North South University

Lab Project

Project Name: Store Management System

Course: CSE215L

Section: 17

Submitted To: Farzana Islam

Date: 08/11/2023

Submitted By:

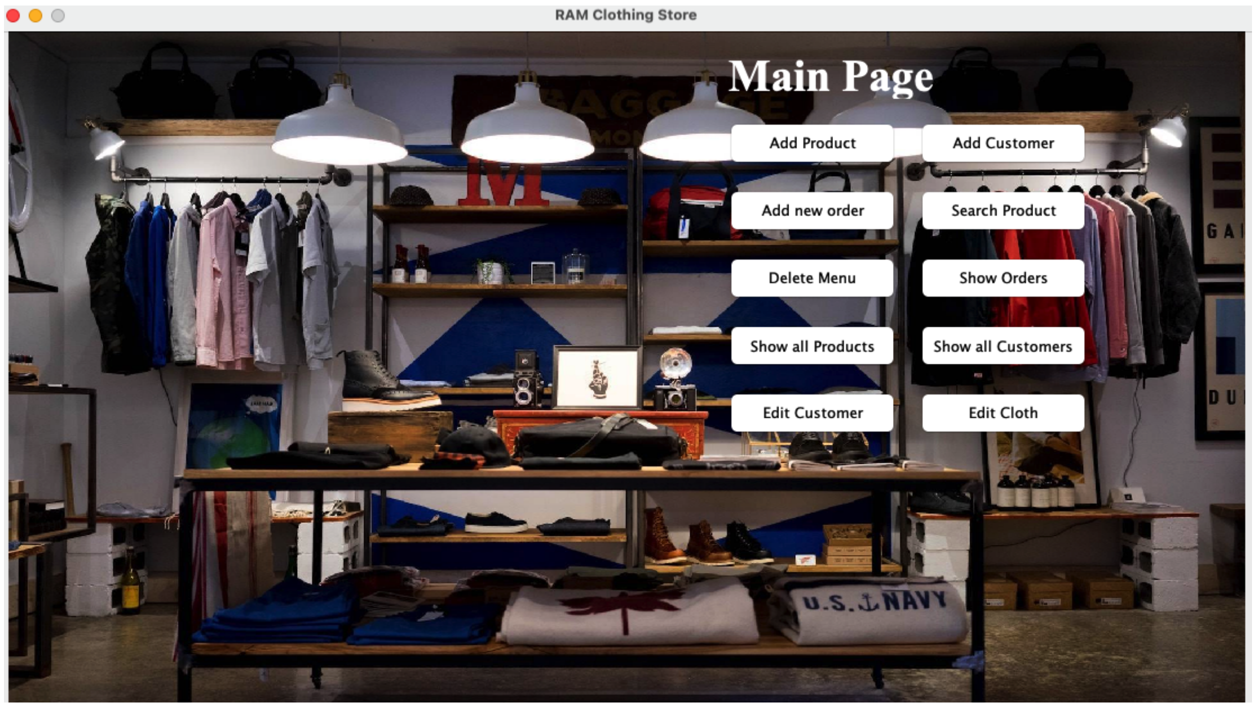
Name	ID
Asif Mahbub	2232119642
Rayed Riasat Rabbi	2311649642
Mezbah Uddin Ahmed	2231194042

Introduction: This is a basic GUI based management system application for a store. In this application, an administrator can add, search, show, and edit their products. He/she can also add and show orders. Additionally, the administrator has the ability to add, show, and edit their customer details.

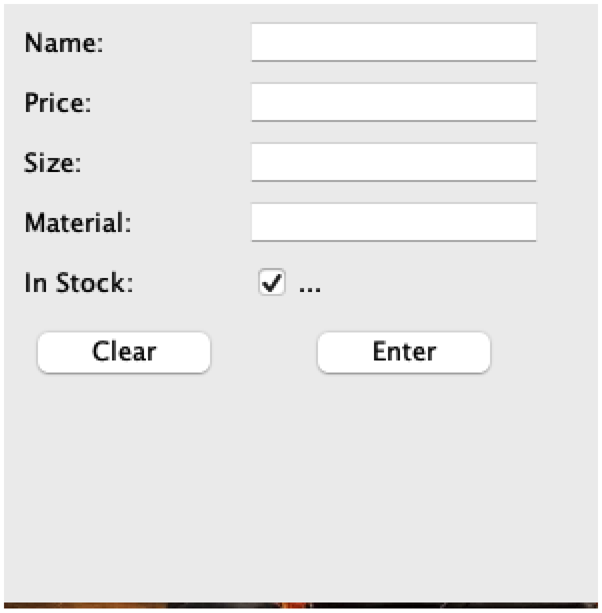
Methodology:

Our full project has an UI, still we are including the output screenshots here for reference

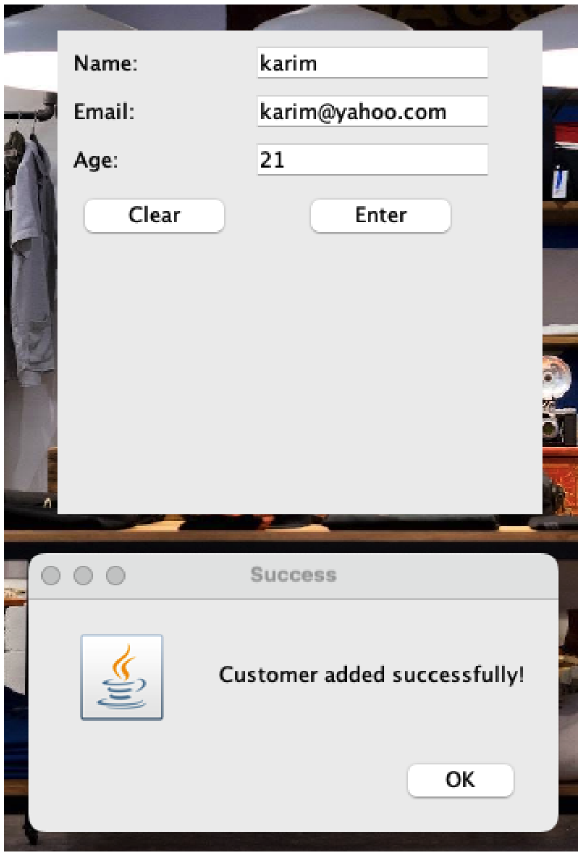
Main Page: This is the first page which the user will see first after they run the program.



Add Product: After clicking on this option the user will be able to add products to the store.

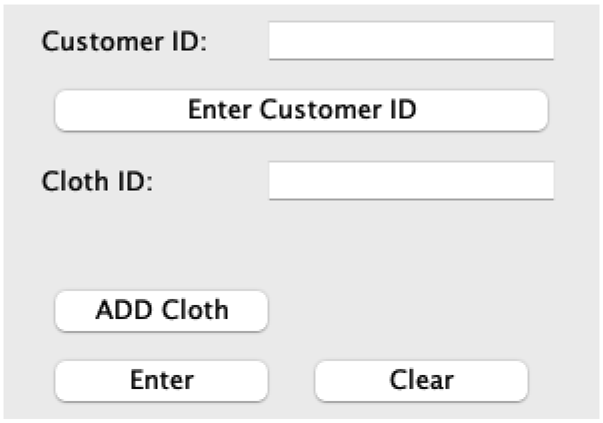


Add Customer: Here the user will be able to add customers after entering their name, email address and age


Name: karim
Email: karim@yahoo.com
Age: 21

Customer added successfully!

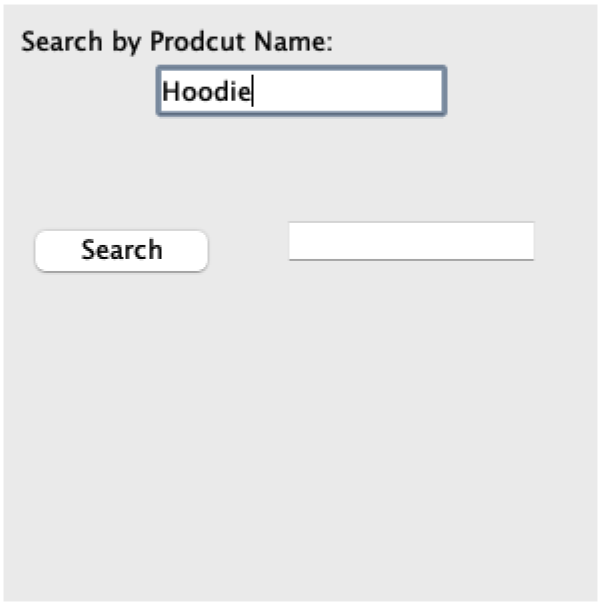
Add new order: This option will take a new order for a customer.


Customer ID:

Cloth ID:

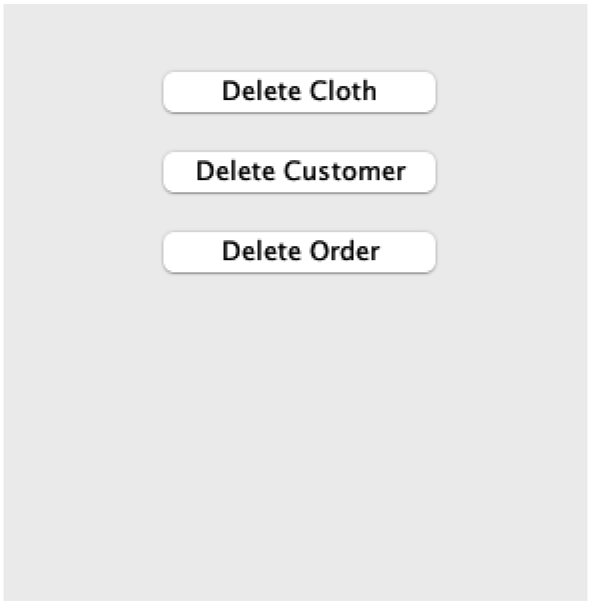
After adding the customer ID and cloth ID it will take the order for the customer

Search Product: We can search a product by its name

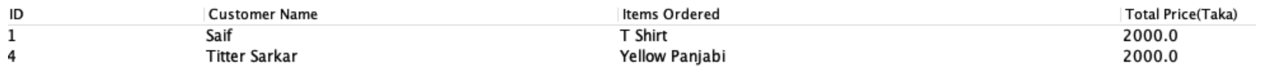


Search by Prodcut Name: Hoodie

Delete Menu: We can delete cloth, customer and order by giving the corresponding ID.

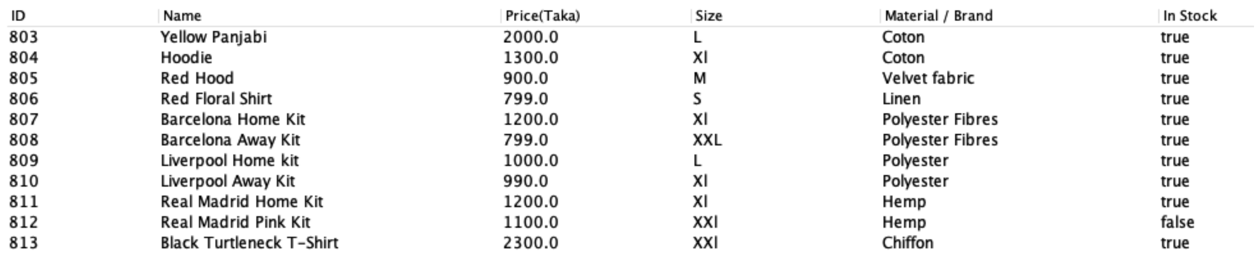


Show order: After taking an order from the customer from the “Add new order” menu. It displays all the orders with a randomly generated order number.



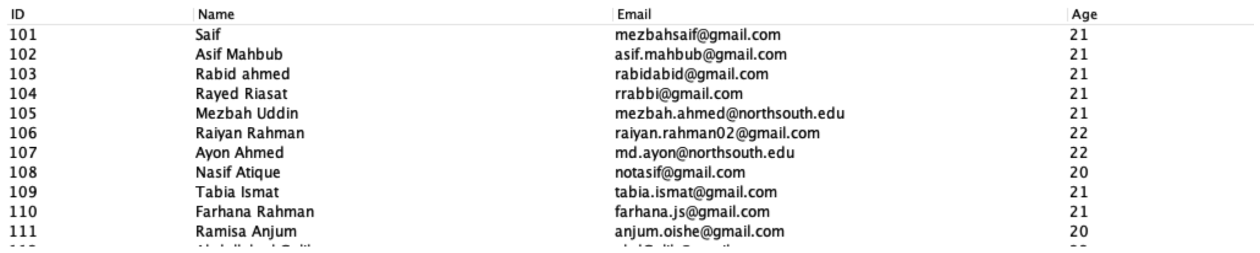
ID	Customer Name	Items Ordered	Total Price(Taka)
1	Saif	T Shirt	2000.0
4	Titter Sarkar	Yellow Panjabi	2000.0

Show all products: This menu shows all the added products into our database. It also generates an ID number and the products are displayed according to the ID.



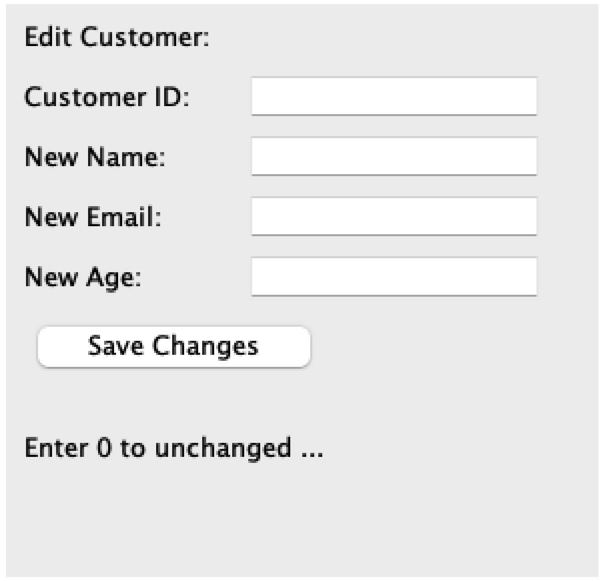
ID	Name	Price(Taka)	Size	Material / Brand	In Stock
803	Yellow Panjabi	2000.0	L	Coton	true
804	Hoodie	1300.0	Xl	Coton	true
805	Red Hood	900.0	M	Velvet fabric	true
806	Red Floral Shirt	799.0	S	Linen	true
807	Barcelona Home Kit	1200.0	Xl	Polyester Fibres	true
808	Barcelona Away Kit	799.0	XXL	Polyester Fibres	true
809	Liverpool Home kit	1000.0	L	Polyester	true
810	Liverpool Away Kit	990.0	Xl	Polyester	true
811	Real Madrid Home Kit	1200.0	Xl	Hemp	true
812	Real Madrid Pink Kit	1100.0	XXl	Hemp	false
813	Black Turtleneck T-Shirt	2300.0	XXl	Chiffon	true

Show all customers: This is similar to the show all products menu. This menu shows all the customer details we have.

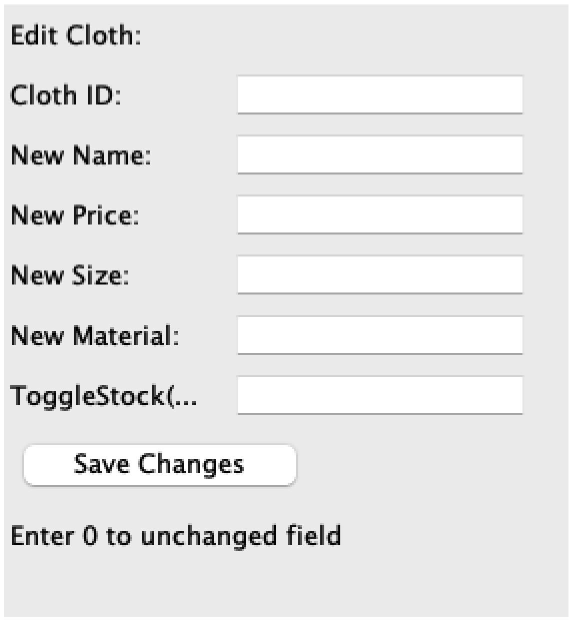


ID	Name	Email	Age
101	Saif	mezbahsaif@gmail.com	21
102	Asif Mahbub	asif.mahbub@gmail.com	21
103	Rabid ahmed	rabidabid@gmail.com	21
104	Rayed Riasat	rrabbi@gmail.com	21
105	Mezbah Uddin	mezbah.ahmed@northsouth.edu	21
106	Raiyan Rahman	raiyan.rahman02@gmail.com	22
107	Ayon Ahmed	md.ayon@northsouth.edu	22
108	Nasif Atique	notasif@gmail.com	20
109	Tabia Ismat	tabia.ismat@gmail.com	21
110	Farhana Rahman	farhana.js@gmail.com	21
111	Ramisa Anjum	anjum.oishe@gmail.com	20

Edit Customer: This menu is used when we need to edit the details of a customer. We can use this menu to edit all the details of a particular customer. However, if we want to keep some fields unchanged, we have to put a 0 in that field to keep that field unchanged. The other field data will be changed after we click “Save Changes”



Edit Cloth: This is exactly similar to the “Edit Customer” menu



Edit Cloth:	
Cloth ID:	
New Name:	
New Price:	
New Size:	
New Material:	
ToggleStock(...	

Enter 0 to unchanged field

<u>Project description</u>: The whole project consists of 4 packages. These are `admin.system`,`com.onlinestore`,`com.onlinestore.exceptions` and `com.onlinestore.model`. Each package contains classes, which are stated below.

`admin.system` package:

It contains the HomePage.java class, which acts as the landing page of this project. From here, the system admin can navigate to all the methods.

<u>UML Diagram:</u>

HomePage
- deleteCustomerButton: JButton - productTable: JTable - ProductScrollPane: JScrollPane - addO rderButton: JButton - orderInputPanel: JPanel - searchButton: JButton - searchPanel: JPanel - deleteCustomerPanel: JPanel - showAllButton: JButton - orderScrollPane: JButton - productTableModel: DefaultTableModel

```
-   deleteClothPanel: JPanel
-   customerInputPanel: JPanel
-   deleteOrderButton: JButton
-   addCustomerButton: JButton
-   showOrderButton: JButton
-   delPanel: JPanel
-   orderTable: JPanel
-   editClothPanel: JPanel
-   clothInputPanel: JPanel
-   customerTable:
-   deleteClothButton: JButton
-   editClothButton: JButton
-   editCustomerPanel: JPanel
-   showCustomerButton: JButton
-    editCustomerButton: JButton
-   customerScrollPane: JScrollPane
-   customerTableModel: DefaultTableModel
-   Order: Order
-   addProductButton: JButton
-   deleteButton: JButton
-   deleteOrderPanel: JPanel
-   orderTableModel: DefaultTableModel
-   store1: OnlineStoreManagementSystem
-----------------------------------------
-   showCustomerTable(): void
-   showOrderTable() : void
+   main(String[]): void
+   actionPerformed(ActionEvent): void
+   showProductTable(): void
    HomePage()
```

`com.onlinestore` package:

Contains OnlineStoreManagementSystem.java class, which acts as a structure of the project. It contains all the essential methods that other classes call.

UML Diagram:

```
            OnlineStoreManagementSystem
-----------------------------------------
-   serialVersionUID: long
-   customers: List <Customer>
-   clothes: List < Cloth >
-   orders: List<Order>
-----------------------------------------
+   addCloth(Cloth): void
+   setCustomers(List<Customer>): void
+   getOrderbyId(int): Order
+   loadDataFromFile(String): OnlineStoreManagementSystem
+   displayClothes (): void
+   editCloth (int, String, double, String, String): void
+   getCustomers(): List<Customer>
+   deleteOrder(int)
+   searchClothersByName(String): List<Cloth>
+   deleteCloth(int): void
+   addCustomer(String, String, int): void
+   addOrder(Order): void
+   createOrder(Customer): Order
+   getOrders(): List <Order>
+   setOrders(List<Order>): void
+   getCustomerById(int): Customer
+   displayCustomers): void
+   saveDataToFile(String): void
+   getClothes (): List <Cloth>
+   deleteCustomer(int): void
+   addCustomer(Customer): void
+   searchCustomerByName(String) List<Customer>
+   getClothById(int): Cloth
+   editCustomer(int, String, String, int): void
+   setClothes(List<Cloth): void
+   addClothToOrder(int, Cloth): void
+   displayOrders(): void
```

`com.onlinestore.exceptions` package:

Contains all the custom exceptions

UML Diagram:

```
CustomerNotFoundException extends
Exception
---------------------------------
+   CustomerNotFoundException(String)
```

```
OrderNotFoundException extends Exception
---------------------------------
+   OrderNotFoundException(String)
```

```
ProductNotFoundException extends Exception
---------------------------------
+   ProductNotFoundException(String)
```

`com.onlinestore.model` package:

UML Diagram:

<<Interface>>
ProductInterface

+ getProductId(): int
+ getName(): String
+ getPrice(): double

<<Interface>>
StockAvailability

+ isInStock: boolean
+ *setInStock(boolean): void*

Product

- inStock: boolean
- serialVersionUID: long
- name: String
- price: double
- nextProductId: int
- productId: int

/* Constructor */
/* accessor-mutator */
+ isInStock(): boolean
- generateProductId(): int

Wearable extends Product

+ Wearable(String, double, String, String)

- Size: String
- serialVersionUID: long
- material: String

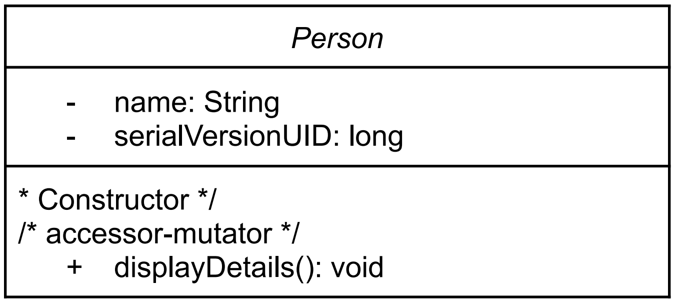
/* Constructor */
/* accessor-mutator */

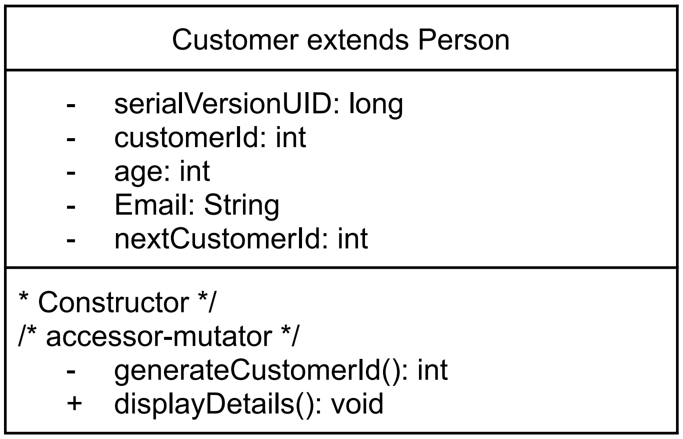
Cloth extends Wearable

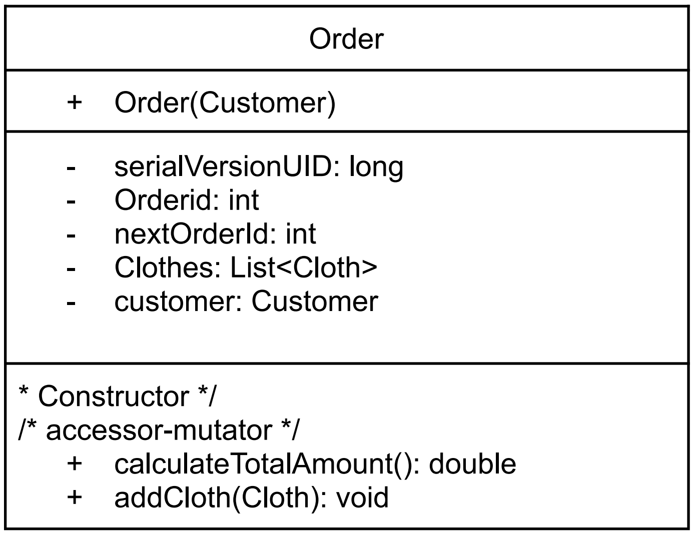
+ Cloth(String, double, String, String, boolean)

- serialVersionUID: long
- inStock: boolean

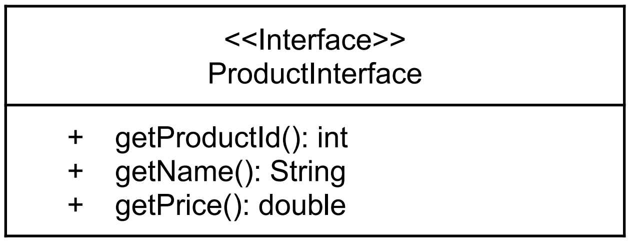
/* Constructor */
/* accessor-mutator */
+ isInStock: boolean

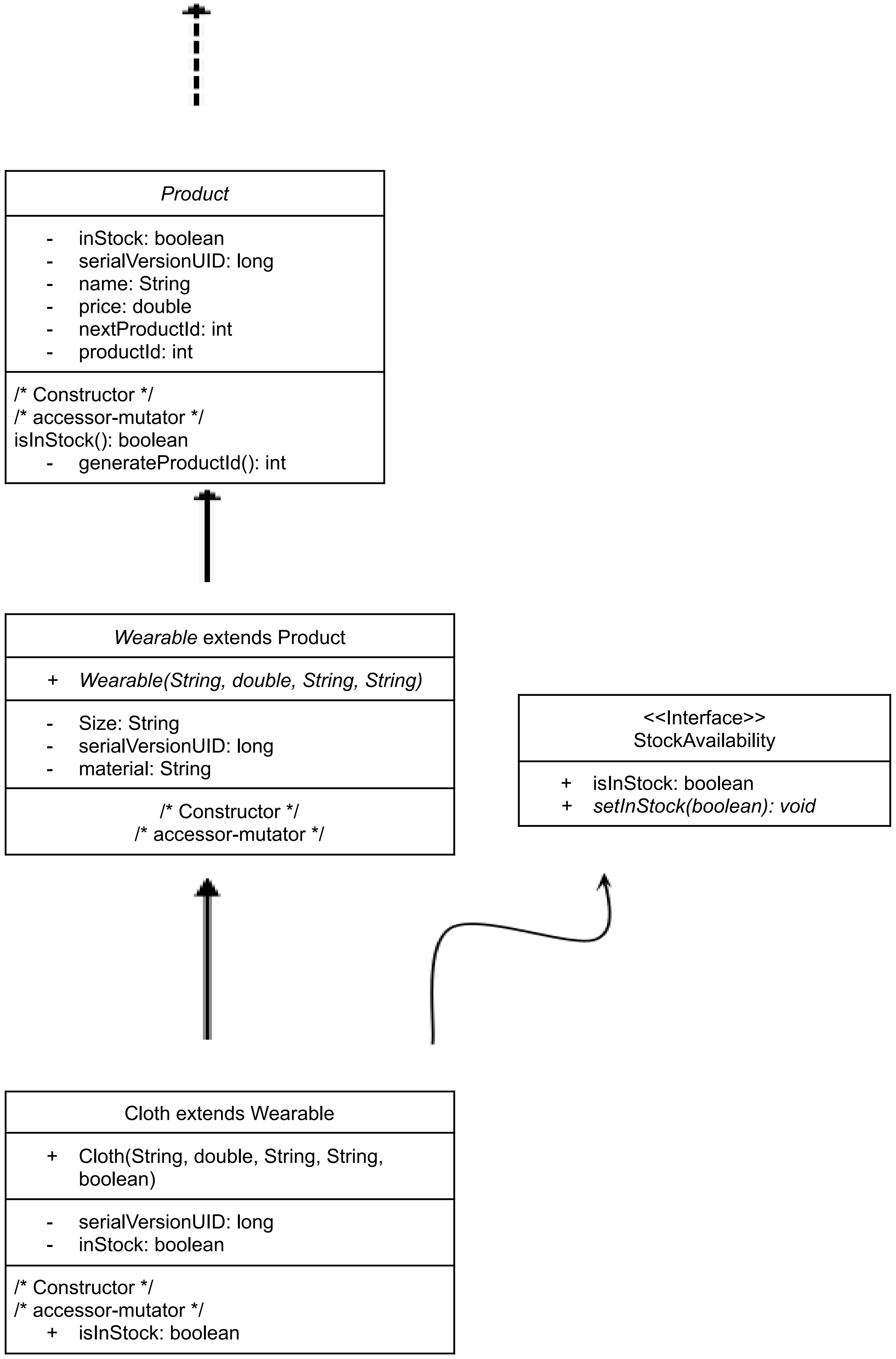






UML diagram in whole:




Product
- inStock: boolean
- serialVersionUID: long
- name: String
- price: double
- nextProductId: int
- productId: int
/* Constructor */
/* accessor-mutator */
isInStock(): boolean
- generateProductId(): int
Wearable extends Product
+ Wearable(String, double, String, String)
- Size: String
- serialVersionUID: long
- material: String
/* Constructor */
/* accessor-mutator */
<<Interface>>
StockAvailability
+ isInStock: boolean
+ setInStock(boolean): void
Cloth extends Wearable
+ Cloth(String, double, String, String, boolean)
- serialVersionUID: long
- inStock: boolean
/* Constructor */
/* accessor-mutator */
+ isInStock: boolean

Order

+ Order(Customer)

- serialVersionUID: long
- Orderid: int
- nextOrderId: int
- Clothes: List<Cloth>
- customer: Customer

* Constructor */
/* accessor-mutator */
+ calculateTotalAmount(): double
+ addCloth(Cloth): void

CustomerNotFoundException extends Exception

+ CustomerNotFoundException(String)

OrderNotFoundException extends Exception

+ OrderNotFoundException(String)

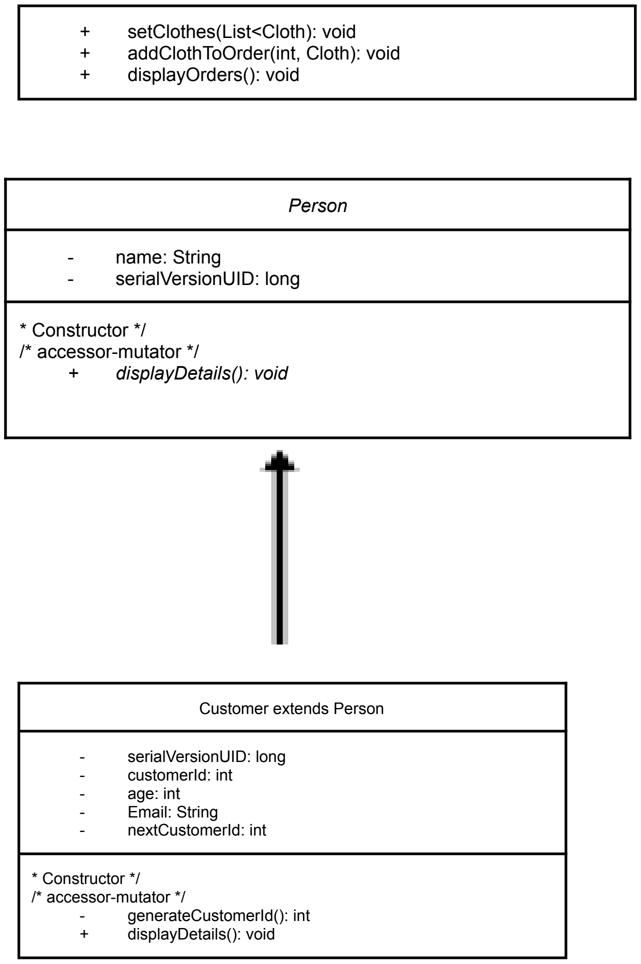
ProductNotFoundException extends Exception

+ ProductNotFoundException(String)

OnlineStoreManagementSystem

- serialVersionUID: long
- customers: List <Customer>
- clothes: List < Cloth >
- orders: List<Order>

+ addCloth(Cloth): void
+ setCustomers(List<Customer>): void
+ getOrderbyId(int): Order
+ loadDataFromFile(String): OnlineStoreManagementSystem
+ displayClothes (): void
+ editCloth (int, String, double, String, String): void
+ getCustomers(): List<Customer>
+ deleteOrder(int)
+ searchClothersByName(String): List<Cloth>
+ deleteCloth(int): void
+ addCustomer(String, String, int): void
+ addOrder(Order): void
+ createOrder(Customer): Order
+ getOrders(): List <Order>
+ setOrders(List<Order>): void
+ getCustomerById(int): Customer
+ displayCustomers): void
+ saveDataToFile(String): void
+ getClothes (): List <Cloth>
+ deleteCustomer(int): void
+ addCustomer(Customer): void
+ searchCustomerByName(String) List<Customer>
+ getClothById(int): Cloth
+ editCustomer(int, String, String, int): void


+ setClothes(List<Cloth>): void
+ addClothToOrder(int, Cloth): void
+ displayOrders(): void
Person
- name: String
- serialVersionUID: long
* Constructor */
/* accessor-mutator */
+ *displayDetails(): void*
Customer extends Person
- serialVersionUID: long
- customerId: int
- age: int
- Email: String
- nextCustomerId: int
* Constructor */
/* accessor-mutator */
- generateCustomerId(): int
+ displayDetails(): void