (logger1 == logger2) {
            System.out.println("Both logger1 and logger2 refer to the same instance.");
        } else {
            System.out.println("Different instances exist. Singleton not working!");
        }
    }
}
```

OUTPUT:



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

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

        // Check if both references point to the same object
        if (logger1 == logger2) {
            System.out.println("Both logger1 and logger2 refer to the same instanc
        } else {
```

Console

<terminated> SingletonTest [Java Application] C:\Users\Amrut\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1

```
Logger instance created!
[LOG]: First log message
[LOG]: Second log message
Both logger1 and logger2 refer to the same instance.
```

Exercise 2: Implementing the Factory Method Pattern

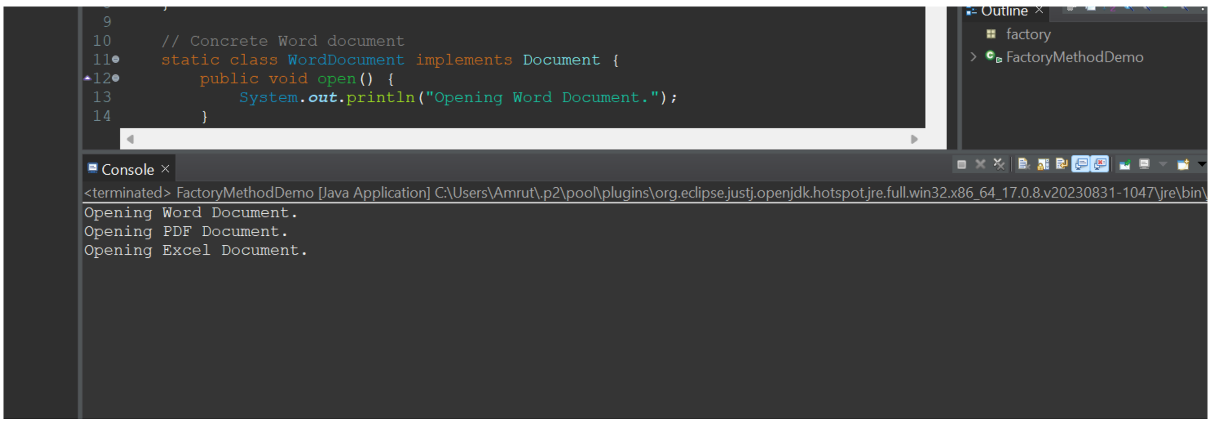
Code:

```
package factory;

public class FactoryMethodDemo {

    // Document interface
    interface Document {
        void open();
    }

    // Concrete Word document
    static class WordDocument implements Document {
        public void open() {
            System.out.println("Opening Word Document.");
        }
    }
```

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

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

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

// Word document factory
static class WordDocumentFactory extends DocumentFactory {
    public Document createDocument() {
        return new WordDocument();
    }
}

// PDF document factory
static class PdfDocumentFactory extends DocumentFactory {
    public Document createDocument() {
        return new PdfDocument();
    }
}
```

```java
    // Excel document factory
    static class ExcelDocumentFactory extends DocumentFactory {
        public Document createDocument() {
            return new ExcelDocument();
        }
    }

    // Main method to test
    public static void main(String[] args) {
        DocumentFactory wordFactory = new WordDocumentFactory();
        Document wordDoc = wordFactory.createDocument();
        wordDoc.open();

        DocumentFactory pdfFactory = new PdfDocumentFactory();
        Document pdfDoc = pdfFactory.createDocument();
        pdfDoc.open();

        DocumentFactory excelFactory = new ExcelDocumentFactory();
        Document excelDoc = excelFactory.createDocument();
        excelDoc.open();
    }
}
```

Output:



```java
    // Concrete Word document
    static class WordDocument implements Document {
        public void open() {
            System.out.println("Opening Word Document.");
        }
```

```
<terminated> FactoryMethodDemo [Java Application] C:\Users\Amrut\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\
Opening Word Document.
Opening PDF Document.
Opening Excel Document.
```