UML and CRC design spec

CSC207 Group Project planning

Employee class (CRC)

Responsibilities

- Stores name, id, login information, etc. about a specific person.

Collaboration

- Restaurant

Employee/Person? class (CRC)

Variables

- count : Int
- id : Int
- name : String
- password : String

Methods

+ getCount() : Int
+ getId() : Int
+ getName() : String
+ toString() : String

Server class (CRC)

Responsibilities

- Store, retrieve and edit a list of customers that this server is catering to right now.
- Check items available for pick-up.
- Pick-up food from the kitchen and then confirm/reject it upon customer's request.

Collaboration

- Customer (for management)
- Kitchen (for pickup, and order)
- Ingredient (for orders)
- Order
- Restaurant (for menu)

Server class (UML)

Variables

- customers : ArrayList<Customer>
- kitchen : Kitchen

Methods

+ constructor(name: String)
+ addCustomer(c : Customer)
+ billCustomer(c : Customer) : String
+ getCustomer(table : Integer) : Customer
+ getCustomerById(id : Integer) : Customer
+ pickUpOrder(o : Order, reject : Bool)
+ getPickUpQueue() : Queue<Order>

Customer class (CRC)

Responsibilities

- Create and store all orders requested by this customer (with additions/subtractions for ingredients).
- Stores table number, server, confirmed orders, and rejected orders.
- Has a print bill method which lists all confirmed orders and prices (including cost of additions), as well as a total amount.

Collaboration

- Server (for all interactions)

Customer class (CRC)

Variables

- server : Server
- pendingOrders : ArrayList<Order>
- preparedOrders : HashMap<Order, Bool>
- table : Int

Methods

+ constructor(name : String, table : Int)
+ createOrder(o: Order)
+ receiveOrder(o : Order, reject : Bool)
+ getBill() : String
+ getTable() : Int

MenuItem class (CRC)

Responsibilities

- Stores name and ingredients (with default amounts, as well as corresponding limits).
- Contains method for getting changeable ingredients (where additions/subtractions are possible) and all ingredients.
- Contains default price for the menu item (without additions).

MenuItem class (CRC)

Variables

- name : String
- ingredients : ArrayList<Ingredient>
- price : Double

Methods

+ constructor(name: String)
+ addIngredient(i : Ingredient)
+ removeIngredient(i : Ingredient)
+ getChangeableIngredients() : ArrayList<Ingredient>
+ getIngredients() : ArrayList<Ingredient>
+ getName() : String
+ getPrice() : Double
+ equals() : Boolean

Order class (CRC)

Responsibilities

- Stores an id (created by incrementing static variable), and customer who created order.
- Stores the corresponding menu item that this order is based on.
- Contains methods for addition and subtraction of ingredients (within limits).
- Contains getter for price (which incorporates additions).

Collaborations

- Customer

Order class (CRC)

Variables

- count : Int
- id : Int
- customer : Customer
- menuItem : MenuItem
- ingredients : ArrayList<Ingredient>

Methods

+ *constructor(m : MenuItem)*
+ getCount() : Int
+ getId() : Int
+ getCustomer() : Customer
+ getMenuItem() : MenuItem
+ getPrice() : Double
+ setIngredientQuantity(ingredient : String, quantity: Int)
+ getIngredients() : ArrayList<Ingredient>

Ingredient (CRC)

Responsibilities

- Each ingredient has a threshold for when more needs to be reordered, as well a default amount to be reordered. When below threshold, item stored into requests.txt (with default reorder of 20 units) for manager to review.
-

Collaborations

+ Inventory
+

Ingredient (UML)

Variables

- - name: String
- - price: double
- - min: Double
- - max: Double
- - default: Double
- - extraCost: Double

Methods

- + constructor (name: String, min: int, max: int, default: int)
- + constructor (name: String, min: Double, max: double, default: double, extraCost : Double)
- + getName() : String
- + getPrice() : Double
- + getMin() : Double
- + getMax() : Double
- + getDefault() : Double
- + getExtraCost() : Double
- + equals() : Boolean

Inventory (CRC)

Responsibilities

- Holds Ingredient with their respective quantities
- Each Ingredient has limited number of additions/subtractions allowed for ingredients
- Should be possible for the Manager to check the inventory and see a printout of all inventory items and their quantities.
- Should be possible for a receiver to scan each item (or list of items) back into the inventory when a new shipment of ingredients arrives.

Collaborations

- Ingredient

Inventory (UML)

Variables

- Inventory: HashMap<Ingredient, Double>

Methods

+ getInventory()
+ removeItem(item: Ingredient, quantity)
+ addItem(item: Ingredient, quantity)
+ belowThresholdCheck()
+ toString() : String

Receiver class (CRC)

Responsibilities

- When food item shipments arrive update the inventory with the items and quantity of each

Collaborations

- Inventory

Receiver (CRC)

Variables

Methods

+ processItem(item: Ingredient, quantity: Double)
+ processItems(items: ArrayList<Ingredient>, quantities: ArrayList<Double>)

EventParser (CRC)

Responsibilities

- Create a simulation restaurant environment(s) (should this be done here?)
- Read lines from a .txt file where each line describes an event within the restaurant.
- Call the relevant methods from the relevant class to simulate each event
- Each event is written in chronological order

Collaborations

- All

EventParser (UML)

Variables

Methods

- readEventFile()
- processEvent(event: String)

Manager (CRC)

Responsibilities

- Access Inventory and see printout of all items
- Add Employees to log in

Collaborations

- Restaurant

Manager (CRC)

Variables

- Restaurant: Restaurant
- employees: Employee[]

Methods

+ getInventory()
+ addCook(name: String, id: int)
+ addServer(name: String, id: int)
+ addReciever(name: String, id: int)
+ addEmployee(e : Employee)
+ removeEmployee(e : Employee)

Restaurant (UML)

Variables

- name: String
- menu: MenuItem[]
- manager : ArrayList<Manager>
- kitchen: Kitchen

Methods:

+ Constructor(name: String, Kitchen: Kitchen)
+ addMenuItem(m : MenuItem)
+ addManager(m : Manager)
+ removeManager(m : Manager)

Restaurant (CRC)

-Can add another kitchen location

-Can add cooks, servers and recievers to a particular kitchen location

-

Cook

Person

Manager

Kitchen (UML)

Variable

- count : Int
- id: int
- OrderQueue: Order[]
- PrepareQueue: Order[]
- PickupQueue: Order[]

Methods

+ getId(): int
+ getOrderQueue(): Queue
+ getPrepareQueue(): Queue
+ getPickupQueue(): Queue
+ itemPickedUp(Order o)
+ itemReady(Order o)
+ itemOrdered(Order o)
+ itemPreparing(Order o)

Cook (UML)

Variables

- id: int
- name: String
- kitchen: Kitchen

Methods

+ orderSeen(o : Order): void
+ orderReady(o : Order): void

Cook (CRC)

Kitchen