OnlineOrderSystem

- ingredient_stock: Dict<str, (Ingredient, Int)>
- main_types: Dict<str,Main>
- products: Dict<str, Product>
- orders: List<Order>
- staff_codes: List<str>

+ search_ingredient(name: str): Ingredient or None
+ has_stock(ingredient: Ingredient OR str, val: int): void
+ has_stock(List<Ingredient, int>): void
+ increase_stock(staff_code: str, ingredient: Ingredient OR str, quantity: int): void
+ get_mains(available_only: boolean): List<MainType>
+ get_product(available_only: boolean): List<Product>
+ create_order(): Order
+ select_main_type(order: Order OR int, main: Main OR str): void
+ config_order(order: Order OR int, product: Product OR str, quantity: int): void
+ checkout(order: Order): void
+ cancel_order(order: Order OR id): void
+ get_order(order_id: int): Order
+ get_orders(staff_code: str, status: str): List<Orders>

Ingredient

- name: str
- unit: str

Product

- name: str
- ingredient_consumptions: List<(Ingredient, int)>
- price: float
- quantity_range_per_order: (int, int)

Drink

Side

IngredientForMain

Base

MajorFilling

Order

- id: int or None
- status: int [0: creating, 1: preparing, 2: ready, 3: picked up]
- main_type: MainType
- products: List<(Product, int)>

+ get_ingredients_consumption(): List<(Ingredient, int)>
+ compute_total_price(): float

MainType

- type: str ["Burger" or "Wrap"]
- ingredient_choices: List<(IngredientForMain, int)>
- minimum_choices: LIst<(IngredientForMain, int)>
- default_choices: LIst<(IngredientForMain, int)>
- base_price: float

+ get_ingredients_consumption(): List<(Ingredient, int)>

1..*

1

1   1..*

0..*

1

1

0..*

1..*

1

0..1

1