

## Dokumentacja wstępna projektu „Symulator Restauracji”

Postanowiliśmy zaprojektować symulator restauracji w języku C++. Będziemy realizować zadanie w oparciu o podejście obiektowe. Poniżej przedstawimy klasy składające się na nasz projekt.

Restaurant
-name: string -waiter: Waiter -menu: Menu -tables: map<uint, Table> -kitchen: Kitchen
+get_name() +get_waiter() +get_menu() +get_tables() +add_table(Table) +remove_table(Table) +get_table_by_id(uint) +remove_client(Client) +get_kichen()

W restauracji znajdują się kuchnia oraz stoliki. Za tworzenie dań odpowiedzialna jest kuchnia. Gotowe dania roznosi kelner.

Menu
-name: string -menu_sections: list<MenuSection>
+get_name() +set_name(string) +get_menu_sections() +add_menu_section(MenuSection) +remove_menu_section(MenuSection) +get_dish_by_name(string) +add_dish_to_menu_section(MenuDish, string) +get_menu_section_by_name(string) +is_dish_in_menu(string) +is_menu_section_in_menu(string)

Menu dzieli się na sekcje. Do obiektów tej klasy mamy możliwość dodania i usuwania dań oraz sekcji, a także wyszukiwanie po nazwie.

Client
-ready_to_order: bool -ready_for_receipt: bool -eating_time: time -chosen_dishes: vector<MenuItem>.
+get_ready_to_order() +switch_ready_to_order() +get_ready_for_receipt() +switch_ready_for_receipt() +eating() +choose_dish()

Klienci mogą zgłosić gotowość do złożenia zamówienia i otrzymania rachunku. Posiadają oni wektor dań, które wybrali. Po otrzymaniu zamówienia spożywają posiłek zgodnie z czasem zapisanym w zmiennej „eating\_time”.

Table
-number_of_seats: uint -is_occupied: bool -ready_to_order: bool -id: uint -ready_for_receipt: bool -clients: vector<Client> -receipt: Receipt
+get_number_of_seats() +set_number_of_seats(uint) +get_is_occupied() +switch_is_occupied() +get_ready_to_order() +is_ready_to_order() +get_id() +set_id() +get_ready_for_receipt() +is_ready_for_receipt()

Receipt
-total_price: int -order: Order
+get_total_price() +show_receipt() +get_order()

Order
-table_id: table_id -ordered_dishes: list<Dish> -is_ready: bool
+get_table_id() +get_ordered_dishes() +switch_is_ready() +get_is_ready()

Stół ma określoną liczbę miejsc, unikalne id oraz flagi informujące czy przy stole ktoś siedzi, czy klienci przy nim siedzący są gotowi do złożenia zamówienia i do zapłaty. Do każdego stołu jest z góry przypisany rachunek. Kelner przy odbieraniu zamówienia tworzy obiekt klasy Order i dodaje do kolekcji ordered\_dishes zamówione przez klientów dania

Kitchen
-ready_dishes: list<Dish> -ready_orders: list<ReadyOrder> -restaurant: Restaurant -orders: list<Order>
+add_to(Vector, KitchenDish) +remove_from(Vector, KitchenDish)

Dish
-time_to_make: time_to_make -is_ready: bool -name: dish_name
+get_time_to_make() +get_is_ready() +switch_is_ready() +get_name()

Kelner po odebraniu zamówienia od klientów dodaje je na listę orders. Następnie kuchnia pobiera jedno zamówienie, wykonuje wszystkie występujące w nim dania i umieszcza je na liście ready\_dishes. Gdy wszystkie dania z danego zamówienia będą już gotowe, to zamówienie przeskakuje z listy orders do listy ready\_orders, z której odbiera je kelner i znosi do stolika.

Waiter
-serviced_tables: set<table_id> -restaurant: Restaurant -is_busy: bool
+get_free_table() +place_at_table(Table, Client) +search_ready_orders() +look_for_action() +give_receipt(Table) +get_serviced_tables() +switch_is_busy() +get_is_busy()

Kelner posiada mapę stołów które aktualnie obsługuje oraz flagę informującą czy jest aktualnie zajęty. Jego zadaniem jest odbieranie zamówień od klientów, zanoszenie gotowych dań oraz wydawanie rachunku.

## Diagram projektu:

