MINI – PROJECT
CRUD using collections

Product Management System Overview

Objective

To develop a comprehensive Java-based Product Management system that provides an intuitive and efficient way to manage product information. The system will enable users to perform CRUD (Create, Read, Update, Delete) operations on product data, ensuring that products are accurately managed and maintained.

Key Features

1. **Product Addition:**
    - Users can add new products to the system by providing details such as ID, name, and price.
2. **Product Retrieval:**
    - Users can retrieve and view details of a specific product or get a list of all products available in the system.
3. **Product Update:**
    - Users can update the information of existing products, including changing the name and price.
4. **Product Deletion:**
    - Users can delete products from the system based on product ID.
5. **Data Validation:**
    - The system will validate product details before adding or updating to ensure data integrity.
6. **Error Handling:**
    - Robust error handling for invalid operations, such as trying to add a product with a duplicate ID or update a non-existent product.
7. **In-Memory Data Storage:**
    - The system uses an in-memory repository to store product data, simulating a database without persistent storage.
8. **Service Layer:**

- A service layer manages interactions between the user interface and the data access layer, providing a clear API for CRUD operations.

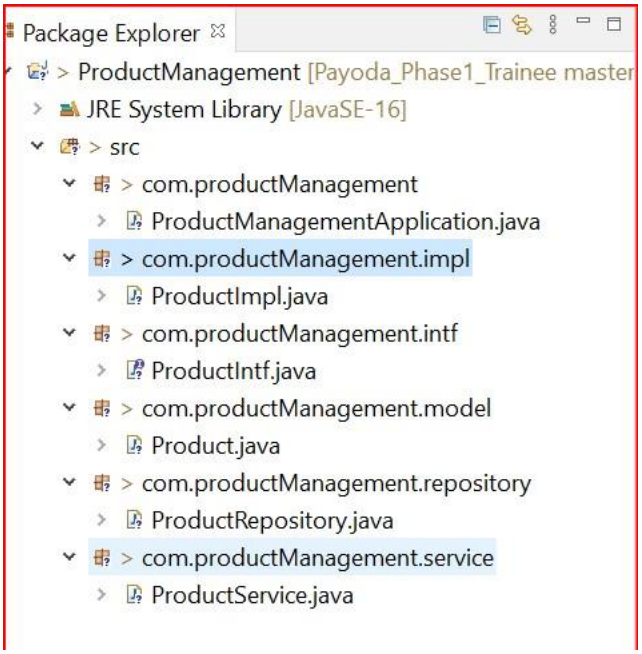
Expected User Interaction

1. **Add Product:**
    - **Action:** Users enter product details through a command-line interface or graphical user interface.
    - **Interaction:** System prompts for product ID, name, and price. After submission, the system confirms the addition.
2. **Retrieve Product:**
    - **Action:** Users request to view a specific product or all products.
    - **Interaction:** System displays the details of the requested product or lists all products.
3. **Update Product:**
    - **Action:** Users provide updated product details for an existing product.
    - **Interaction:** System retrieves the product by ID, applies updates, and confirms the changes.
4. **Delete Product:**
    - **Action:** Users request to delete a product by providing the product ID.
    - **Interaction:** System removes the product from the repository and confirms deletion.
5. **Handle Errors:**
    - **Action:** Users encounter errors during invalid operations.
    - **Interaction:** System displays error messages and provides instructions for corrective actions.

Technologies Used

1. **Java:**
    - **Description:** The primary programming language used to implement the application's functionality.
    - **Usage:** Core logic, CRUD operations, data modeling, and error handling.
2. **Collections Framework:**
    - **Description:** Java Collections Framework (List, Set, Map) used for in-memory data management.
    - **Usage:** Store and manage product data within the application.

Project structure:



Code:

com.productManagement:

ProductManagementApplication.java file:

```
package com.productManagement;


import java.util.ArrayList;
import java.util.Scanner;


import com.productManagement.model.Product;
import com.productManagement.service.ProductService;


/**
```

```
* @author sanjeevkumar.v
*
*/
public class ProductManagementApplication {



@SuppressWarnings("resource")
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
ProductService productService = new ProductService();
System.out.println("----------WELCOME TO ADMIN DASHBOARD----------");
try {
char in1;
do {
System.out.print("a. Add Products\nb. Delete Products\nc. Get Products\nd. Update Products\ne. Exit\nEnter your choice: ");
in1 = sc.next().charAt(0);
sc.nextLine();
switch(in1) {
case 'a':
System.out.println("Enter Product Details [ProductName : Description : Price : SupllierId : Goods : Goods Description]: ");
String productInfo = sc.nextLine();
String product = productService.createProduct(productInfo);
if(product != null) {
System.out.println(product);
```

```
}
else {
System.out.println("Product not added.");
}
break;
case 'b':
System.out.println("Enter ProductId: ");
String productId = sc.nextLine();
if(productService.deleteProduct(productId)) {
System.out.println("Successfully product deleted.");
}
else {
System.out.println("Product not deleted/exists.");
}
break;
case 'c':
ArrayList<Product> list = (ArrayList<Product>) productService.getAllProduct();
System.out.println();
if(!list.isEmpty()) {
System.out.printf("%-30s%-30s%-30s%-30s%-50s%-15s%-20s", "productId", "productName",
"Description", "Unit Price", "supplier", "goods", "goodsDescription");
System.out.println();
System.out.println("----------------------------------------------------------------------------------------
------");
   for (Product p : list) {
      System.out.println(p);
```

```
    }
    System.out.println();
}
else {
System.out.println("Products are not available.");
}
break;
case 'd':
System.out.println("Enter Product Details [ProductId : ProductName : Description : Price :
SupllierId : Goods : Goods Description]: ");
String productInf = sc.nextLine();
String productDet = productService.updateProduct(productInf);
if(productDet != null) {
System.out.println(productDet);
}
else {
System.out.println("Product details not updated.");
}
break;
case 'e':
System.out.println(".....Thank you.....\nClosing Application.");
break;
default:
System.out.println("Your choice is wrong. Please select correct option.");
break;
}
```

```
}while(in1 !='e');
} catch(Exception e) {
System.out.println();
System.out.println("Something error. Please try again.....");
System.out.println();
sc.nextLine();
main(args);
}
}
}
```

com.productManagement.impl:

ProductImpl.java file:

```
package com.productManagement.impl;

import java.util.List;

import com.productManagement.intf.ProductIntf;
import com.productManagement.model.Product;
import com.productManagement.repository.ProductRepository;

/**
* @author sanjeevkumar.v
*
*/
public class ProductImpl implements ProductIntf {
```

```
private ProductRepository repository = new ProductRepository();

@Override
public boolean createProduct(Product product) {
// TODO Auto-generated method stub
return repository.addProduct(product);
}

@Override
public Product getProduct(String productId) {
// TODO Auto-generated method stub
return repository.getProduct(productId);
}

@Override
public boolean updateProduct(Product product) {
// TODO Auto-generated method stub
return repository.updateProduct(product);
}

@Override
public boolean deleteProduct(String productId) {
// TODO Auto-generated method stub
return repository.deleteProduct(productId);
}
```

```
@Override
public List<Product> getAllProduct() {
// TODO Auto-generated method stub
return repository.getAllProduct();
}
}
```

com.productManagement.intf:

ProductIntf.java file:

```
package com.productManagement.intf;

import java.util.List;

import com.productManagement.model.Product;

/**
* @author sanjeevkumar.v
*
*/
public interface ProductIntf {

boolean createProduct(Product product);
Product getProduct(String productId);
boolean updateProduct(Product product);
boolean deleteProduct(String productId);
```

```
List<Product> getAllProduct();


}
```

com.productManagement.model:

ProductIntf.java file:

```
package com.productManagement.model;



/**
* @author sanjeevkumar.v
*
*/
public class Product {

private String productId;
private String productName;
private String description;
private double unitPrice;
private String supplierInfo;
private String goods;
private String goodsDescription;



public Product(String productId, String productName, String description, double unitPrice, String supplierInfo,
```

```
String goods, String goodsDescription) {
super();
this.productId = productId;
this.productName = productName;
this.description = description;
this.unitPrice = unitPrice;
this.supplierInfo = supplierInfo;
this.goods = goods;
this.goodsDescription = goodsDescription;
}
public String getProductId() {
return productId;
}
public void setProductId(String productId) {
this.productId = productId;
}
public String getProductName() {
return productName;
}
public void setProductName(String productName) {
this.productName = productName;
}
public String getDescription() {
return description;
}
public void setDescription(String description) {
```

```
this.description = description;
}
public double getUnitPrice() {
return unitPrice;
}
public void setUnitPrice(double unitPrice) {
this.unitPrice = unitPrice;
}
public String getSupplierInfo() {
return supplierInfo;
}
public void setSupplierInfo(String supplierInfo) {
this.supplierInfo = supplierInfo;
}
public String getGoods() {
return goods;
}
public void setGoods(String goods) {
this.goods = goods;
}
public String getGoodsDescription() {
return goodsDescription;
}
public void setGoodsDescription(String goodsDescription) {
this.goodsDescription = goodsDescription;
}
```

```
@Override

public String toString() {

System.out.printf("%-30s%-30s%-30s%-30.2f%-50s%-15s%-20s", productId, productName,
description, unitPrice, supplierInfo,

goods, goodsDescription);

return "";

}

}
```

com.productManagement.repository:

ProductRepository.java file:

```
package com.productManagement.repository;

import java.util.*;


import com.productManagement.model.Product;


/**

* @author sanjeevkumar.v

*

*/

public class ProductRepository {


private Map<String, Product> productMap = new HashMap<String, Product>();


public boolean addProduct(Product productInfo) {

productMap.put(productInfo.getProductId(), productInfo);
```

```
return productMap.containsKey(productInfo.getProductId());
}

public Product getProduct(String productId) {
return productMap.get(productId);
}

public boolean updateProduct(Product productInfo) {
return productMap.put(productInfo.getProductId(), productInfo) != null;
}

public boolean deleteProduct(String industryId) {
productMap.remove(industryId);
return !productMap.containsKey(industryId);
}

public List<Product> getAllProduct() {
   return new ArrayList<>(productMap.values());
}
}
```

com.productManagement.service:

ProductService.java file:

```
package com.productManagement.service;

import java.text.SimpleDateFormat;
```

```
import java.util.Date;

import java.util.List;


import com.productManagement.impl.ProductImpl;

import com.productManagement.intf.ProductIntf;

import com.productManagement.model.Product;


public class ProductService {



private ProductIntf productIntf = new ProductImpl();


public String createProduct(String productInfo) {


String[] product = productInfo.split(":");


String productId = "PROD" + generateUniqueId();

Product productDetails = new Product(productId, product[0], product[1],
Integer.parseInt(product[2]), product[3],

product[4], product[5]);

if(productIntf.createProduct(productDetails)) {

return "ProductId: "+ productId;

}


return null;

}
```

```
public Product getProduct(String productId) {
Product product = productIntf.getProduct(productId);

if(product != null) {
return product;
}
return null;
}

public String updateProduct(String productInfo) {

String[] product = productInfo.split(":");

Product productDetails = new Product(product[0], product[1], product[2],
Integer.parseInt(product[3]), product[4],
product[5], product[6]);
return productIntf.updateProduct(productDetails) == true ? "Updated Successfully" : "Not
updated.";
}

public boolean deleteProduct(String productId) {
return productIntf.deleteProduct(productId);
}

 public List<Product> getAllProduct() {
```

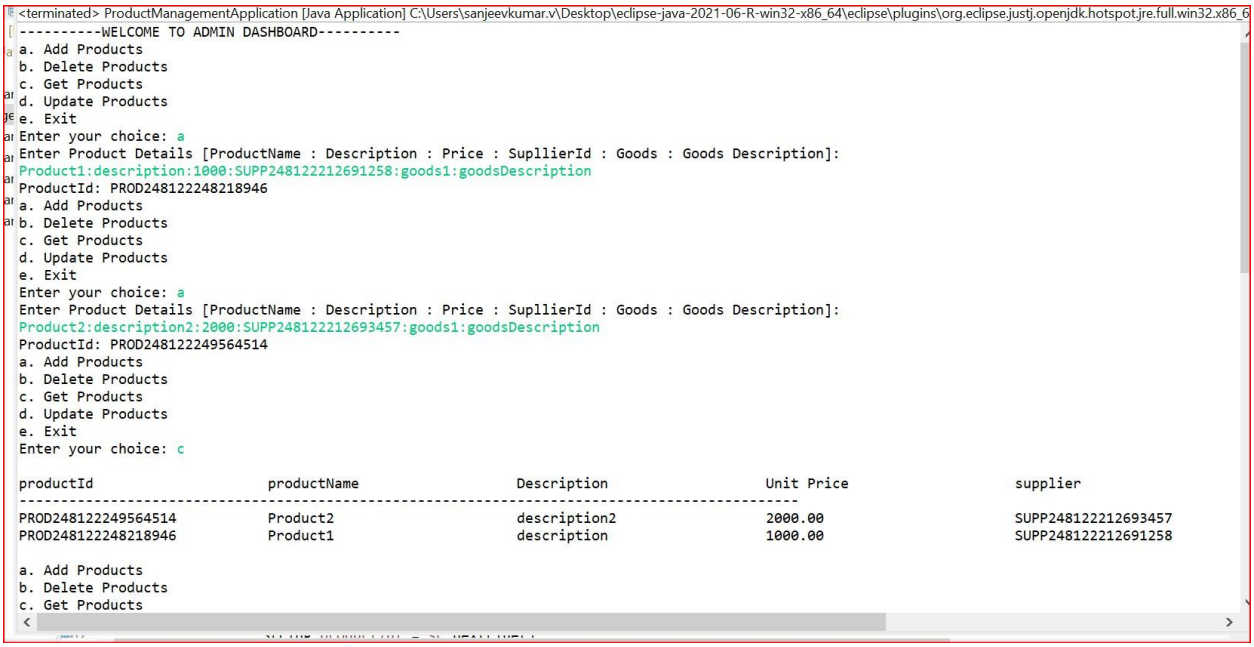
```
      return productIntf.getAllProduct();
   }

public String generateUniqueId() {
     return generateSCMId();
   }

private String generateSCMId() {
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyMddHHmmSS");
    String timestamp = dateFormat.format(new Date());
   int randomSuffix = (int) (Math.random() * 1000); // Add a random suffix for uniqueness
   return timestamp + String.format("%03d", randomSuffix);
   }

}
```

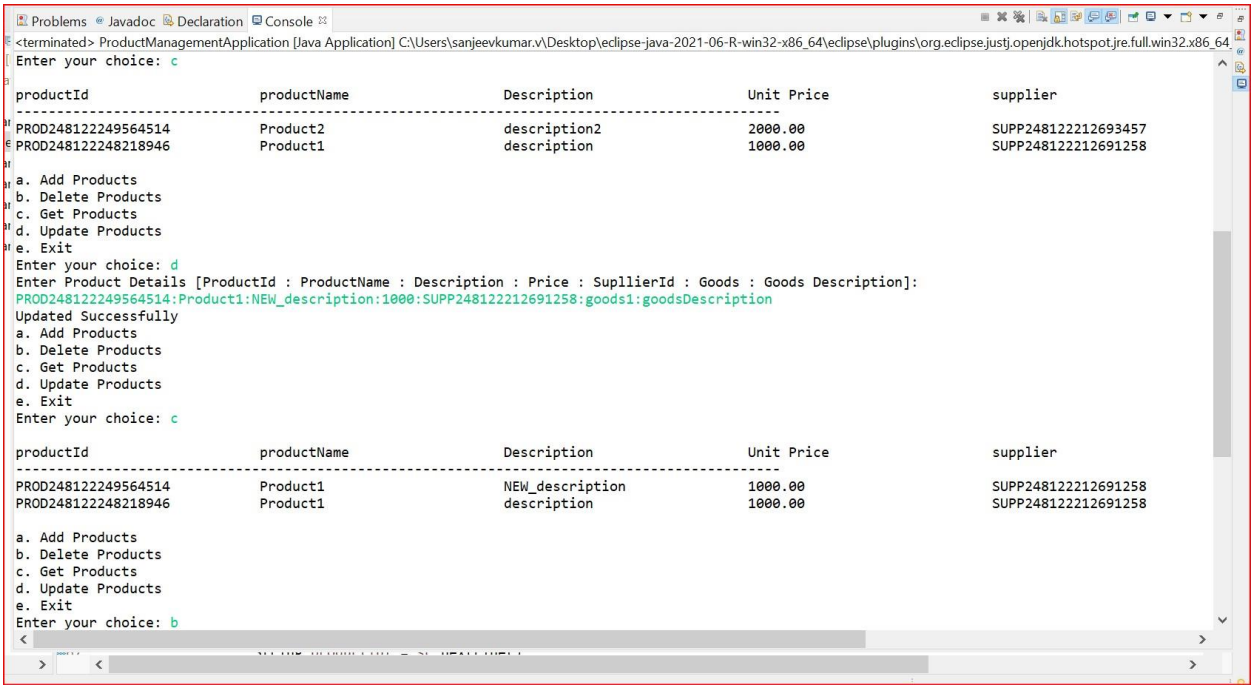
Output:



```
<terminated> ProductManagementApplication [Java Application] C:\Users\sanjeevkumar.v\Desktop\eclipse-java-2021-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_6
----------WELCOME TO ADMIN DASHBOARD----------
a. Add Products
b. Delete Products
c. Get Products
d. Update Products
e. Exit
Enter your choice: a
Enter Product Details [ProductName : Description : Price : SupllierId : Goods : Goods Description]:
Product1:description:1000:SUPP248122212691258:goods1:goodsDescription
ProductId: PROD248122248218946
a. Add Products
b. Delete Products
c. Get Products
d. Update Products
e. Exit
Enter your choice: a
Enter Product Details [ProductName : Description : Price : SupllierId : Goods : Goods Description]:
Product2:description2:2000:SUPP248122212693457:goods1:goodsDescription
ProductId: PROD248122249564514
a. Add Products
b. Delete Products
c. Get Products
d. Update Products
e. Exit
Enter your choice: c

productId                   productName                 Description                 Unit Price                  supplier
-------------------------------------------------------------------------------------------------
PROD248122249564514         Product2                    description2                2000.00                     SUPP248122212693457
PROD248122248218946         Product1                    description                 1000.00                     SUPP248122212691258

a. Add Products
b. Delete Products
c. Get Products
```



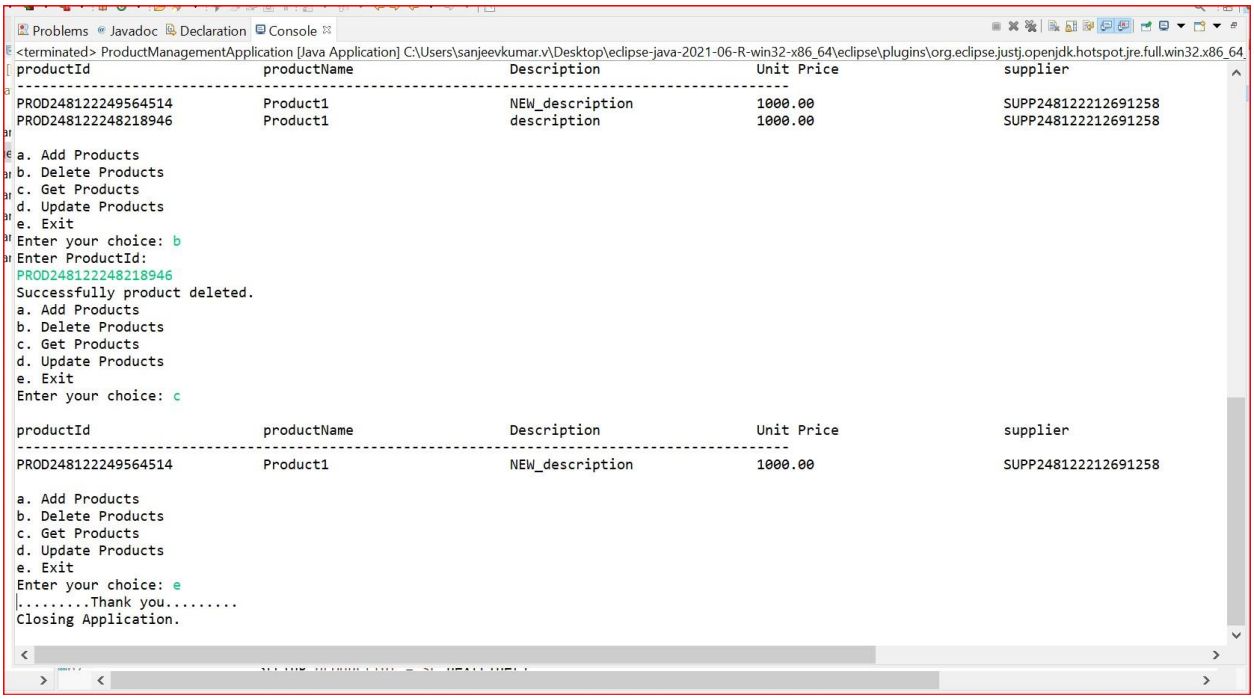
```
Problems  Javadoc  Declaration  Console
<terminated> ProductManagementApplication [Java Application] C:\Users\sanjeevkumar.v\Desktop\eclipse-java-2021-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
Enter your choice: c

productId                   productName                 Description                 Unit Price                  supplier
-------------------------------------------------------------------------------------------------
PROD248122249564514         Product2                    description2                2000.00                     SUPP248122212693457
PROD248122248218946         Product1                    description                 1000.00                     SUPP248122212691258

a. Add Products
b. Delete Products
c. Get Products
d. Update Products
e. Exit
Enter your choice: d
Enter Product Details [ProductId : ProductName : Description : Price : SupllierId : Goods : Goods Description]:
PROD248122249564514:Product1:NEW_description:1000:SUPP248122212691258:goods1:goodsDescription
Updated Successfully
a. Add Products
b. Delete Products
c. Get Products
d. Update Products
e. Exit
Enter your choice: c

productId                   productName                 Description                 Unit Price                  supplier
-------------------------------------------------------------------------------------------------
PROD248122249564514         Product1                    NEW_description             1000.00                     SUPP248122212691258
PROD248122248218946         Product1                    description                 1000.00                     SUPP248122212691258

a. Add Products
b. Delete Products
c. Get Products
d. Update Products
e. Exit
Enter your choice: b
```



Use Case Specifications

1. Use Case: Add Product

Use Case ID: UC01

Name: Add Product

Description: Allows a user to add a new product to the system.

Actors: Product Manager, Administrator

Preconditions: The user must be authenticated and have the necessary permissions.

Postconditions: The new product is added to the system and stored in the repository.

Main Flow:

1. The user provides product details: ID, name, and price.
2. The system validates the product details.
3. The system creates a new `Product` object.
4. The system adds the `Product` object to the repository.

5. The system confirms that the product has been successfully added.

Alternative Flow:

- **Generate Product ID:** If the product data is generate (e.g., missing fields or Generate ID), the system displays an message and requests correct information.

Exceptions:

- **Repository Error:** If the repository fails to store the product, an error message is displayed.

2. Use Case: Update Product

Use Case ID: UC02

Name: Update Product

Description: Allows a user to update the details of an existing product.

Actors: Product Manager, Administrator

Preconditions: The product must exist in the system.

Postconditions: The product details are updated in the repository.

Main Flow:

1. The user provides the updated product details (ID, name, price).
2. The system retrieves the `Product` object from the repository using the ID.
3. The system validates the updated product details.
4. The system updates the existing `Product` object with new details.
5. The system saves the updated `Product` object in the repository.
6. The system confirms that the product has been successfully updated.

Alternative Flow:

- **Product Not Found:** If the product with the given ID does not exist, the system displays an error message.

Exceptions:

- **Repository Error:** If the repository fails to update the product, an error message is displayed.

3. Use Case: Delete Product

Use Case ID: UC03

Name: Delete Product

Description: Allows a user to delete a product from the system.

Actors: Product Manager, Administrator

Preconditions: The product must exist in the system.

Postconditions: The product is removed from the repository.

Main Flow:

1. The user provides the ID of the product to be deleted.
2. The system retrieves the `Product` object from the repository using the ID.
3. The system removes the `Product` object from the repository.
4. The system confirms that the product has been successfully deleted.

Alternative Flow:

- **Product Not Found:** If the product with the given ID does not exist, the system displays an error message.

Exceptions:

- **Repository Error:** If the repository fails to delete the product, an error message is displayed.

5. Use Case: Get All Products

Use Case ID: UC04

Name: Get All Products

Description: Allows a user to retrieve a list of all products in the system.

Actors: Product Manager, Administrator

Preconditions: The system must have products stored in the repository.

Postconditions: A list of all products is displayed to the user.

Main Flow:

1. The user requests the list of all products.
2. The system retrieves the list of all `Product` objects from the repository.
3. The system displays the list of products to the user.

Exceptions:

- **Repository Error:** If the repository fails to retrieve the list of products, an error message is displayed.