1. Design for class : CartController

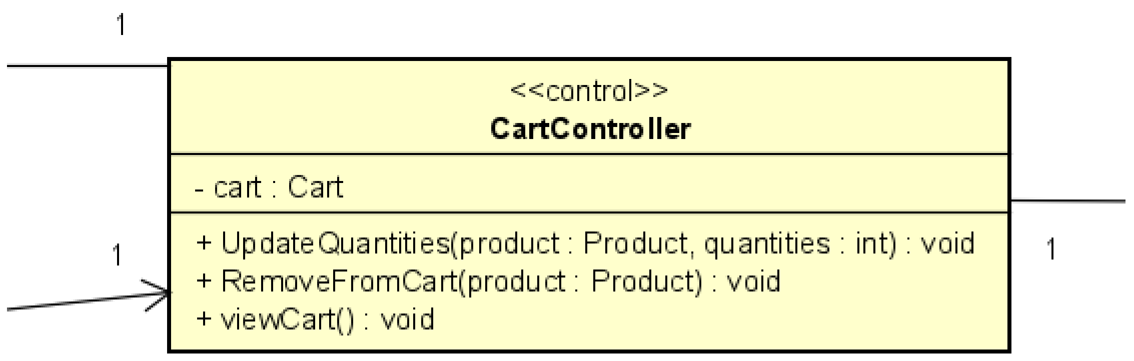

1
<<control>>
CartController
- cart : Cart
+ UpdateQuantities(product : Product, quantities : int) : void
+ RemoveFromCart(product : Product) : void
+ viewCart() : void
1
1

Table 1. Attribute design

#	Name	Data Type	Default Value	Description
1	cart	Cart	Null	Represents the shopping cart associated with the controller

Table 2: Operation design

#	Name	Return Type	Description (Purpose)
1	UpdateQuantities	void	Updates the quantity of a product in the cart
2	RemoveFromCart	void	Removes a product from the cart
3	viewCart	void	Displays the contents of the cart

Method: UpdateQuantities

- **Parameters:**
    - product: Product -> The product to update
    - quantity: int -> new quantity for the product
- **Exceptions:**
    - InvalidQuantityException if the quantity is negative
    - ProductNotFoundException if the product is not in the cart

- **How to use parameters/attributes (Algorithm to implement operation):**

1. Check if quantity > 0, otherwise throw InvalidQuantityException
2. Verify if the product exists in cart
3. If found, update its quantity; otherwise, throw ProductNotFoundException

Method: RemoveFromCart

- **Parameters:**
  - product: Product -> the product to remove
- **Exceptions:**
  - ProductNotFoundException if the product is not in the cart
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Check if product exists in cart
2. If found, remove it
3. Otherwise, throw ProductNotFoundException

Method: viewCart

- **How to use parameters/attributes (Algorithm to implement operation):**
  1. Retrieve all products in the cart
  2. Display product details

2. Design for class “CartScreen”

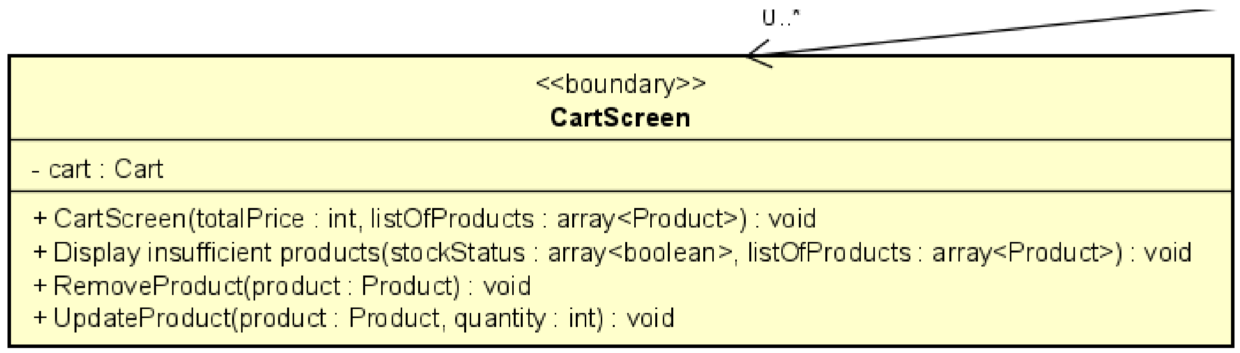

0..*
<<boundary>>
CartScreen
- cart : Cart
+ CartScreen(totalPrice : int, listOfProducts : array<Product>) : void
+ Display insufficient products(stockStatus : array<boolean>, listOfProducts : array<Product>) : void
+ RemoveProduct(product : Product) : void
+ UpdateProduct(product : Product, quantity : int) : void

Table 1. Attribute design

#	Name	Data Type	Default Value	Description
1	cart	Cart	Null	Represents the shopping cart shown on the screen

Table 2: Operation design

#	Name	Return Type	Description (Purpose)
1	CartScreen	void	Displays cart details including products and total price
2	Display insufficient products	void	Shows products with low stock
3	RemoveProduct	void	Removes a product from the cart
4	UpdateProduct	void	Updates the quantity of a product in the cart

Method: CartScreen

- **Parameters:**
    - totalPrice: float -> total price of the cart
    - listOfProducts: array<Product> -> list of products in the cart
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Display totalPrice
2. List products with their name, price, and quantity

Method: Display insufficient products

- **Parameters:**
    - stockStatus: array<Boolean> -> indicates product availability
    - listOfProducts: array< Product> -> list of products to check
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Check each product’s availability
2. If stock is low, mark it as insufficient

Method: RemoveProduct

- **Parameters:**
    - product : Product -> the product to remove
- **How to use parameters/attributes (Algorithm to implement operation):**

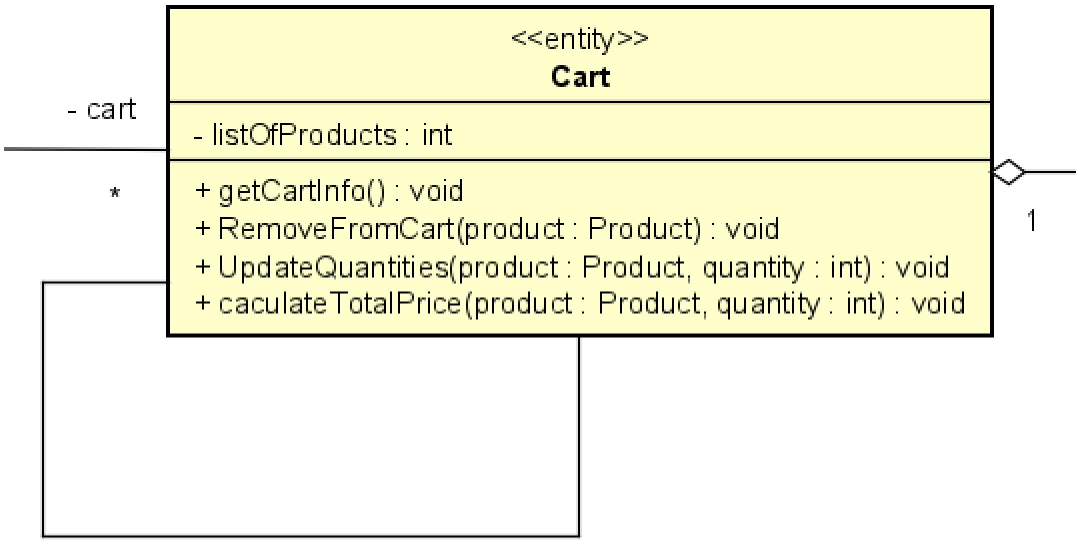
1. Call function CartController.RemoveFromCart(product)

Method: UpdateProduct

- **Parameters:**
    - product : Product -> the product to remove
    - quantity: int -> desired quantity of product
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Call function CartController.UpdateProduct(product, quantity)

3. Design for class "Cart"



<<entity>>
Cart
- listOfProducts : int
+ getCartInfo() : void
+ RemoveFromCart(product : Product) : void
+ UpdateQuantities(product : Product, quantity : int) : void
+ caculateTotalPrice(product : Product, quantity : int) : void
- cart
*
1

Table 1. Attribute design

#	Name	Data Type	Default Value	Description
1	listOfProducts	array of Product	Empty array	Stores all products in the cart

Table 2: Operation design

#	Name	Return Type	Description (Purpose)
1	getCartInfo	void	Retrieves cart details
2	RemoveFromCart	void	Removes a product from the cart
3	UpdateQuantities	void	Updates the quantity of a product
4	calculateTotalPrice	float	Computes the total cost of products in the cart

Method: getCartInfo

- • **How to use parameters/attributes (Algorithm to implement operation):**
    1. Fetch all products and display them

Method: calculateTotalPrice

- **Parameters:**
    - product: Product -> product to calculate
    - quantity: int, quantity of the product
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Multiply product.currentPrice * quantity
2. Return total price

Method: RemoveFromCart

- **Parameters:**
    - product: Product -> product to be removed
- **Exceptions:**
    - ProductNotFoundException if the product is not in the cart
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Search product in listOfProducts
2. If found, remove product from the cart, otherwise, throws ProductNotFoundException

Method: UpdateQuantities

- **Parameters:**
    - product: Product -> product to be removed
    - quantity: int -> desired quantity of product
- **Exceptions:**
    - ProductNotFoundException if the product is not in the cart
    - InvalidProductQuantityException if the product is insufficient
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Search product in listOfProducts
2. If found, check the availability of product that the stock quantity is still enough for customers to update

4. Design for class “Product”

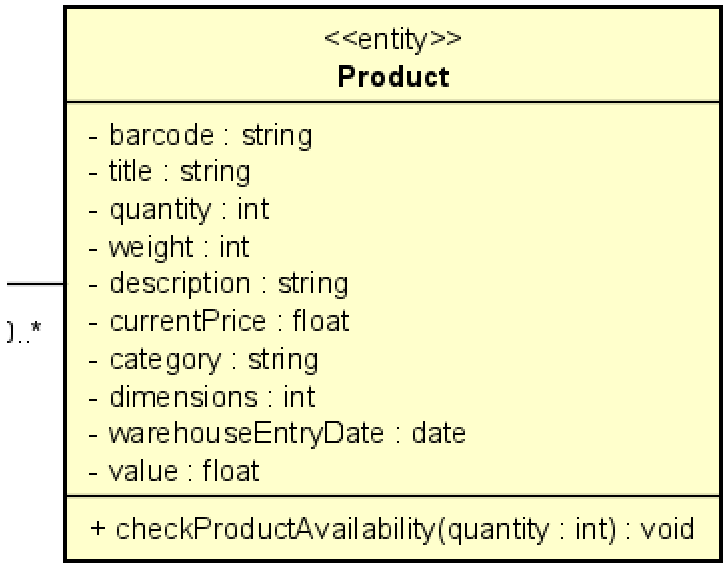


Table 1. Attribute design

#	Name	Data type	Default value	Description
1	description	string	""	Description of the product.
2	Barcode	String	“”	Unique identifier of the product
3.	warehouseEntryDate	Date	Null	Date when the product enterd the warehouse
4	Weight	Int	0	Product weight
5	Dimensions	Int	0	Available quantity
6	Quantity	Int	0	Available quantity
7	currentPrice	Float	0.0	Price per unit
8	Category	String	“”	Product category
9	Value	Float	0.0	Product value

10	Title	String	“”	Product title

Table 2: Operation design

#	Name	Return Type	Description (Purpose)
1	checkProductAvailability(quantity)	boolean	Verifies if the requested quantity is available

Method: checkProductAvailability

- **Parameters:**
    - quantity: int, requested quantity
- **Exceptions:**
    - InsufficientStockException if quantity is more than available stock
- **How to use parameters/attributes (Algorithm to implement operation):**

1. Check if quantity <= this.quantity
2. If yes, return true
3. Otherwise, throw InsufficientStockException

5. Design for class “HomeScreen”

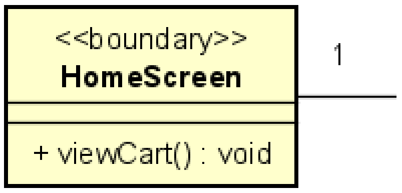


Table 2: Operation design

#	Name	Return Type	Description (Purpose)
1	viewCart()	void	Displays the cart screen

Method: viewCart

- **How to use parameters/attributes (Algorithm to implement operation):**

    1. Fetch cart details from CartController
    2. Display cart information