RMIT UNIVERSITY

RMIT International University Vietnam

Individual Project - Object-Oriented Programming Shopping System Design Report

Subject Code:	**COSC2440**
Subject Name:	**Further Programming**
Location & Campus	**SGS Campus**
Student name:	**Nguyen Quoc An**
Student Number:	**s3938278**
Lecturer’s name:	**Dr. Dang Tran Tri**

I. Introduction

The purpose of this report is to present, analyze and evaluate the system design for the Online Shopping Service. This will include the demonstration, breakdown, and analysis of the system's class, as well as evaluation of the software according to the OOP design principles and the industry's Software Quality Standard.

II. Functionality

The following deliverables have been achieved for the system:

- Users can browse, search, view, and purchase products.
- Each product contains a unique name, description, quantity available, and price.
- There are 2 types of products, digital and physical, with physical products having an additional weight
- Some products can be giftable and will contain a greeting message
- Users can change or view product information
- Users can create shopping carts and add or remove products to shopping carts
- Users can view all products in all shopping carts
- Shopping carts contain a set of product names with a quantity of one.
- Users can calculate the total price and total weight of the shopping carts.
- Users can sort shopping carts based on their total weight in ascending or descending order.
- Users can manage products and shopping carts through a text-based UI.

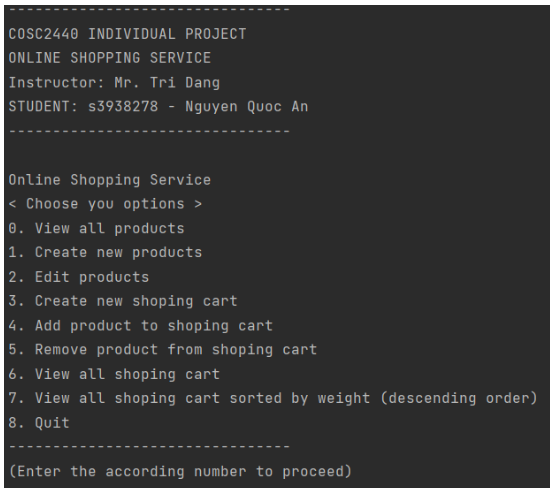
III. Assumption Made

- User input is correct and corresponds to the requirements of the program
- Products have a unique name
- Shopping carts have unique names generated automatically based on their cart count
- A shopping cart can only contain 1 product of a type

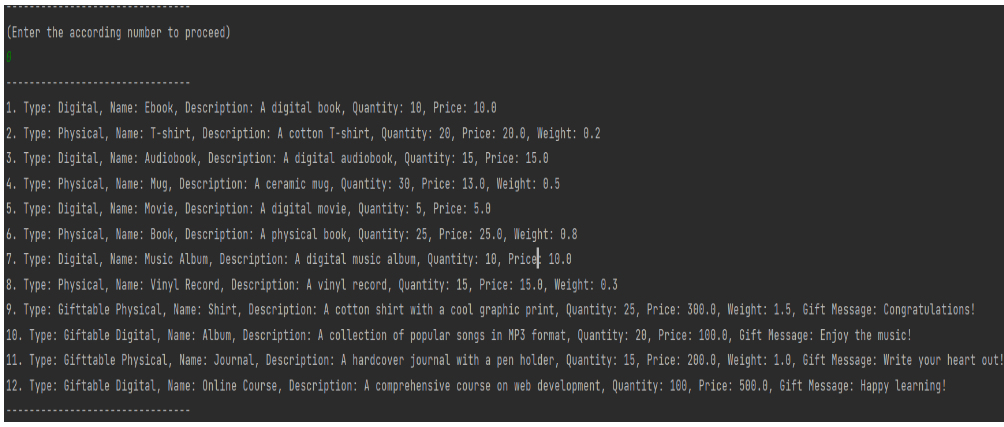
IV. Program Demonstration

The users will interact with the shopping system through a simple text-based interface, using numerical and character inputs.

1. Users can view all products contained in the product list by pressing "0"

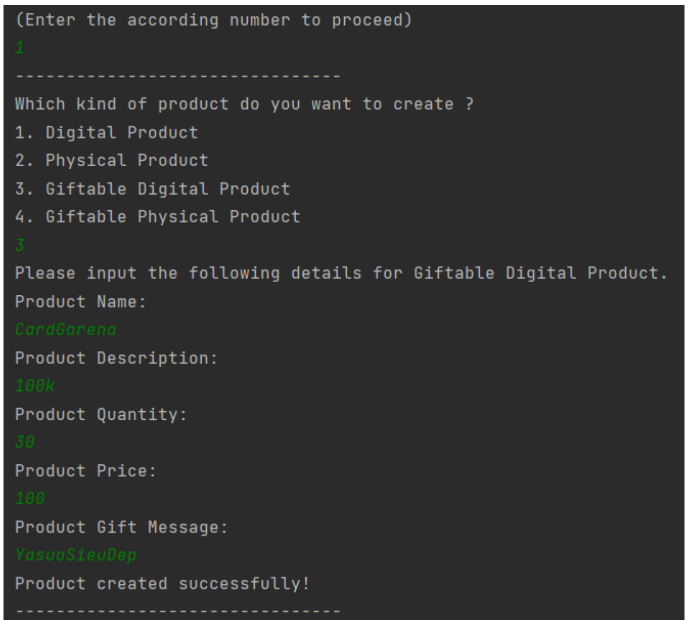


```
--------------------------------
COSC2440 INDIVIDUAL PROJECT
ONLINE SHOPPING SERVICE
Instructor: Mr. Tri Dang
STUDENT: s3938278 - Nguyen Quoc An
--------------------------------

Online Shopping Service
< Choose you options >
0. View all products
1. Create new products
2. Edit products
3. Create new shoping cart
4. Add product to shoping cart
5. Remove product from shoping cart
6. View all shoping cart
7. View all shoping cart sorted by weight (descending order)
8. Quit
--------------------------------
(Enter the according number to proceed)
```

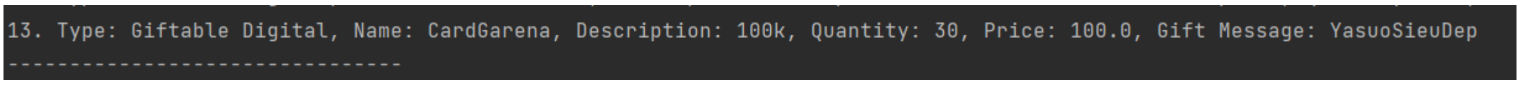


```
--------------------------------
(Enter the according number to proceed)
0
--------------------------------
1. Type: Digital, Name: Ebook, Description: A digital book, Quantity: 10, Price: 10.0
2. Type: Physical, Name: T-shirt, Description: A cotton T-shirt, Quantity: 20, Price: 20.0, Weight: 0.2
3. Type: Digital, Name: Audiobook, Description: A digital audiobook, Quantity: 15, Price: 15.0
4. Type: Physical, Name: Mug, Description: A ceramic mug, Quantity: 30, Price: 13.0, Weight: 0.5
5. Type: Digital, Name: Movie, Description: A digital movie, Quantity: 5, Price: 5.0
6. Type: Physical, Name: Book, Description: A physical book, Quantity: 25, Price: 25.0, Weight: 0.8
7. Type: Digital, Name: Music Album, Description: A digital music album, Quantity: 10, Price: 10.0
8. Type: Physical, Name: Vinyl Record, Description: A vinyl record, Quantity: 15, Price: 15.0, Weight: 0.3
9. Type: Gifttable Physical, Name: Shirt, Description: A cotton shirt with a cool graphic print, Quantity: 25, Price: 300.0, Weight: 1.5, Gift Message: Congratulations!
10. Type: Giftable Digital, Name: Album, Description: A collection of popular songs in MP3 format, Quantity: 20, Price: 100.0, Gift Message: Enjoy the music!
11. Type: Gifttable Physical, Name: Journal, Description: A hardcover journal with a pen holder, Quantity: 15, Price: 200.0, Weight: 1.0, Gift Message: Write your heart out!
12. Type: Giftable Digital, Name: Online Course, Description: A comprehensive course on web development, Quantity: 100, Price: 500.0, Gift Message: Happy learning!
--------------------------------
```

2. Users can create new products in the product list by pressing “1”

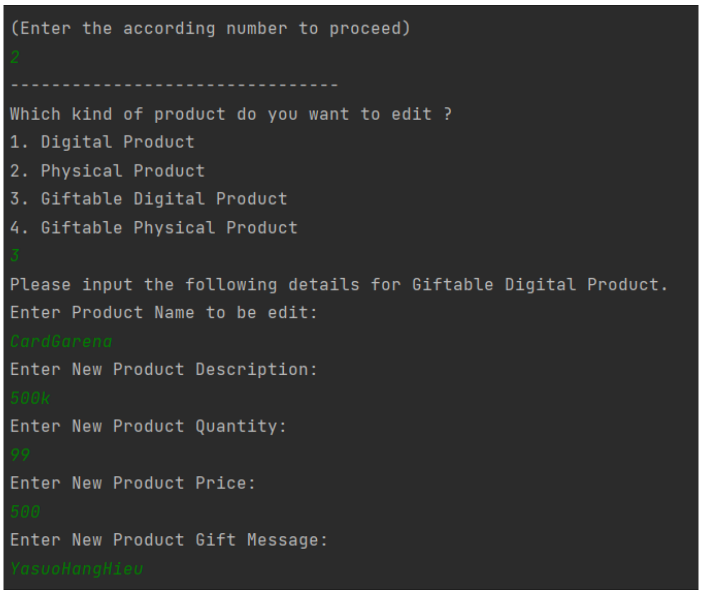


```
(Enter the according number to proceed)
1
-------------------------------
Which kind of product do you want to create ?
1. Digital Product
2. Physical Product
3. Giftable Digital Product
4. Giftable Physical Product
3
Please input the following details for Giftable Digital Product.
Product Name:
CardGarena
Product Description:
100k
Product Quantity:
30
Product Price:
100
Product Gift Message:
YasuoSieuDep
Product created successfully!
-------------------------------
```

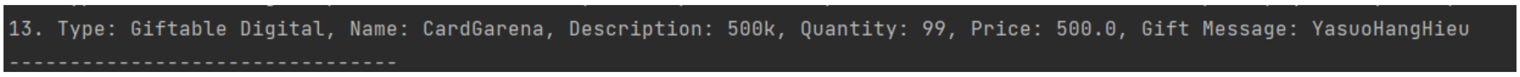


```
13. Type: Giftable Digital, Name: CardGarena, Description: 100k, Quantity: 30, Price: 100.0, Gift Message: YasuoSieuDep
-------------------------------
```

3. Users can edit the product information by pressing “2”



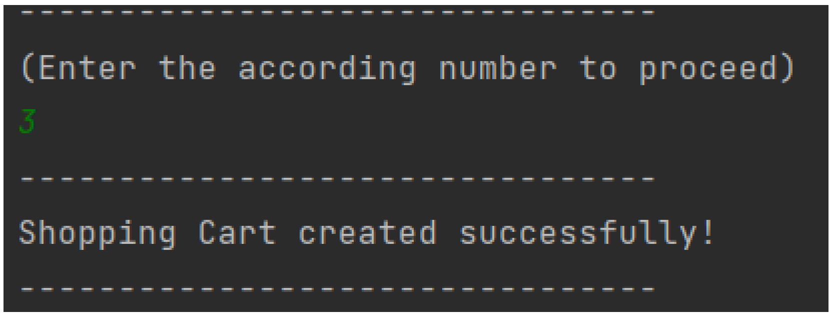
```
(Enter the according number to proceed)
2
-------------------------------
Which kind of product do you want to edit ?
1. Digital Product
2. Physical Product
3. Giftable Digital Product
4. Giftable Physical Product
3
Please input the following details for Giftable Digital Product.
Enter Product Name to be edit:
CardGarena
Enter New Product Description:
500k
Enter New Product Quantity:
99
Enter New Product Price:
500
Enter New Product Gift Message:
YasuoHangHieu
```



```
13. Type: Giftable Digital, Name: CardGarena, Description: 500k, Quantity: 99, Price: 500.0, Gift Message: YasuoHangHieu
-------------------------------
```

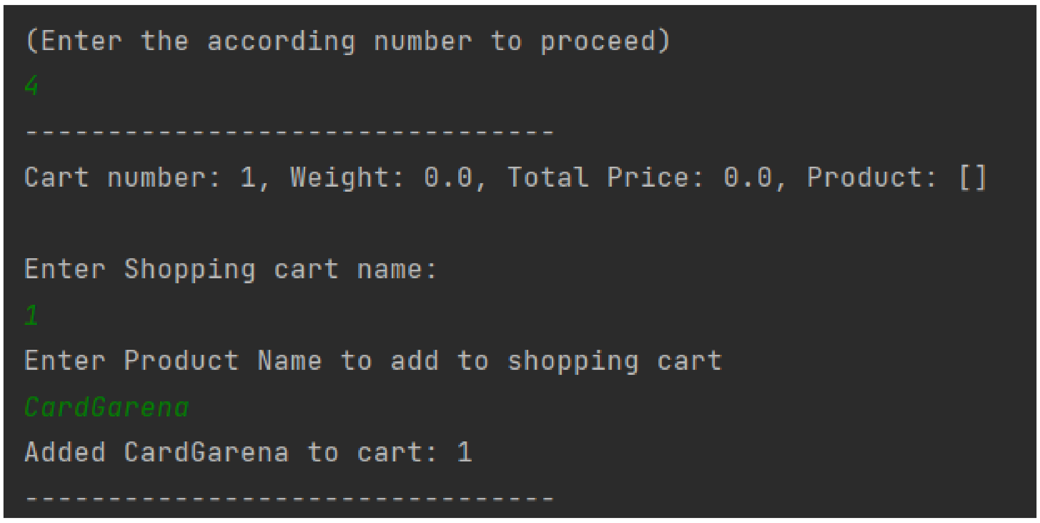
4. Users can create new shopping carts by pressing “3”

The name of the shopping cart will be created automatically based on the cart count number



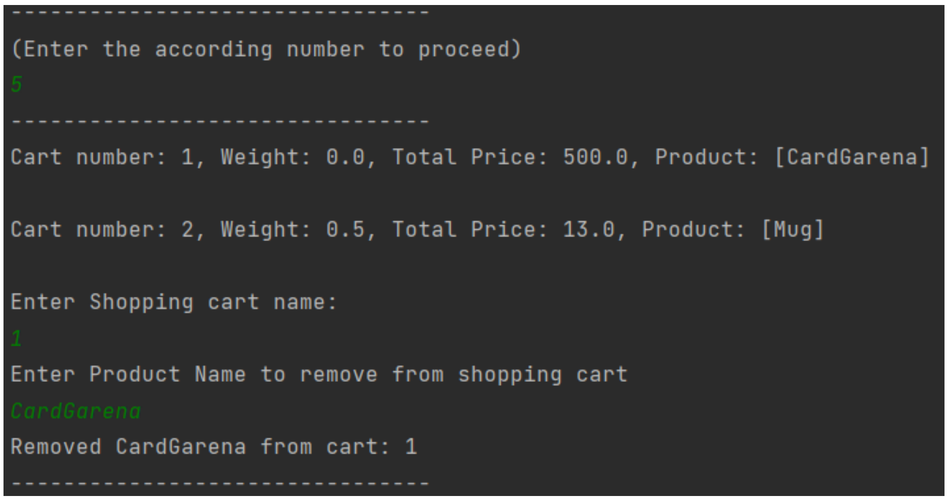
```
-------------------------------
(Enter the according number to proceed)
3
-------------------------------
Shopping Cart created successfully!
-------------------------------
```

5. Users can add a product to a chosen cart by pressing "4"



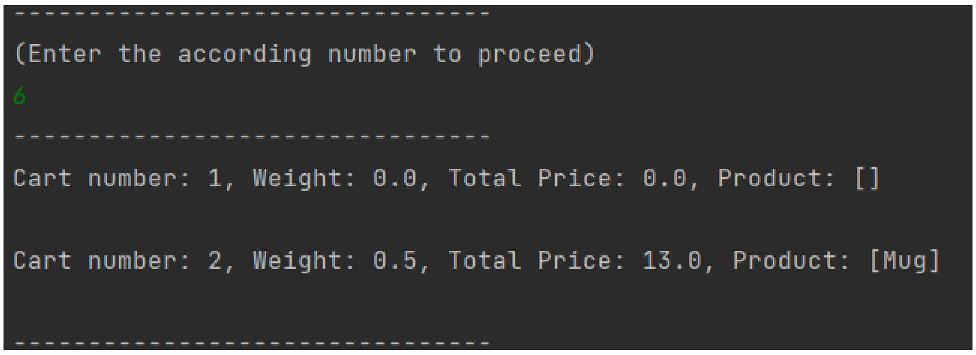
```
(Enter the according number to proceed)
4
--------------------------------
Cart number: 1, Weight: 0.0, Total Price: 0.0, Product: []

Enter Shopping cart name:
1
Enter Product Name to add to shopping cart
CardGarena
Added CardGarena to cart: 1
--------------------------------
```

6. Users can remove a product from a chosen cart by pressing "5"



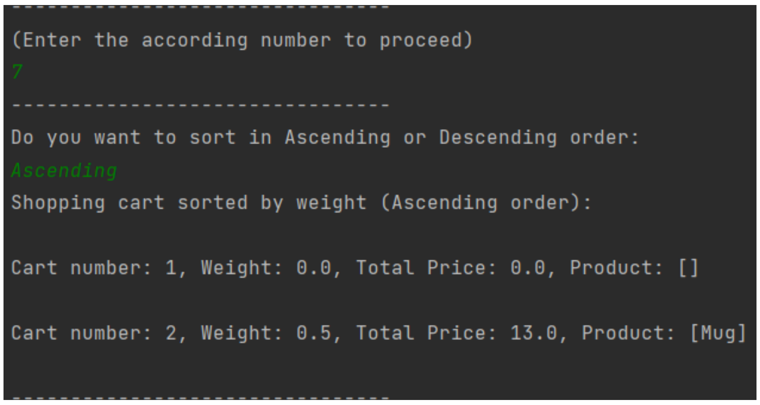
```
--------------------------------
(Enter the according number to proceed)
5
--------------------------------
Cart number: 1, Weight: 0.0, Total Price: 500.0, Product: [CardGarena]

Cart number: 2, Weight: 0.5, Total Price: 13.0, Product: [Mug]

Enter Shopping cart name:
1
Enter Product Name to remove from shopping cart
CardGarena
Removed CardGarena from cart: 1
--------------------------------
```

7. Users can view all shopping carts based on their creation order by pressing "6"

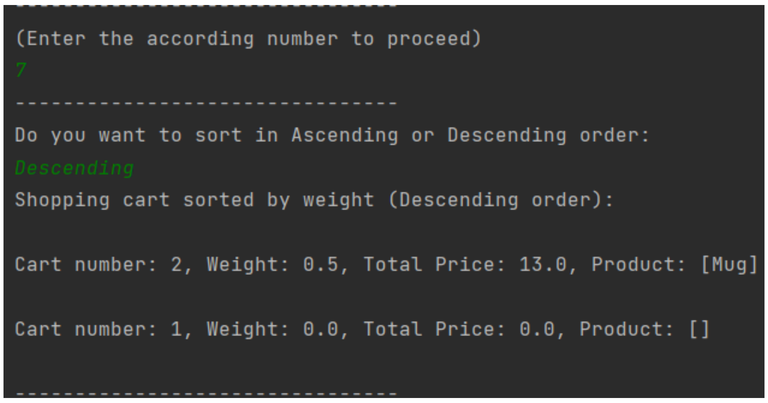


```
--------------------------------
(Enter the according number to proceed)
6
--------------------------------
Cart number: 1, Weight: 0.0, Total Price: 0.0, Product: []

Cart number: 2, Weight: 0.5, Total Price: 13.0, Product: [Mug]

--------------------------------
```

8. Users can view all shopping carts sorted by their total weight in ascending or descending order by pressing "7"

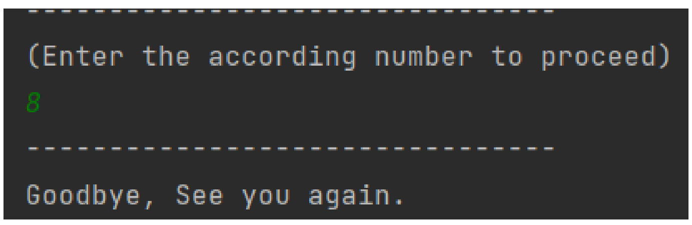


```
--------------------------------
(Enter the according number to proceed)
7
--------------------------------
Do you want to sort in Ascending or Descending order:
Ascending
Shopping cart sorted by weight (Ascending order):

Cart number: 1, Weight: 0.0, Total Price: 0.0, Product: []

Cart number: 2, Weight: 0.5, Total Price: 13.0, Product: [Mug]

--------------------------------
```



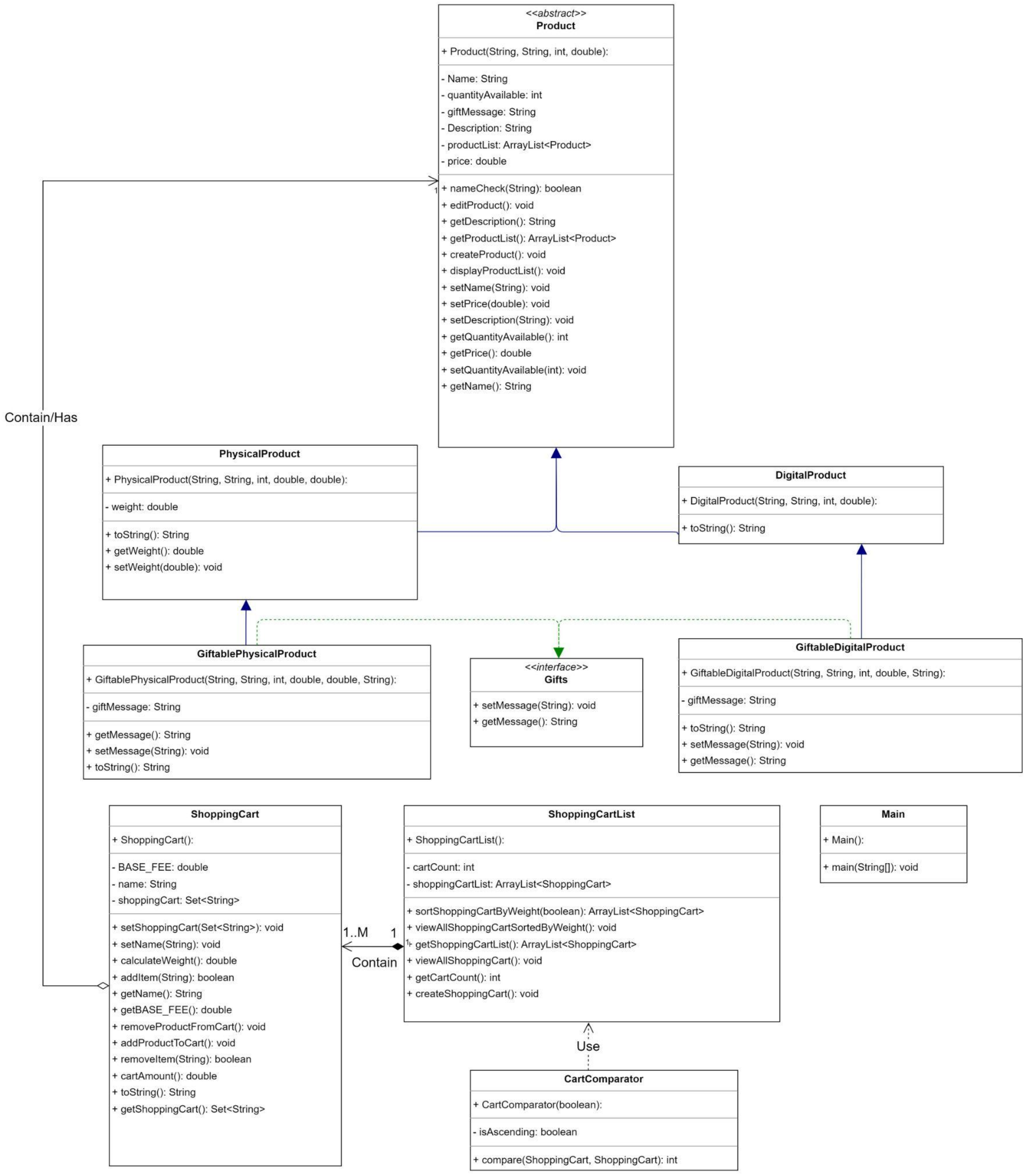
```
--------------------------------
(Enter the according number to proceed)
7
--------------------------------
Do you want to sort in Ascending or Descending order:
Descending
Shopping cart sorted by weight (Descending order):

Cart number: 2, Weight: 0.5, Total Price: 13.0, Product: [Mug]

Cart number: 1, Weight: 0.0, Total Price: 0.0, Product: []

--------------------------------
```

9. Users can quit the program by pressing “8”



```
--------------------------------
(Enter the according number to proceed)
8
--------------------------------
Goodbye, See you again.
```

V. UML Diagram and Class Breakdown



<<abstract>>
Product
+ Product(String, String, int, double):
- Name: String
- quantityAvailable: int
- giftMessage: String
- Description: String
- productList: ArrayList<Product>
- price: double
+ nameCheck(String): boolean
+ editProduct(): void
+ getDescription(): String
+ getProductList(): ArrayList<Product>
+ createProduct(): void
+ displayProductList(): void
+ setName(String): void
+ setPrice(double): void
+ setDescription(String): void
+ getQuantityAvailable(): int
+ getPrice(): double
+ setQuantityAvailable(int): void
+ getName(): String
1
Contain/Has
PhysicalProduct
+ PhysicalProduct(String, String, int, double, double):
- weight: double
+ toString(): String
+ getWeight(): double
+ setWeight(double): void
DigitalProduct
+ DigitalProduct(String, String, int, double):
+ toString(): String
GiftablePhysicalProduct
+ GiftablePhysicalProduct(String, String, int, double, double, String):
- giftMessage: String
+ getMessage(): String
+ setMessage(String): void
+ toString(): String
<<interface>>
Gifts
+ setMessage(String): void
+ getMessage(): String
GiftableDigitalProduct
+ GiftableDigitalProduct(String, String, int, double, String):
- giftMessage: String
+ toString(): String
+ setMessage(String): void
+ getMessage(): String
ShoppingCart
+ ShoppingCart():
- BASE_FEE: double
- name: String
- shoppingCart: Set<String>
+ setShoppingCart(Set<String>): void
+ setName(String): void
+ calculateWeight(): double
+ addItem(String): boolean
+ getName(): String
+ getBASE_FEE(): double
+ removeProductFromCart(): void
+ addProductToCart(): void
+ removeItem(String): boolean
+ cartAmount(): double
+ toString(): String
+ getShoppingCart(): Set<String>
1..M
1
Contain
ShoppingCartList
+ ShoppingCartList():
- cartCount: int
- shoppingCartList: ArrayList<ShoppingCart>
+ sortShoppingCartByWeight(boolean): ArrayList<ShoppingCart>
+ viewAllShoppingCartSortedByWeight(): void
+ getShoppingCartList(): ArrayList<ShoppingCart>
+ viewAllShoppingCart(): void
+ getCartCount(): int
+ createShoppingCart(): void
Main
+ Main():
+ main(String[]): void
Use
CartComparator
+ CartComparator(boolean):
- isAscending: boolean
+ compare(ShoppingCart, ShoppingCart): int

<u>**Abstract Product Class:**</u>

- **Name**: **String** - The name of the product, also used as a unique identifier of the products
- **quantityAvailable**: **int** - The number of available products in the product list
- **giftMessage: String** - The attached gift message of the product, to implement for the giftable product
- **Description**: **String** - The description of the product
- **productList**: **ArrayList<Product>** - A list containing all available products to buy
- **price**: **double** - Price of the product
-

All products consist of a unique name, quantity, gift message, description, and price. Those attributes are contained in the abstract class Product. The gift message is included because every type of product can be a gift, so it is available for later implementation.

The product list is left contained in the Product class because there is no related attribute or operation that needs to access and write to the list frequently, so it is not necessary to split it into a separate class for this system.

The class's attributes are set with a private access modifier and contain all the public getter and setter methods, for the sub-classes to inherit and for retrieving data from the outside of the class.

The abstract Product class has two sub-classes Digital Product and Physical Product class.

<u>**Concrete Digital Product Class:**</u>
Digital Product Class extends the Product Class with no additional attributes, and only overrides a **toString** method to return: "DIGITAL - <product name>"

<u>**Concrete Physical Product Class:**</u>
Physical Product Class extends the Product Class with an additional attribute:

- **Weight**: double - The weight of the product

The class provides a public getter and setter method to access this weight attribute, it also overrides a **toString** method to return: "PHYSICAL - <product name>"

<u>**Concrete Giftable Physical Product and Giftable Digital Product Class:**</u>
These 2 classes extend from the according parent classes Physical Product and Digital Product, with the implementation of the Gifts interface. This implementation provides the 2 classes with additional methods:

+ **getMessage(): String** - Get the gift message of the giftable product
+ **setMessage(String): void** - Set the gift message for the giftable product

<u>**Shopping Cart Class:**</u>

- **BASE_FEE: double** - The fee for delivery of the physical products, currently set to 0.1
- **name: String** - The unique name of the shopping cart base on the cart count, also used as a unique identifier of the shopping cart
- **shoppingCart: Set<String>** - A set of strings containing all products have been added to the cart

Every shopping cart have a unique identifier name based on the current number of the cart count, and a set of strings to store the name of all the products have been added to the cart. The shopping cart provides methods to add, and remove items from the cart, calculate the total weight and price of the cart as well as other getters and setters methods.

<u>**Shopping Cart List Class:**</u>

- **cartCount: int** - Number of carts have been created
- **shoppingCartList: ArrayList<ShoppingCart>** - A list containing every shopping cart

This class is a list containing and manages all the shopping carts created by the user. This class provides a variety of methods:

+ **sortShoppingCartByWeight(boolean): ArrayList<ShoppingCart>** - Return a list of shopping carts sorted by weight to use for viewAllShoppingCartSortedByWeight() method
+ **viewAllShoppingCartSortedByWeight(): void -** Print all the shopping carts in ascending or descending order
+ **viewAllShoppingCart(): void** - Print all the shopping carts in creation order
+ **createShoppingCart(): void** - Create a new shopping cart and add it in the shopping cart list

<u>**Gifts Interface**</u>

This interface allows the implemented class the ability to create and access a gift message for that product

<u>**Cart Comparator Class**</u>

This class implements the Comparator class and overrides the **compare()** method to compare the weight of 2 shopping carts in ascending or descending order.

<u>**Main Class**</u>

This class contains a main method that is used to run the program and provide the user with a simple text-based UI.

VI. Object-Oriented Programming Design Evaluation

There are 4 principles of OOP design in developing a software application, this section will evaluate how the program apply those principles.

<u>**Encapsulation**</u>

- **Modularity**

The whole program is divided into smaller inter-connected classes that interact with each other, in this case, products, and shopping carts are created and managed in different classes. Shopping Cart List and Product List are also kept independent. This makes the system easier to manage and maintain in the future.

- **Data and Information hiding**

Every class of this shopping system has its attributes set as private, to prevent other classes from accessing their private member directly and therefore avoid unauthorized access or manipulation. Instead, public getters and setters methods are provided for those attributes that need to be accessed by external classes.

An example of this is the getProductList() method of the product class used for other classes to interact with the product list

<u>**Inheritance**</u>

- **Inheritance**

The program makes use of this “IS A” relationship by making an abstract Product Class and having Physical/Digital Products Class extend the Product class, to inherit the predefined attributes and methods of the parent class. This 2 class then extends further into another 2 sub-classes Giftable Physical/Digital Product with an implementation of the Gifts interface

There is also composition where ShoppingCartList contains Shopping Carts with one-to-many relationships

- **Flexibility and Code reuse**

As said above, the child product class will inherit all the defined behavior of its parents, which will save time and resource by reusing the pre-written code, this also increases the flexibility of the program if the developer needs to update the program, then they only need to make a change to the abstract parent class.

<u>**Abstraction**</u>

- **Abstract Class**

Abstract class Product makes the backbone for all types of products and prevents a general product to be made.

- **Interface**

Interface: Giftable Digital/Physical implement Gifts interface forcing those types of product to have implementation for setting a gift message, this also decreases the coupling of the system by separating the 2 types of products.

<u>**Polymorphism**</u>

- **Method Overriding**

The sub-classes Digital Product and Physical Product override the constructor of the Product class to add more control of the attributes and re-use the code from the parents class. The 2 Giftable Physical and Digital Product classes also override the method getMessage() and getMessage() to have their own implementation.

In conclusion, the system has applied the concepts of an OOP design. Although not all principles were used, the program has shown some applications as well as the advantages of those implementations.

VII. Software Quality Evaluation

According to the ISO/IEC 9126 in Software Engineering, there are 6 characteristics to determine the quality of a software product (GeeksforGeeks 2022). Although, this section will only cover the appropriate as well as additional characteristics applied to this project's requirements.

<u>**Requirement Satisfaction/Functionality**</u>

The design of the program has successfully delivered all requirements of the project description, as demonstrated in section 4 - Program Demonstration.

<u>**Reliability**</u>

With the assumption that user input is always correct and appropriate, All functions are able to perform, without breaking the program.

<u>**Maintainability**</u>

The software follows and applies the principles of OOP design as evaluated by section VI - Object-Oriented Programming Design Evaluation. The system is easy to maintain and expand for new features in the future without impacting the pre-existing code, by implementing new interfaces and adding attributes, and methods for the parent classes. The code also has adequate documentation and is reader-friendly.

Efficiency

For the scale of the project, this system prioritizes saving storage space, instead of computing resources. For instance, Shopping Cart only stores a set of strings for the contained products, instead of the whole product object, also when the user calculates the total weight or price of a shopping cart, the program has to go through the whole product list, instead of storing a total attribute, which will result in more computing resources being used

Testability

The software provides a unit test for every major functionality and testable methods, the test cases attempted to cover every case scenario that could happen, and make sure the error throw a clear and descriptive exception message.

Usability

The software provides the user with an intuitive text-based user interface according to the project description. It provides the user with enough information and guidance on how to use the program, it also provides feedback messages for the user on every tasks.

VIII. Conclusion

In conclusion, the system design for the Online Shopping Service has been presented, analyzed, and evaluated according to the Object-Oriented Programming design principles and the industry's Software Quality Standard - ISO/IEC 9126 in Software Engineering, and the program has successfully achieved its objectives and provides a solid foundation for further development and implementation.

IX. Reference

Stemmler K (2022) *4 Principles of Object-Oriented Programming*, https://khalilstemmler.com website, accessed 2 April 2023. https://khalilstemmler.com/articles/object-oriented/programming/4-principles/

GeeksforGeeks (2022) *ISO/IEC 9126 in Software Engineering*, GeeksforGeeks website, accessed 2 April 2023. https://www.geeksforgeeks.org/iso-iec-9126-in-software-engineering/