Engineering Document

Table of Contents

Homework #1

Changes #1 12/6/21

Changes

- **Use case 0**
- **Reflection questions**

Worked with Jake Wallis
Making changes to Use Case 0

1) Customers requests to view cart **external initiator**
2) Shopping cart retrieves all items in it.
3) Shopping cart totals the cost of all item's discounts
4) Shopping cart totals the cost of all items with discounts
5) Shopping cart totals the amount of tax owed from all items.
6) Shopping cart returns all the information that is generated to the customer.

Special Requirement

- [Modifiability] The algorithm that taxes are estimated with varies significantly with the customer's full address
- [Persistence] Every customer has their own cart
- [Persistence] Each customer's cart state must persist between sessions, even when the system loses power unexpectedly

Emails must be sent before thursday

Reflection questions

1. What is the scope of this project? What are you making? What are you not making?
1a) We are not dealing with anything happening in the front end, GUI and anything like that, we are dealing with more backend code where we get requests to do stuff.
2. Who is your audience? Who will use your code?
2a) our audience is the people coding the front end code and sending in requests to do certain things.
3. How will people use your code? What is their "UI?"
3a) People in the front end of the code will send requests to do certain tasks. We do not have a specific UI for the front end.
4. What aspects of this assignment seem the most technically challenging? What are you going to do to mitigate those risks to completing the assignment?
4a) The tasks that seem the most difficult seem to be the idea of connecting parts together and making our own ideas up on what is wrong and what is right.

Changes #2 12/7/21

Changes

- Added use case 1, 2 and partially 3

Use Case 1: Add item to cart
Actor: Customer
Precondition: customer found an item they want to buy
Basic Flow:

1) Customer clicks on button that states “add to cart”
    a) Customer states the item they want based on Id
2) Item request is sent to database
3) Database locates the item based on Id
4) Item is then added to cart(some kind of list)

Exception Flows:
Any Step: Item goes out of stock before customer can add it to their cart

- System does not add the item to the customer’s cart
- System displays an error message that the item is no longer available

Special Requirements:

- [Modifiability] Vendors change items for sale all the time
- [Persistence] if no items of some type are in stock, information about the item is lost

Use case 2: Apply discount code
Basic Flow:

1) Customer has a discount code
2) Customer clicks on, “apply discount code”
    a) System prompts if the customer has a discount code or not
    b) A system prompts the user to input the discount code
3) customer is prompted to input the discount code
4) System checks to see if discount is valid based on code
    a) If code is valid, apply discount by subtracting
    b) If code is invalid, do not apply discount
5) System updates current discount and total to reflect discount

Exception Flows:
Step 1.a. Customer applies an expired code
    1. System rejects the expired code
Step 1.b. Customer applies a code that does not apply to the contents of the shopping cart
    1. System indicates which items are missing
Step 1.c. Customer enters five expired/inapplicable discount codes within a span of 24 hours
    1. System silently rejects all further codes from this customer for 24 hours

Special Requirements:

• [Modifiability] Vendors change discount codes all the time.
• [Modifiability] Discounts vary in what items and what quantities are required to be in the cart, but otherwise all have the same behavior

Use case 3: Modify cart Quantity
Basic Flow:

1) Customer sees that there is a + and - sign on a item
2) Customer clicks on one of those modifications on the item
3) Based on signal, the system sends

Changes #3 12/10/21

Changes:

- ## Textual Analysis

Making changes to Use Case 0

1. Customers requests to view cart **external initiator**
2. Shopping cart retrieves all items in it.
3. Shopping cart totals the cost of all item's discounts
4. Shopping cart totals the cost of all items with discounts
5. Shopping cart totals the amount of tax owed from all items.
6. Shopping cart returns all the information that is generated to the customer.

Special Requirement

- [Modifiability] The algorithm that taxes are estimated with varies significantly with the customer's full address
- [Persistence] Every customer has their own cart
- [Persistence] Each customer's cart state must persist between sessions, even when the system loses power unexpectedly

Use Case 1: Add item to cart
Actor: Customer
Precondition: customer found an item they want to buy
Basic Flow:

5) Customer clicks on button that states "add to cart"
    a) Customer states the item they want based on Id
6) Item request is sent to database
7) Database locates the item based on Id
8) Item is then added to cart(some kind of list)

Exception Flows:
Any Step: Item goes out of stock before customer can add it to their cart

- System does not add the item to the customer's cart
- System displays an error message that the item is no longer available

Special Requirements:

- [Modifiability] Vendors change items for sale all the time
- [Persistence] if no items of some type are in stock, information about the item is lost

Use case 2: Apply discount code
Basic Flow:

1. Customer has a discount code
2. Customer clicks on, "apply discount code"
    a. System prompts if the customer has a discount code or not
    b. A system prompts the user to input the discount code

3. customer is prompted to input the discount code
4. System checks to see if discount is valid based on code
    a. If code is valid, apply discount by subtracting
    b. If code is invalid, do not apply discount
5. System updates current discount and total to reflect discount

Exception Flows:

Step 1.a. Customer applies an expired code

1. System rejects the expired code

Step 1.b. Customer applies a code that does not apply to the contents of the shopping cart

1. System indicates which items are missing

Step 1.c. Customer enters five expired/inapplicable discount codes within a span of 24 hours

1. System silently rejects all further codes from this customer for 24 hours

Special Requirements:

• [Modifiability] Vendors change discount codes all the time.

• [Modifiability] Discounts vary in what items and what quantities are required to be in the cart, but otherwise all have the same behavior

Use case 3: Modify cart Quantity

Basic Flow:

1. Customer sees that there is a + and - sign on a item
2. Customer clicks on one of those modifications on the item
3. Based on signal, the system tells shopping cart to increase quantity of item contained or, decrease quantity of item
4. System removes discounts that only apply to buying the old quantity of items.
5. System updates total to reflect the new cart.

Alternate Flow

Step 1.a. Customer sets the desired quantity to 0

• System shows the new cart without the removed item Exception Flows:

Step 1.a. One or more items went out of stock since the customer last updated the shopping cart

1. The system displays the cart again and highlights which item is out of stock

Special Requirements:

• [Integrity Constraint] Quantity cannot be negative

Changes #4 12/13/21

Changes helped by Jake Wallis:

- **Changes to textual analysis**
- **Changes to use cases**
- **Changes to what use case 0**
    - **Applying discounts**

Making changes to Use Case 0

1. Front End: Customers requests to view cart **external initiator**
2. Shopping cart retrieves all items in it.
3. Shopping cart totals the amount of tax owed from all items.
4. Shopping cart returns all the information that is generated .

Special Requirement

- [Modifiability] The algorithm that taxes are estimated with varies significantly with the customer's full address
- [Persistence] Every customer has their own cart
- [Persistence] Each customer's cart state must persist between sessions, even when the system loses power unexpectedly

Use Case 1: Add item to cart
Actor: Customer
Precondition: customer found an item they want to buy
Basic Flow:

1. Front End: Customer clicks on button that states "add to cart"
    a. Customer states the item they want based on Id
2. Item request is sent to database
3. Database locates the item based on Id
4. Item is then added to cart(some kind of list)

Exception Flows:
Any Step: Item goes out of stock before customer can add it to their cart

- System does not add the item to the customer's cart
- System displays an error message that the item is no longer available

Special Requirements:

- [Modifiability] Vendors change items for sale all the time
- [Persistence] if no items of some type are in stock, information about the item is lost

Use case 2: Apply discount code
Basic Flow:

1. Front End: Customer has a discount code

2. Front End: Customer clicks on, “apply discount code”
    a. System prompts if the customer has a discount code or not
    b. A system prompts the user to input the discount code
3. Front End: customer is prompted to input the discount code
4. System checks to see if discount is valid based on list
    a. If code is valid, apply discount by subtracting
    b. If code is invalid, do not apply discount
5. Shopping cart totals the cost of all item’s discounts
6. Shopping cart totals the cost of all items with discounts
7. System updates current discount and total to reflect discount

Exception Flows:

Step 1.a. Customer applies an expired code

1. System rejects the expired code

Step 1.b. Customer applies a code that does not apply to the contents of the shopping cart

1. System indicates which items are missing

Step 1.c. Customer enters five expired/inapplicable discount codes within a span of 24 hours

1. System silently rejects all further codes from this customer for 24 hours

Special Requirements:

• [Modifiability] Vendors change discount codes all the time.

• [Modifiability] Discounts vary in what items and what quantities are required to be in the cart, but otherwise all have the same behavior

Use case 3: Modify cart Quantity

Basic Flow:

1. **Front End:** Customer sees that there is a + and - sign on a item
2. **Front End:** Customer clicks on one of those modifications on the item
3. Based on signal, the system tells shopping cart to increase quantity of item contained or, decrease quantity of item
4. System removes discounts that only apply to buying the old quantity of items.
5. System updates total to reflect the new cart.

Alternate Flow

Step 1.a. Customer sets the desired quantity to 0

• System shows the new cart without the removed item Exception Flows:

Step 1.a. One or more items went out of stock since the customer last updated the shopping cart

1. The system displays the cart again and highlights which item is out of stock

Special Requirements:

• [Integrity Constraint] Quantity cannot be negative

Changes #5 12/14/21

Changes:

- **Changes to textual analysis**
- **Changes to use cases**

Making changes to Use Case 0

1. Front End: Customers requests to view cart **external initiator**
2. Shopping cart retrieves all items in it.
3. Shopping cart totals the amount of tax owed from all items.
4. Shopping cart returns all the information that is generated .

Special Requirement

- [Modifiability] The algorithm that taxes are estimated with varies significantly with the customer's full address
- [Persistence] Every customer has their own cart
- [Persistence] Each customer's cart state must persist between sessions, even when the system loses power unexpectedly

Use Case 1: Add item to cart
Actor: Customer
Precondition: customer found an item they want to buy
Basic Flow:

1. Front End: Customer clicks on button that states "add to cart"
    a. Customer states the item they want based on Id
2. Item request is sent to Shopping cart
3. Shopping cart locates the item based on Id
4. Item is then added to Shoppingcart

Exception Flows:
Any Step: Item goes out of stock before customer can add it to their cart

- System does not add the item to the customer's cart
- System displays an error message that the item is no longer available

Special Requirements:

- [Modifiability] Vendors change items for sale all the time
- [Persistence] if no items of some type are in stock, information about the item is lost

Use case 2: Apply discount code
Basic Flow:

1. Front End: Customer has a discount code

2. Front End: Customer clicks on, “apply discount code”
    a. System prompts if the customer has a discount code or not
    b. A system prompts the user to input the discount code
3. Front End: customer is prompted to input the discount code
4. Shopping cart checks to see if discount is valid based in list
    a. If code is valid, apply discount by subtracting
    b. If code is invalid, do not apply discount
5. Shopping cart totals the cost of all item’s discounts
6. Shopping cart totals the cost of all items with discounts
7. Updates current discount and total to reflect discount being applied

Exception Flows:

Step 1.a. Customer applies an expired code

1. System rejects the expired code

Step 1.b Customer enters five expired/inapplicable discount codes within a span of 24 hours

1. System silently rejects all further codes from this customer for 24 hours

Special Requirements:

• [Modifiability] Vendors change discount codes all the time.

Use case 3: Modify cart Quantity

Basic Flow:

1. **Front End:** Customer sees that there is a + and - sign on a item
2. **Front End:** Customer clicks on one of those modifications on the item
3. Shopping cart increases quantity of item contained or, decreases quantity of item
4. Remove all applied discounts from modified item.

Alternate Flow

Step 1.a. Customer sets the desired quantity to 0

• System shows the new cart without the removed item Exception Flows:

Step 1.a. One or more items went out of stock since the customer last updated the shopping cart

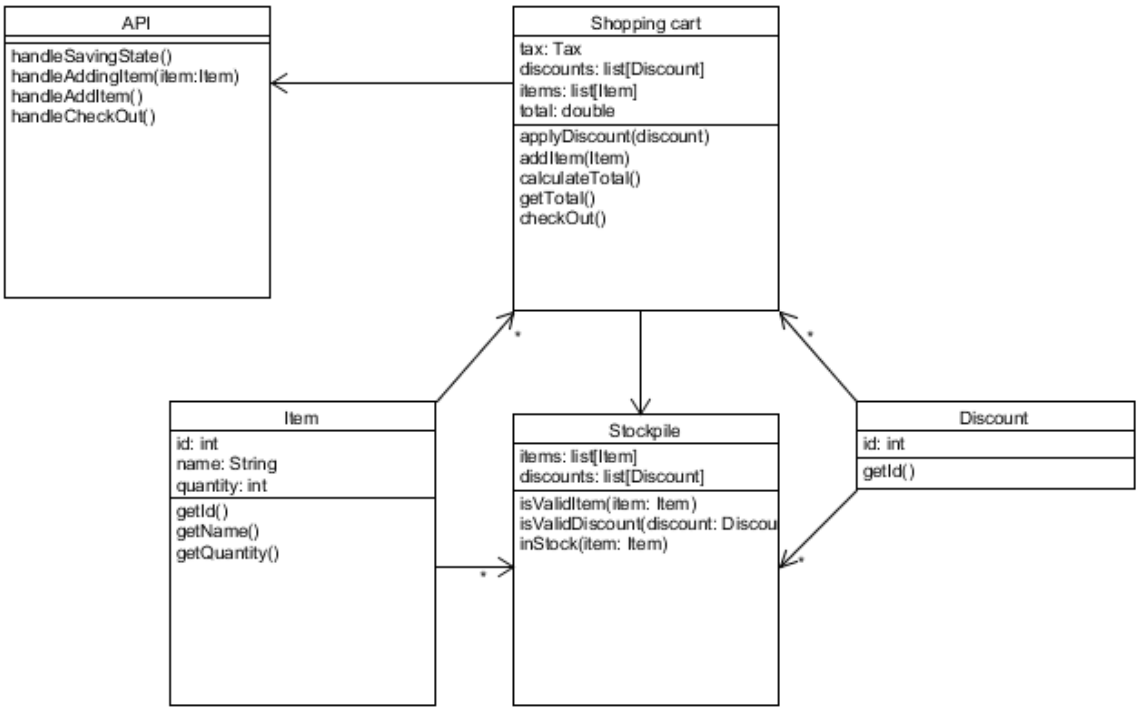
1. The system displays the cart again and highlights which item is out of stock

Special Requirements:

• [Integrity Constraint] Quantity cannot be negative

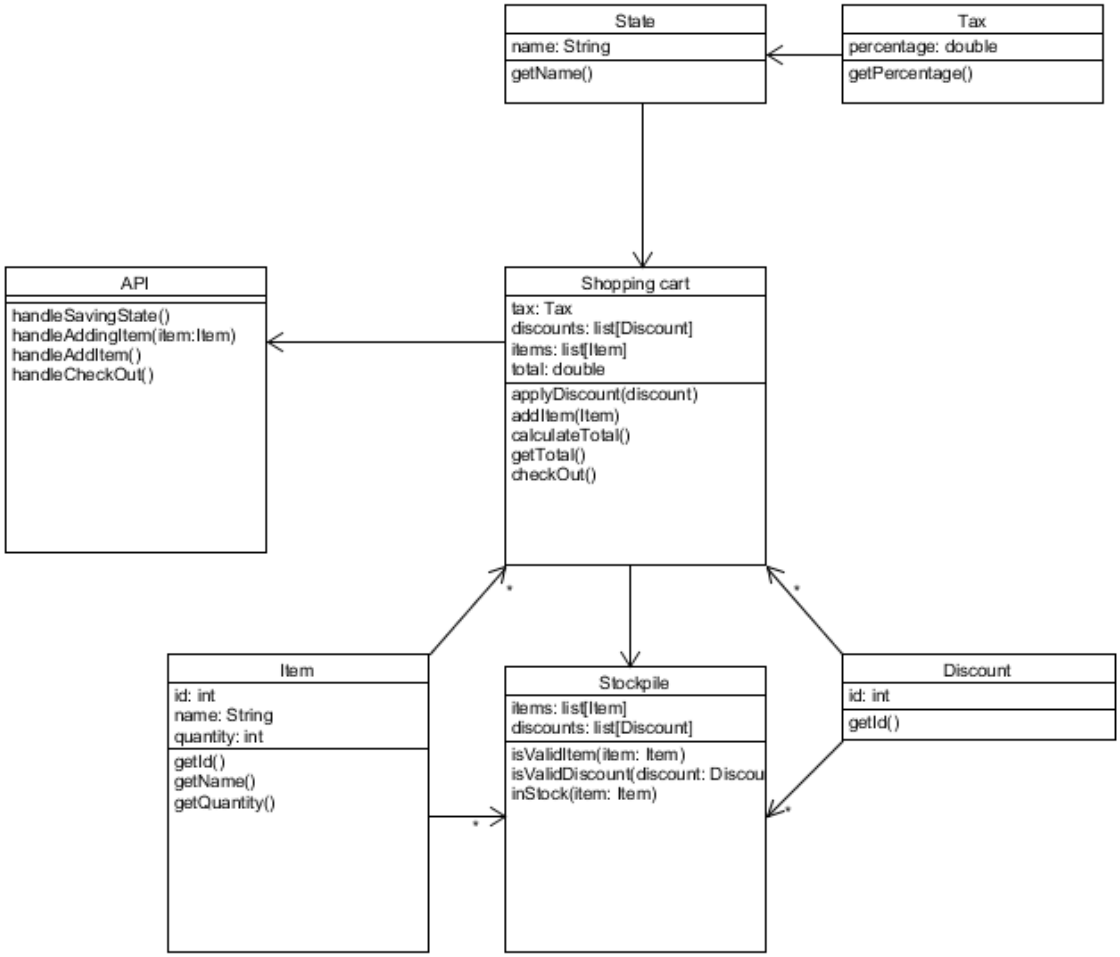
Changes #6 12/15/21

- Initial UML Diagram



Changes #7 12/15/21

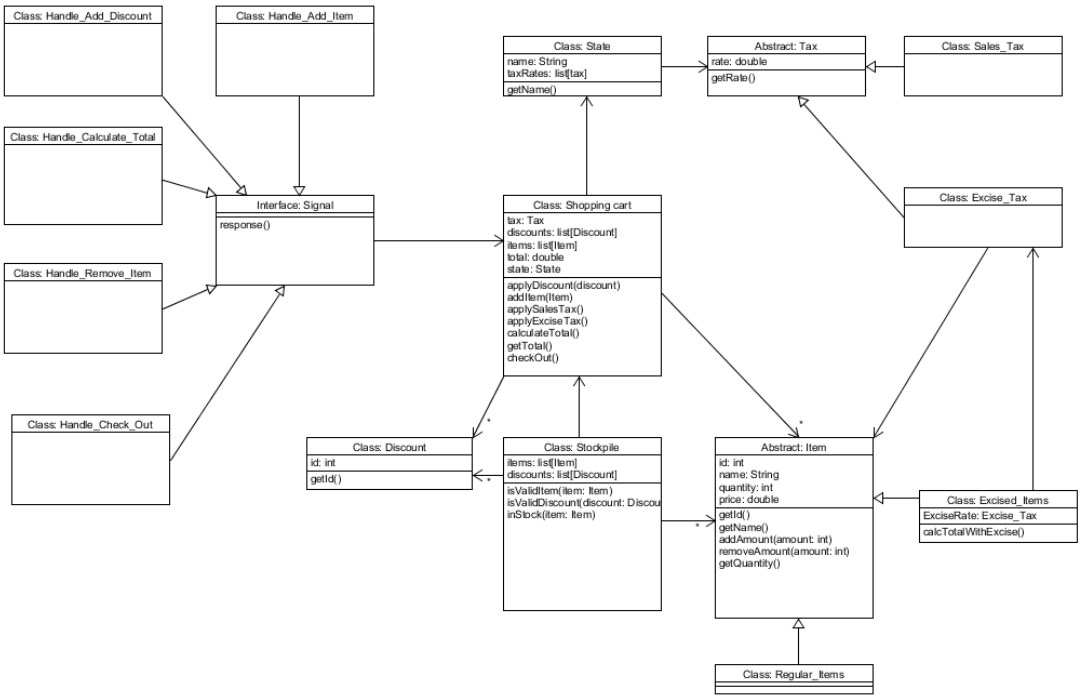
- Change to UML diagram



Changes #8 12/16/21

Changes Helped by Anakin, David Samson, and Robb Budak:

- UML Diagram changes, including different taxes and a different approach to handling signals sent from the front end



Changes #9 12/16/21

Changes

- very small changes to signals and stuff
- Implemented into eclipse and build test cases for complicated methods