UML class Diagram

Mains order

- Total price: float
- Ingredients: dictionary{<Mains Ingredient>: int}

+ calc_total_price(self): float
+ Ingredients(self): dictionary{<Mains Ingredient>: int}
+ check_if_exceed_max(ingredients): bool
+ get_total_price(self): float

Ingredient

- Name: string
- Price: float
- Maximum_allowable: int
- Quantity_type: string

+ char * Name(self)
+ set_name(self, char*name)
+ float price(self)
+ set_price(self, price)
+ maximum_allowable(self)
+ set maximum_allowable(self, int)
+ check_within_max (self, int)

Bun

Patty

Mains Ingredient

Side

Drink

Wrap

Burgers

Ordering System

- Current_orders: list<order>
- Completed_orders: list<order>

+ get_order_by_orderID(int): object
+ get_current_orders(self): list<order>
+ get_completed_orders(self): list<order>
+ append_completed_orders(list): void
+ create_order(<main>, <drink>, <side>): <order>
+ add_to_order (<order>, <ingredient>, int): void
+ search_ingredients (string): list<ingredient>
+ view_order(list<order>) : void
+ checkout_order(list<order>): Order_ID
+ clear_orders_list(list<order>): void

Inventory System

- Ingredients: dictionary{<ingredient>: int}

+ get_ingredient(self, string): <ingredient>
+ update_stock_quantity(self, string, int): void
+ add_stock_type(self, string, string, int): void
+ get_stock_quantity(self, string): int
+ delete_stock(self, string): void

Drink Order

- Total price: float
- Ingredients: dictionary{<drink>: int}

+ calc_total_price(self): float
+ get_total_price(self): float

Side Order

- Total price: float
- Ingredients: dictionary{<side>: int}

+ calc_total_price(self): float
+ get_total_price(self): float

Order

- Price: float
- Status: string
- Order_ID: int
- Mains: list<mains order>
- Sides: <side order>
- Drinks: <drink order>

+ create_drink(<ingredient>, int): <drink>
+ create_side(<ingredient>, int): <side>
+ create_main(<ingredient>, int): <main>
+ update_Order_status(self, string): void
+ calculate_price (self): float
+ get_status(self): string
+ get_ID(self): int
+ get_price(self): float
+ get_mains(self): list[<mains order>]
+ get_sides(self): <side order>
+ get_drinks(self): <drink order>