Birzeit University
Faculty of Engineering and Technology
Computer Science Department
SWEN6304: Software Design and Architecture
Project #1

Student Name: Duha Jarrar
Student ID: 1225272

Date: 25 Nov 2023

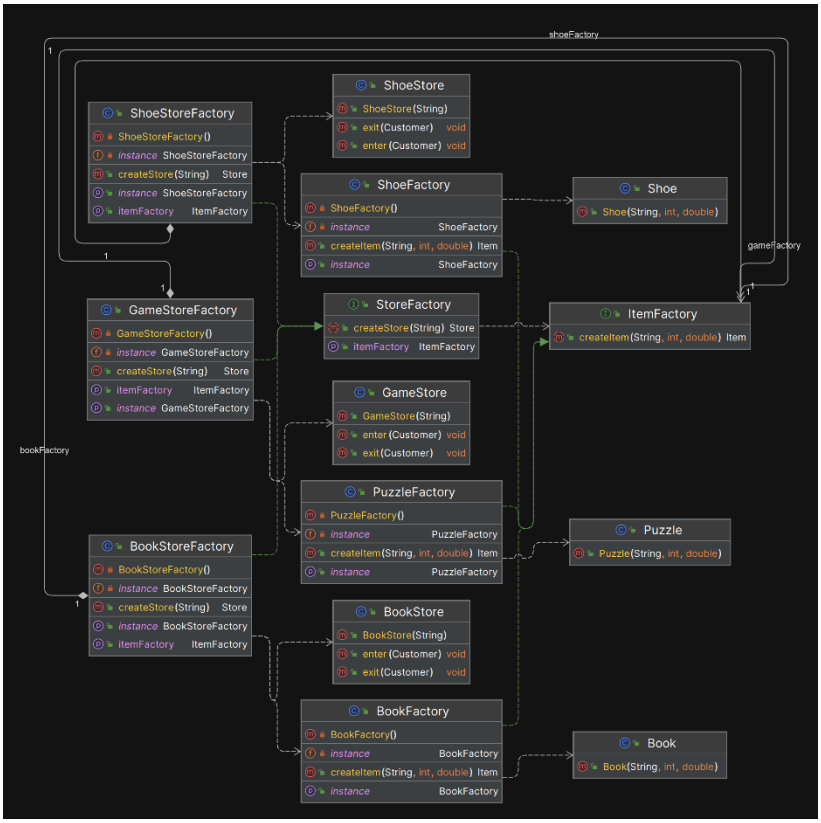
❖ Online Shopping Application Patterns

Source Code: https://github.com/duhajarrar/shop , you can find the UML images on GitHub too.

The following UML diagram shows the whole system we created.

- **Factory Method Pattern:**

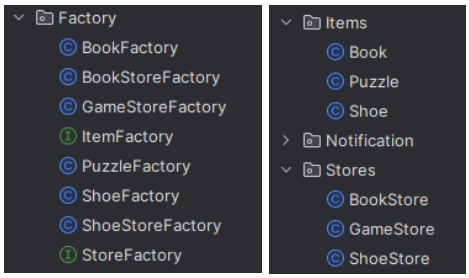
  The pattern used in creating stores and items, each store factory has its own item factory to create its products, the following image shows the UML diagram of how the pattern is used:



The following image shows how to use the pattern in creating items and stores:

```
StoreFactory bookStoreFactory = BookStoreFactory.getInstance();
Store bookStore = bookStoreFactory.createStore("Book Store");
Item book = bookStoreFactory.getItemFactory().createItem("Design Book 1",bookStore.getStoreId(),100);
```

The following image shows the classes created to apply the pattern:

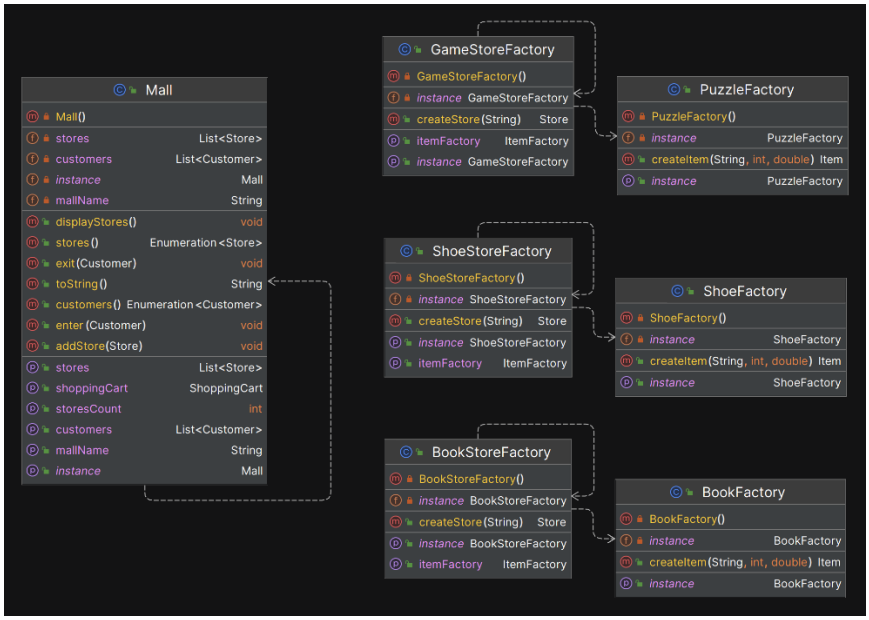


- **Singleton Pattern:**

  The pattern used in the mall and the previous factories, the following image shows how we use it:

```
Mall mall = Mall.getInstance();
StoreFactory gameStoreFactory = GameStoreFactory.getInstance();
StoreFactory shoeStoreFactory = ShoeStoreFactory.getInstance();
```

  The following image shows the UML diagram of classes that used singleton:



- **Chain of Responsibility and Decorator Patterns:**

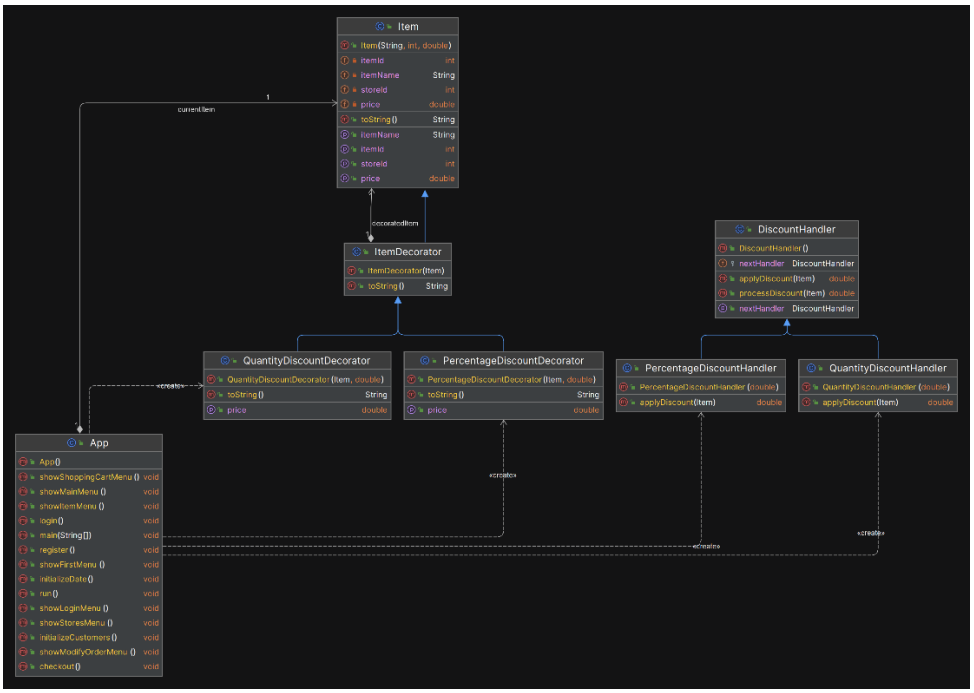
  Those patterns are used for adding discounts for the items. The *decorator pattern* is used to add one discount to the items, and the following image shows how to do that using it:

```
Item sportBoot1 = shoeStoreFactory.getItemFactory().createItem( itemName: "Nike boot",shoeStore.getStoreId(), price: 200);
Item discountedSportBoot = new QuantityDiscountDecorator(sportBoot1, discountQuantity: 100);
```

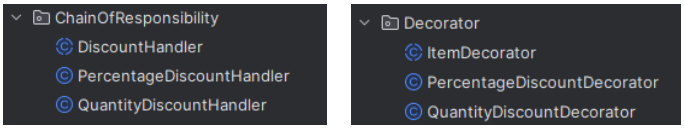
  The *chain of responsibility pattern* is used to add a sequence of discounts as shown in the following:

```
DiscountHandler quantDiscountHandler = new QuantityDiscountHandler( discountQuantity: 10);
quantDiscountHandler.setNextHandler(new PercentageDiscountHandler( discountPercentage: 20));
book.setPrice(quantDiscountHandler.applyDiscount(book));
```

  The following image shows the UML class diagram of the patterns classes:

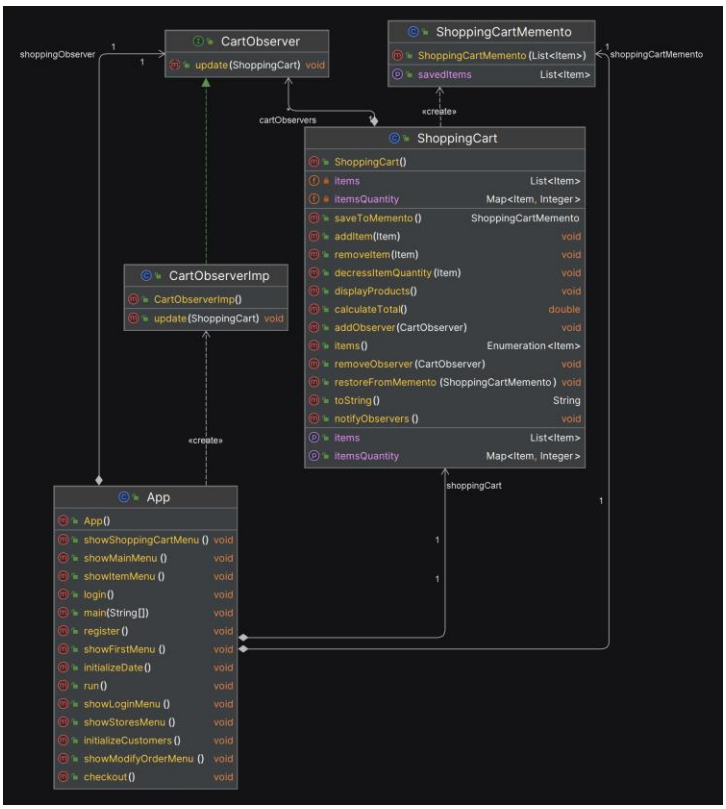


  The code of those patterns shown in the classes shown in the following image:



- **Memento and Observer Patterns:**

  Memento is used to save and restore the cart, and the observer is used to notify any changes that happen to the cart. The image below shows the UML diagram of the patterns:



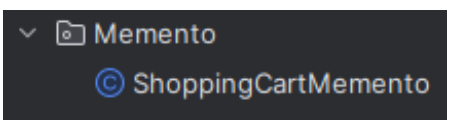
  To use memento pattern we do the following:

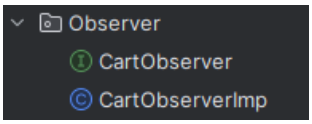
```
ShoppingCartMemento shoppingCartMemento = shoppingCart.saveToMemento();
```

  To use cart observer we do the following:

```
CartObserver shoppingObserver = new CartObserverImp();
shoppingCart.addObserver(shoppingObserver);
```

  The code of the patterns is found in the following file:



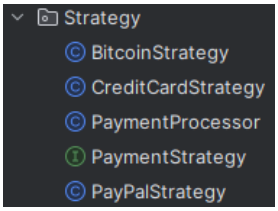


- **Strategy Pattern:**

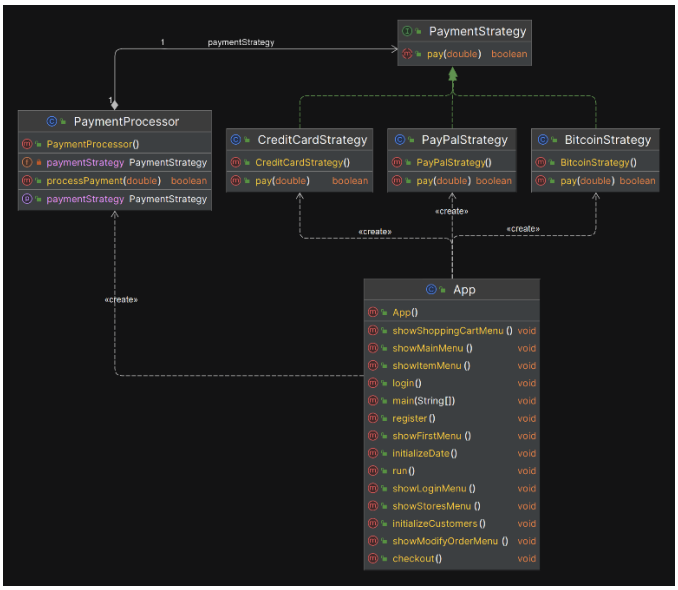
It used to pay with different ways of payment. To use it we do the following:

```
PaymentProcessor paymentProcessor = new PaymentProcessor(
paymentProcessor.setPaymentStrategy(new CreditCardStrategy());
paymentProcessor.processPayment(totalAmount);
```

The code files of the pattern:

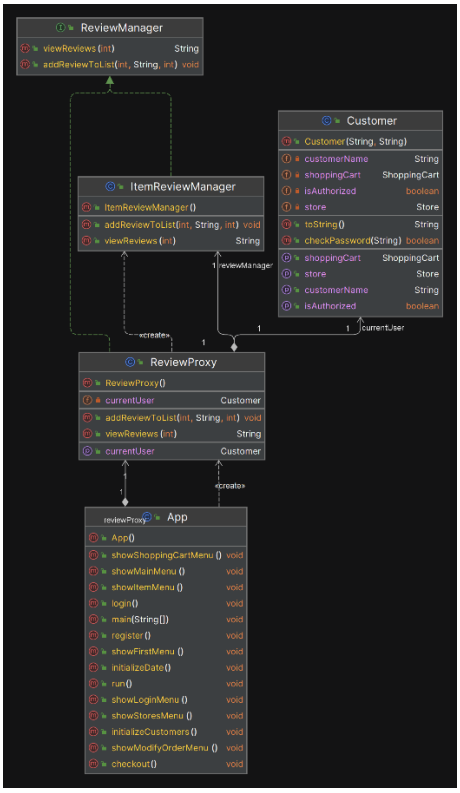


The image below shows the UML diagram of the patterns:



- **Proxy Pattern:**

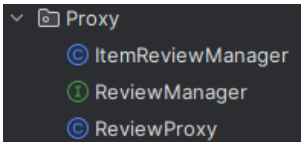
  The pattern used to prevent any user who is not logged in from adding reviews and rating the products and viewing reviews, the following image shows the UML diagram of the classes used to apply it:



  To use it we logged in using the AuthManager and then did the following:

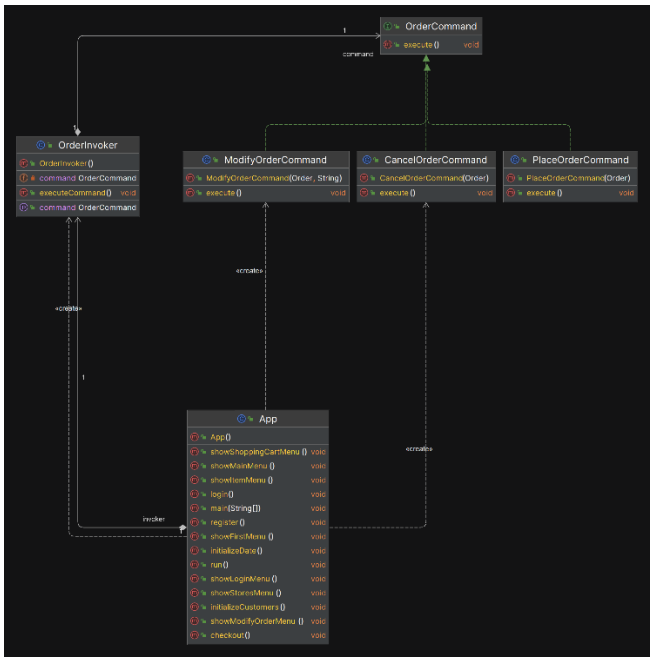
```
ReviewProxy reviewProxy = new ReviewProxy();
reviewProxy.setCurrentUser(currentUser);
reviewProxy.addReviewToList(currentItem.getItemId(),review_message, rate);
reviewProxy.viewReviews(currentItem.getItemId());
```

  You can find the code files of it in the following:



- **Command Pattern:**

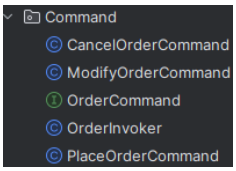
  The pattern used to add actions in the order, as place, modify, and cancel, the following is a UML diagram of it:



  The following is how we use the pattern:

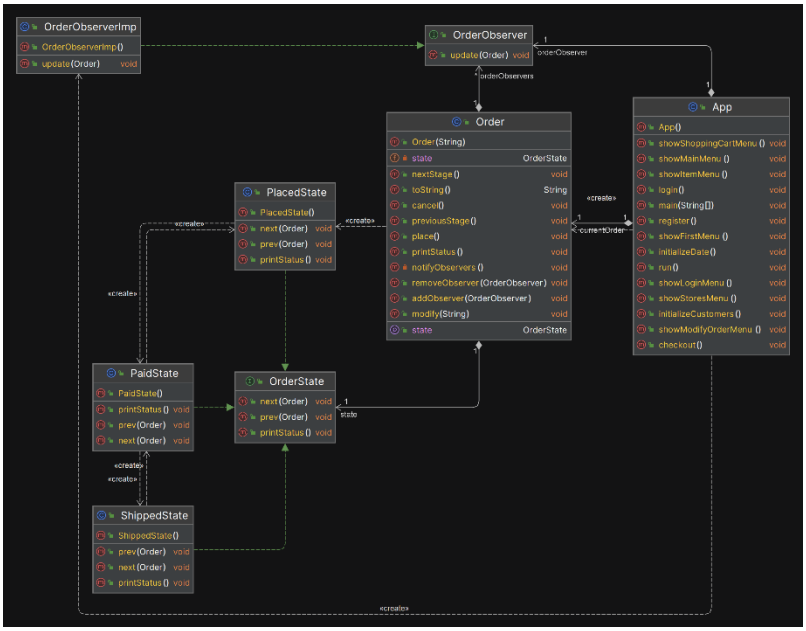
```
OrderCommand modifyOrderCommand = new ModifyOrderCommand(currentOrder, shoppingCart.toString());
invoker.setCommand(modifyOrderCommand);
invoker.executeCommand();
```

  You can find the code files of it in the following:



- **State Pattern:**

Its used to describe the order state and next state, the following is a UML diagram of it:

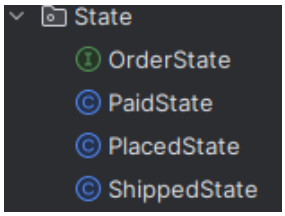


The following is how we use the pattern:

```
public Order(String orderDetails) {
    this.orderDetails = orderDetails;
    this.state = new PlacedState();
```

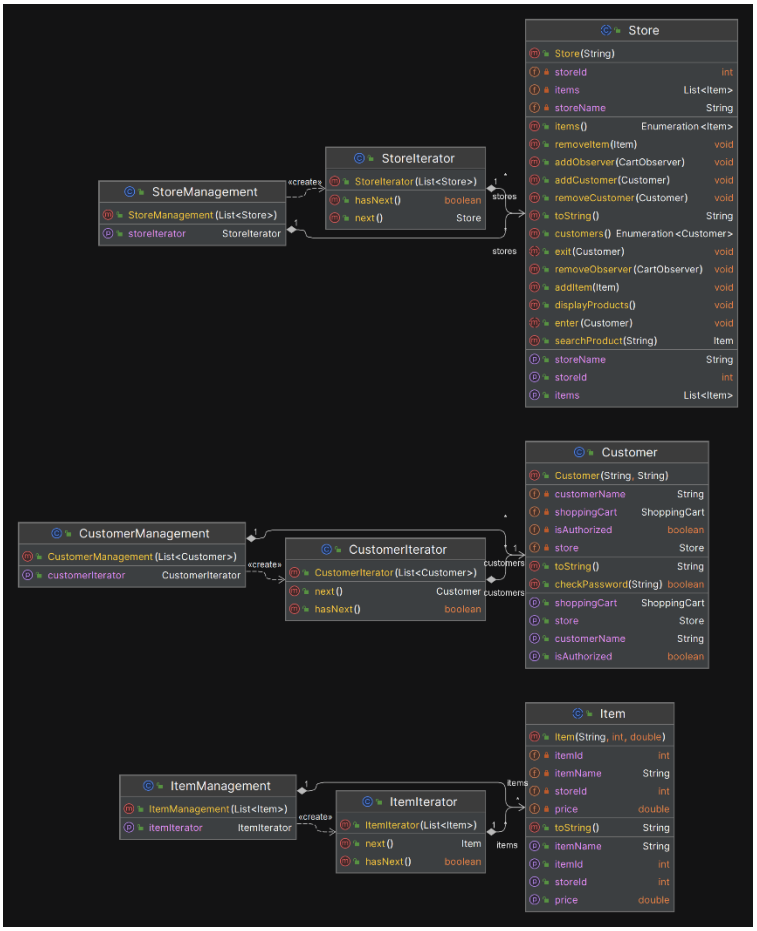
```
currentOrder.printStatus();
currentOrder.nextStage();
```

You can find the code files of it in the following:



- **Iterator Pattern:**

  It's used to iterate in list of things one by one, the following is a UML diagram of it:



The following is how we use the pattern:

```
ItemManagement itemManagement = new ItemManagement(items);
ItemIterator iterator = itemManagement.getItemIterator();
while(iterator.hasNext()){
    System.out.println(iterator.next());
}
```

You can find the code files of it in the following: