My Project

Generated by Doxygen 1.8.16

Chapter 1

Todo List

Member Customer::expel ()

Revise the design of expellation and the unit test cases so expellation can be unit tested without this global variable counter hack.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 Cashier Class Reference

Interacts with Customers, charges them money, and parses their orders.

```
#include <Cashier.hpp>
```

3.1.1 *

Public Member Functions

- Cashier (double money_)

  Constructs the Cashier with a set amount of money.
- double get_money () const

  Gets the amount of money that the cashier holds.
- void serve_customer (std::queue< Customer > &line, std::stack< Order > &orders)

  Serves a customer by taking them off the queue, and possibly putting an order, if valid, onto the order stack.

3.1.2 Detailed Description

Interacts with Customers, charges them money, and parses their orders.

Note

Must give each Customer upon interaction a unique ID number, counting up from 0.

3.1.3 Member Function Documentation

3.1.3.1 serve_customer()

```
void Cashier::serve_customer (
            std::queue< Customer > & line,
            std::stack< Order > & orders )
```

Serves a customer by taking them off the queue, and possibly putting an order, if valid, onto the order stack.

- Retrieves the customer from the queue, popping them off.
- Receives the customer's desired order items in string format:
  <number-of-items> <name-of-item-with-no-spaces>
  - Multiple items are separated simply by a space
- If the Cashier detects an order for an invalid item that is not on the menu, it must expel() the customer.
- Must calculate the cost of all the items ordered, and charge the customer.
  - It the customer does not have enough money to pay, do not charge the customer, but instead, expel() them.
- Once paid for, the order items must be consolidated into an Order, which is tagged with the unique customer ID generated at the beginning of this function, and then push it onto the stack.

See also

Customer
Order

The documentation for this class was generated from the following file:

- Cashier.hpp

3.2 Cook Class Reference

Prepares dishes using a SupplyRunner and a Kitchen.

```
#include <Cook.hpp>
```

3.2.1 *

Public Member Functions

- Cook (SupplyRunner &runner_, Kitchen &kitchen_)
  Constructs a Cook with references to a SupplyRunner and a Kitchen, which it must use in the process of cooking.
- void prepare_dish (std::stack< Order > &orders, std::queue< std::pair< std::size_t, Dish >> &finished_↩ dishes)
  Prepares Dishes from a single order.

3.2.2 Detailed Description

Prepares dishes using a SupplyRunner and a Kitchen.

Note

Uses dependency injection with the constructor.

3.2.3 Member Function Documentation

3.2.3.1 prepare_dish()

```
void Cook::prepare_dish (
             std::stack< Order > & orders,
             std::queue< std::pair< std::size_t, Dish >> & finished_dishes )
```

Prepares Dishes from a single order.

- Take an Order from the stack.
- For each order item in the Order,
  - Lookup its required ingredients in the RecipeBook.
  - Then, ask the SupplyRunner to get the correct amount of ingredients.
  - Put the vector of ingredients into an IngredientMap.
  - Send the IngredientMap to the Kitchen to have it turned into a dish.
  - Put the finished Dish onto the queue, embedding it in a pair that also contains the Customer ID of origin.

The documentation for this class was generated from the following file:

- Cook.hpp

3.3 Customer Class Reference

A class to represent restaurant customers.

`#include <Customer.hpp>`

3.3.1 *

Public Member Functions

- Customer (double money_, std::string order_str_)

  Constructs a Customer with a set amount of money and a string to emit upon have its order taken as a string.
- double get_money () const

  Gets the current amount of money on the Customer.
- bool is_expelled () const

  Returns whether the Customer was expelled.
- std::string take_order ()

  Returns the order_str on the Customer.
- double charge_money (double amount)

  Attempt to charge the amount of money from the Customer.
- void refund_money (double amount)

  Refunds money back to the customer.
- std::string get_order () const

  Returns the order string.
- void expel ()

  Expels the Customer from the restaurant.

3.3.2 *

Friends

- std::ostream & operator<< (std::ostream &lhs, const Customer &rhs)
  Prints out a string representation of the Customer's fields.

3.3.3 Detailed Description

A class to represent restaurant customers.

3.3.4 Member Function Documentation

3.3.4.1 charge_money()

```
double Customer::charge_money (
             double amount )
```

Attempt to charge the amount of money from the Customer.

Note

If the Customer doesn't have enough money, it will simply return its current money amount and drain its money to 0.

3.3.4.2 expel()

```
void Customer::expel ( )
```

Expels the Customer from the restaurant.

Note

For the purposes of testing, this is attached to the `extern std::size_t expelled_count` variable.

Todo Revise the design of expellation and the unit test cases so expellation can be unit tested without this global variable counter hack.

3.3.4.3 get_order()

```
std::string Customer::get_order ( ) const
```

Returns the order string.

See also

take_order()

3.3.4.4 refund_money()

```
void Customer::refund_money (
             double amount )
```

Refunds money back to the customer.

Note

There are no bounds checking on this code, so you can refund negative money to charge money as well. Please do not do this; this is not how this member function is supposed to be used.

The documentation for this class was generated from the following file:

- Customer.hpp

3.4 Kitchen Class Reference

A Kitchen that can be used to prepare Dish objects by providing them a moved-map of ingredients.

`#include <Kitchen.hpp>`

3.4.1 *

Public Member Functions

- Dish prepare_dish (IngredientMap &&ingredients)

    Consumes the map of ingredients (std::string) and returns the corresponding Dish.

3.4.2 Detailed Description

A Kitchen that can be used to prepare Dish objects by providing them a moved-map of ingredients.

3.4.3 Member Function Documentation

3.4.3.1 prepare_dish()

```
Dish Kitchen::prepare_dish (
             IngredientMap && ingredients )
```

Consumes the map of ingredients (std::string) and returns the corresponding Dish.

Note

This will return Dish::INEDIBLE if a map of ingredients that dosn't correspond to a proper dish is given.

You must noticably MOVE the ingredients in or give it a map literal.

See also

Ingredient.hpp

Note

Look at the lab manual for more information about the item costs.

The documentation for this class was generated from the following file:

- Kitchen.hpp

3.5 Order Class Reference

Represents a single Order for a list of items.

```
#include <Order.hpp>
```

3.5.1 *

Public Member Functions

- Order (std::size_t id_, std::vector< std::string > items_)

  Constuctors an Order with a customer ID and the ordered items.
- std::size_t get_id () const

  Gets the ID associated with the Order.
- std::vector< std::string > get_items () const

  Gets the vector of order items (strings).

3.5.2 *

Friends

- std::ostream & operator<< (std::ostream &lhs, const Order &rhs)

  Prints out a readable reprsentation of an Order to an ostream&.

3.5.3 Detailed Description

Represents a single Order for a list of items.

The documentation for this class was generated from the following file:

- Order.hpp

3.6 SupplyRunner Class Reference

Gets Ingredients from the Storeroom.

```
#include <SupplyRunner.hpp>
```

3.6.1 *

Public Member Functions

- SupplyRunner (Storeroom &storeroom_)

  Constructs a SupplyRunner with the Storeroom it gets its Ingredients from.
- std::vector< Ingredient > get_ingredients (IngredientMap ingredients)

  Gets ingredients from the Storeroom, if they exist, as a vector.

3.6.2 Detailed Description

Gets Ingredients from the Storeroom.

3.6.3 Member Function Documentation

3.6.3.1 get_ingredients()

```
std::vector<Ingredient> SupplyRunner::get_ingredients (
             IngredientMap ingredients )
```

Gets ingredients from the Storeroom, if they exist, as a vector.

Note

Will remove ingredients from the storeroom if all the ingredeints asked for are all found in sufficient numbers.

Exceptions

const	char∗ if there are not enough ingredients in the Storeroom.

Note

Upon not finding enough ingredients to return, the SupplyRunner will not modify the Storeroom in any way, including removing Ingredients from the Storeroom (decreasing the Ingredient count on the map).

The documentation for this class was generated from the following file:

- SupplyRunner.hpp

Index