Restaurant Manager

Barak Zan 305634487

Arik Rinberg 317156669

1. Introduction
    a. The system "Restaurant Manager" is an application designed to assist managing a restaurant with hundreds of customers every day, and a large menu.
    b. The acting manager, Yossi Salat, wants a system that will help him manage the restaurant by providing the following:
        i. Easy to use digital menu for quick and accurate order time, including different types of the same dish.
        ii. Streamline communication between waiters and food preparation.
        iii. Enable customers to split the bill easily providing different payment methods.
        iv. Provide statistical information about the restaurant's performance for future advancements.
2. Functional requirements
    a. Initialize/change restaurant
        i. Add/remove table
        ii. Add/remove dish
            1. Add/remove dish extras
        iii. Add/remove stations
        iv. Alter number of waiters
    b. Save/load restaurant parameters
        i. Save/load the current status of:
            1. Tables
            2. Menu
            3. Stations
            4. Number of waiters
    c. Seat new customers
        i. When a group of customers arrive, a waiter accepts the group and gives them a table. The waiter records the number of guests and time of arrival for statistical analysis.
    d. Take order
        i. Take the order from each customer at the table creates a new unique order in system tied to the table.
        ii. Issue tasks to relevant stations, and notify the stations.
        iii. Record the orders.
    e. Change order
        i. If the order hasn't been delivered to the table, allow the customer to change the order.
        ii. Issue updates to relevant stations and notify the stations.
    f. Cancel order
        i. If the order hasn't been delivered to the table, allow the customer to cancel the order.

ii. Notify the relevant stations about the cancellation.

iii. Record that the order has been canceled.

g. Prepare dish

i. When a station receives a new order or order change, begin preparing the dish, and mark the dish as being prepared.

ii. When a station receives an order cancellation, cancel the order.

iii. When a dish is prepared, mark it as ready. If this is the last dish to be prepared mark the order as ready, and notify the waiters.

h. Supply order

i. When an order is marked as ready, take it to the table. The order is marked as delivered.

ii. Record the time of delivery.

i. Finish meal

i. The system prints a receipt for the table for the relevant order, and deliver to the table.

ii. Record the time of delivery of the receipt.

j. Take payment

i. Each customer may pay for himself or for the whole table.

ii. The customer can pay in three ways:

1. By cash

2. Credit card

a. Verify the card with credit card company

3. Application $eatWithIt$.

iii. Print receipt if there was no use of the application $eatWithIt$.

iv. Record the payment methods and amounts.

k. Get analytical report

i. Manager requests analytics from database

ii. Relevant data is queried and supplied to manager

3. Non-functional requirements

a. Performance

i. The system should be able to scale the amount of waiters/dishes/tables without effecting customer satisfaction

b. Portability

i. The system should be able to apply to any different restaurant requirements.

4. System evolution

a. Require minimum charge for certain payments.

b. Allow additional payment methods.

c. Allow customer built dishes.

d. Allow customer to share dishes to social applications.

e. Allows for cancelled dish that has been prepared to be reassigned to a different order when possible.

Question 2

Part 1

Actors

1. Manager
   Modify the state of the system, for example number of tables. Can view statistical data. The manager is allowed to perform any operation.
2. Waiter
   Can assign customers to a table, and create and alter an order. Is allowed to take payment, and serve ready orders. Can alter the order's state, for example mark it as served.
3. Station worker (for example, bartender)
   Receives dish orders and makes the dishes. Can alter the dishes state, for example mark the dish as ready.

Use cases

1. Seat new customers:
   When customers arrive, a waiter assigns the customers to a table and records relevant information.
2. Take order:
   A waiter creates a new order in the system. Order dish tasks are sent to the relevant stations.
3. Change order:
   A waiter alters an existing order in the system that isn't marked as delivered. The relevant stations are notified.
4. Cancel order
   A waiter cancels an existing order in the system that isn't marked as delivered. The relevant stations are notified and records relevant information.
5. Prepare dish
   A station worker receives an order dish task. Mark the dish task as being prepared and prepare it. Once ready mark the dish task as ready.
6. Supply order
   Once an order is marked as ready, the waiter delivers the order to the table and marks it as delivered and records relevant information.
7. Finish meal
   When the customers have finished the meal, the waiter supplies the receipt and records relevant information.
8. Take payment
   When the customers are ready to pay, the waiter takes payment and records relevant information.
9. Get statistical data
   The manager queries the statistical data. The data is extracted from the system logs, analyzed, and returned to the manager.
10. Modify system state
    The manager modifies the system state.

Part 2

Take order use case

Actors: waiters, station workers

Goal description: The customer's order is added to the system.

Reference to requirement document: 2.d

Preconditions: The customers are seated at a table.

Description:

1. The waiter gets the order from each customer.
2. Waiter creates an order, and adds all the dishes.
3. The system initializes all dishes
    a. Marks them all as not ready
    b. Alerts the stations
4. System updates analytics DB

Postcondition:

Success – The order was added and recorded

Failure – No order was added

Variations: None

Exceptions:

1. Network failure: failure
2. Power cut: failure
3. Waiter/client have heart attack: failure

Payment analytics use case

Actors: manager

Goal description: Create analytical report on payment method and amounts for manager

Reference to requirement document: 2.k

Preconditions: Manager is logged in.

Description:

1. Manager queries analytics DB
2. The system gathers the information
3. The system creates the report
4. The system displays the report for the manager

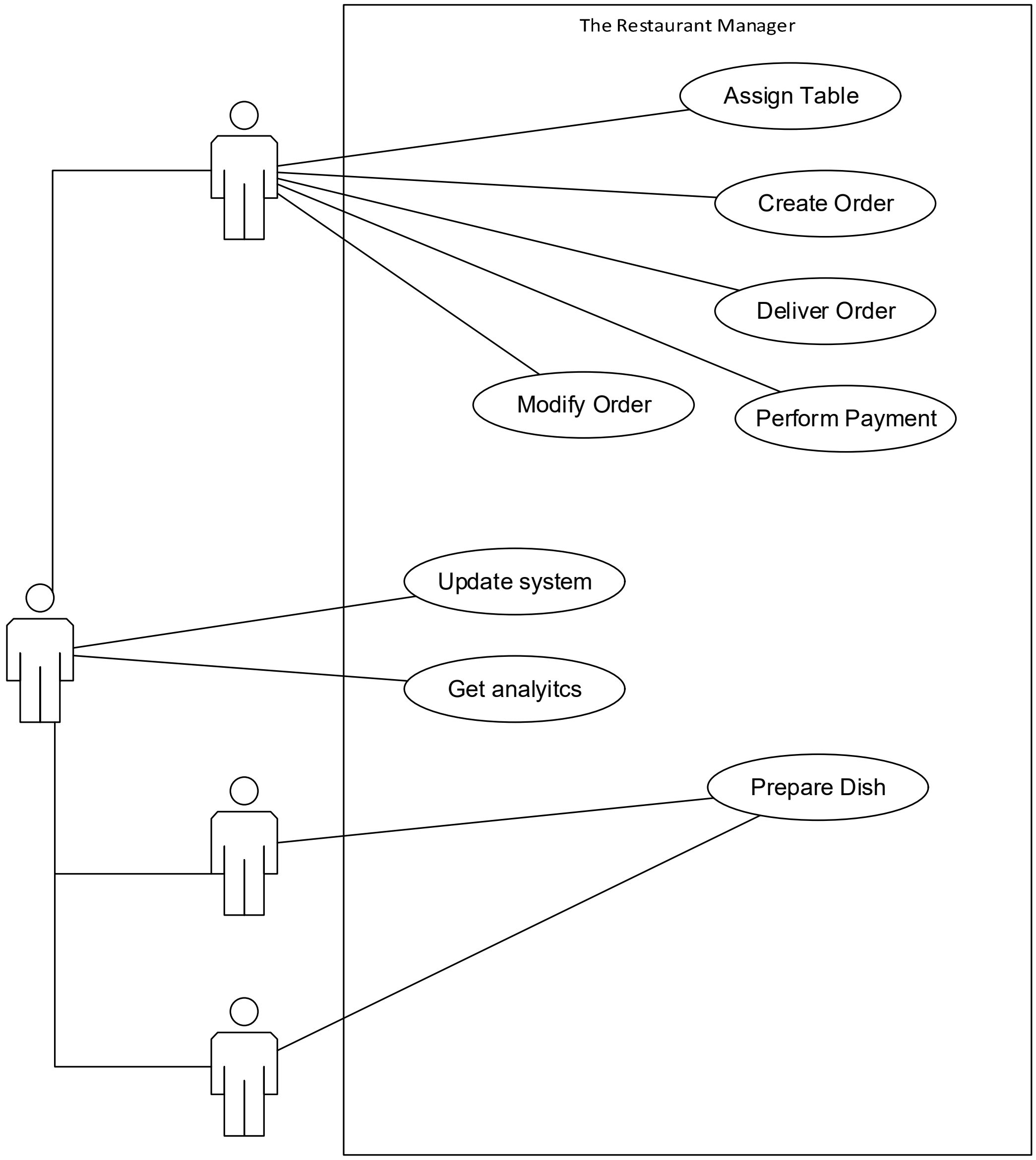
Postcondition: None

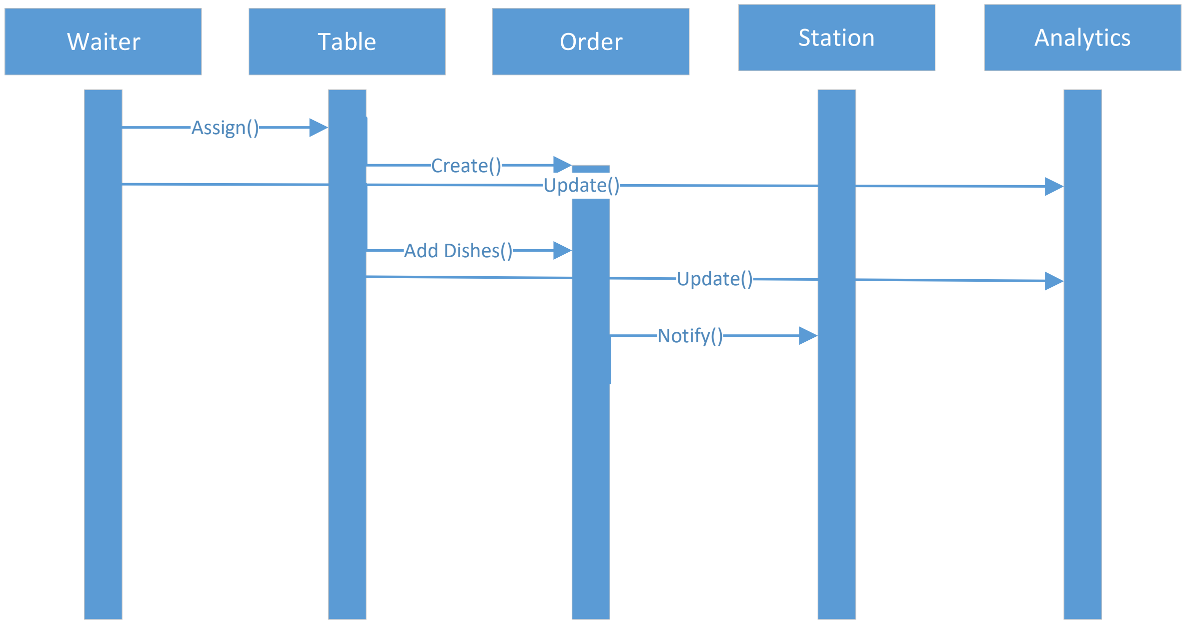
Variations: No relevant data - success

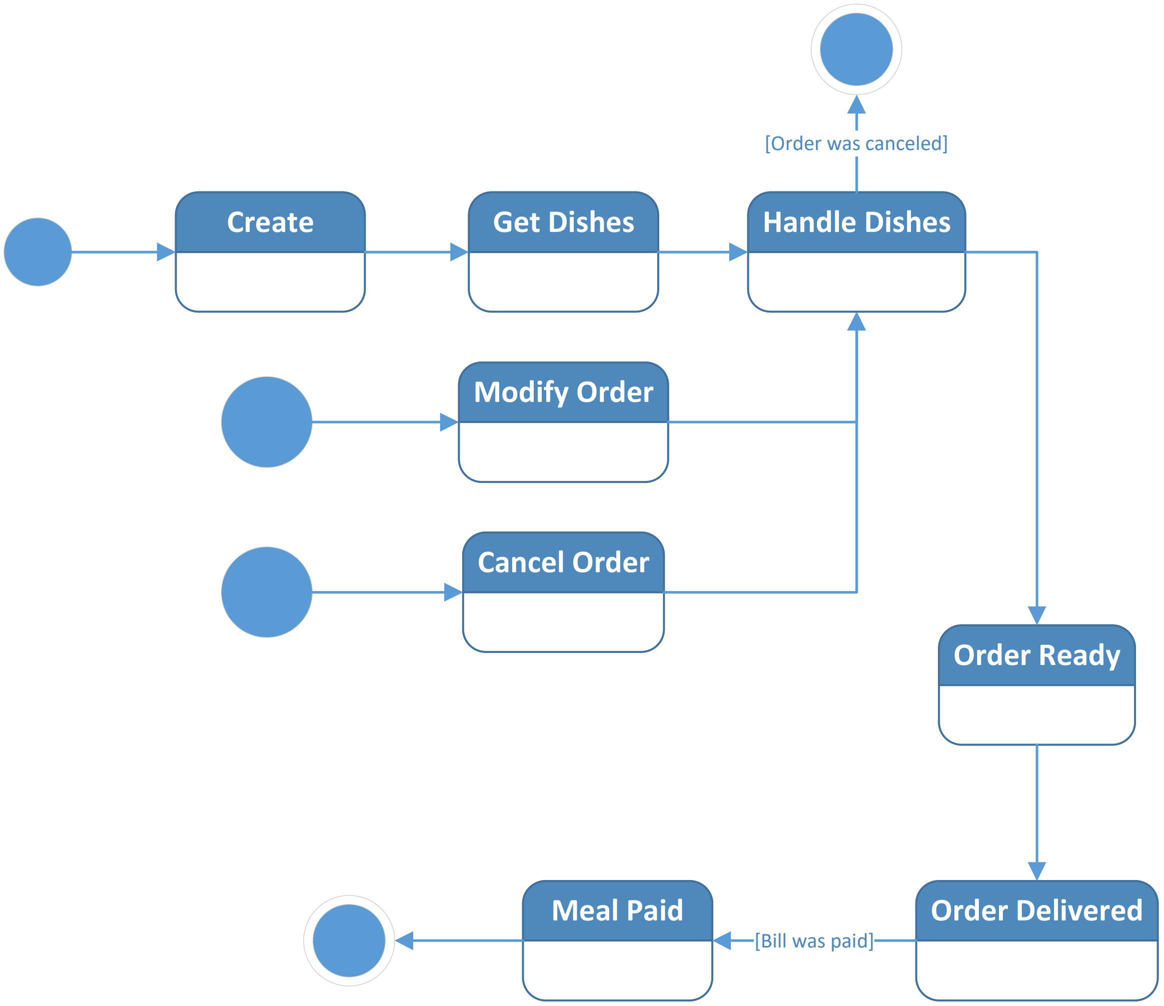
Exceptions:

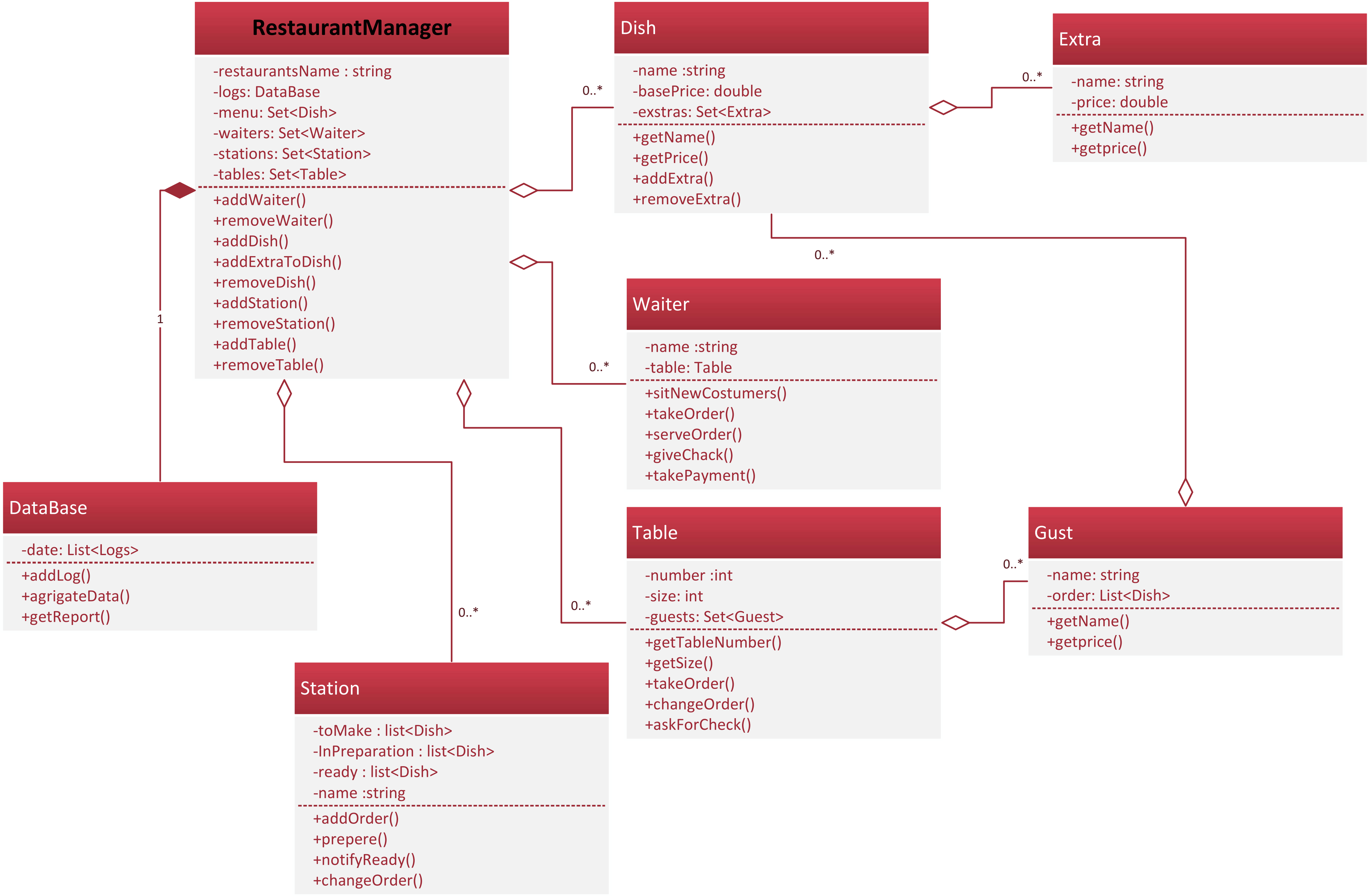
1. Network failure: failure
2. Power cut: failure

Part 3




Waiter
Table
Order
Station
Analytics
Assign()
Create()
Update()
Add Dishes()
Update()
Notify()


[Order was canceled]
Create
Get Dishes
Handle Dishes
Modify Order
Cancel Order
Order Ready
Order Delivered
[Bill was paid]
Meal Paid


RestaurantManager
-restaurantsName : string
-logs: DataBase
-menu: Set<Dish>
-waiters: Set<Waiter>
-stations: Set<Station>
-tables: Set<Table>
+addWaiter()
+removeWaiter()
+addDish()
+addExtraToDish()
+removeDish()
+addStation()
+removeStation()
+addTable()
+removeTable()
Dish
-name :string
-basePrice: double
-exstras: Set<Extra>
+getName()
+getPrice()
+addExtra()
+removeExtra()
Extra
-name: string
-price: double
+getName()
+getprice()
0..*
0..*
0..*
1
Waiter
-name :string
-table: Table
+sitNewCostumers()
+takeOrder()
+serveOrder()
+giveChack()
+takePayment()
0..*
DataBase
-date: List<Logs>
+addLog()
+agrigateData()
+getReport()
Table
-number :int
-size: int
-guests: Set<Guest>
+getTableNumber()
+getSize()
+takeOrder()
+changeOrder()
+askForCheck()
Gust
-name: string
-order: List<Dish>
+getName()
+getprice()
0..*
0..*
0..*
Station
-toMake : list<Dish>
-InPreparation : list<Dish>
-ready : list<Dish>
-name :string
+addOrder()
+prepere()
+notifyReady()
+changeOrder()