# Startdocument for 'The catering company (Assignment 9)'

Startdocument of **Michel Disbergen**. Student number **4999738**.

## Problem Description

A company provides catering for people. For the period indicated with a start date and an end date, the administration for a number of customers are processed. One day a customer can order an appetizer (3 euros) and or a main course (5 euros) and or a dessert (2 euros). A program should be developed that enables each customer to name, number of starters, number of main courses and number of desserts is stored that the customer has had. Ultimately, the company wants to presented the following information:

- The total revenue;
- The average turnover per day;
- The name of the customer who has the highest amount consumed;
- The name of the customers who have had appetizers but never had one dessert;
- The names of the customers who for an amount higher than the average per customer.

### Input & Output

In this section the in- and output of the application will be described.

#### Input

In the tables below all the input (that the user has to input in order to make the application work) are described.

##### CateringCompany

| Case | Type | Conditions | function/method | additional info |
|---|---|---|---|---|
| String name | `String` | not empty | constructor, setter | name of a dish |
| List assignments | `List<Assignement>` | - | setter | Assigntments a catering company has |

##### Assignment

| Case | Type | Conditions | function/method | additional info |
|---|---|---|---|---|
| String name | `String` | not empty | constructor, setter | name of the catering company |
| DateTime startDate | `DateTime` | full date, including year, month, day and time. | constructor, setter | day the catering company starts taking care of their customers which are related to a company |
| DateTime endDate | `DateTime` | full date, including year, month, day and time. | constructor, setter | day the catering company stops taking care of their customers which are related to a company |
| List customers | `List<Customer>` | - | setter | all customers who have pruchased an appetizer, main course or dessert at the catering company |
| List availableDishes | `List<Dish>` | - | setter | all dishes that the catering company has available for selling. |

##### Dish

| Case | Type | Conditions | function/method | additional info |
|---|---|---|---|---|
| float price | `float` | price > 0 | constructor, setter | price of a dish |
| String name | `String` | not empty | constructor, setter | name of a dish |
| boolean appetizer | `boolean` | true or false | constructor, setter | specifies if the dish is an appetizer. If true, mainCourse and dessert must be false |
| boolean mainCourse | `boolean` | true or false | constructor, setter | specifies if the dish is a maincourse. If true, appetizer and dessert must be false |
| boolean dessert | `boolean` | true or false | constructor, setter | specifies if the dish is a dessert. If true, maincourse and appetizer must be false |

##### Customer

| Case | Type | Conditions | function/method | additional info |
|---|---|---|---|---|
| String name | `String` | not empty | constructor, setter | name of a customer |
| float wallet | `float` | - | constructor, setter | the amount of money a customer has in his/her wallet |
| List orderedDishes | `List<Dish>` | - | setter | a list of all the dishes that a customer has consumed from the catering company |

#### Output

##### CateringCompany

| Case | Type | function/method |
|---|---|---|
| Get the name of a catering company | `String` | Name() |
| Get the list of assigntments a cateringCompany has | `List<Assigment>` | Assignments() |

##### Assignment

| Case | Type | function/method |
|---|---|---|
| Get the name of the catering company | `String` | Name() |
| Get the list of customers | `List<Customer>` | Customers() |
| Get all available dishes | `List<Dish>` | AvailableDishes() |
| Get the startdate of when the catering company is taking care of the catering | `DateTime` | StartDate() |
| Get the enddate of when the catering company is taking care of the catering | `DateTime` | EndDate() |

| Case | Type | function/method |
|---|---|---|
| Get the entire revenue the catering company has made | `float` | calculateRevenue() |
| Calculate the average revenue over a specific date | `float` | calculateAverageRevenue() |
| Retrieve the biggest consumer in customer terms | `Customer` | getBiggestConsumer() |
| Get all customers that have had appetizers, but no desserts | `List<Customer>` | getAppetizerCustomer() |
| Get all the customers that have spend money on dishes which is above average | `List<Customer>` | getAboveAverageCustomers() |
| Once a customer orders a dish, sell that dish to the customer | `Dish` | sellDish() |

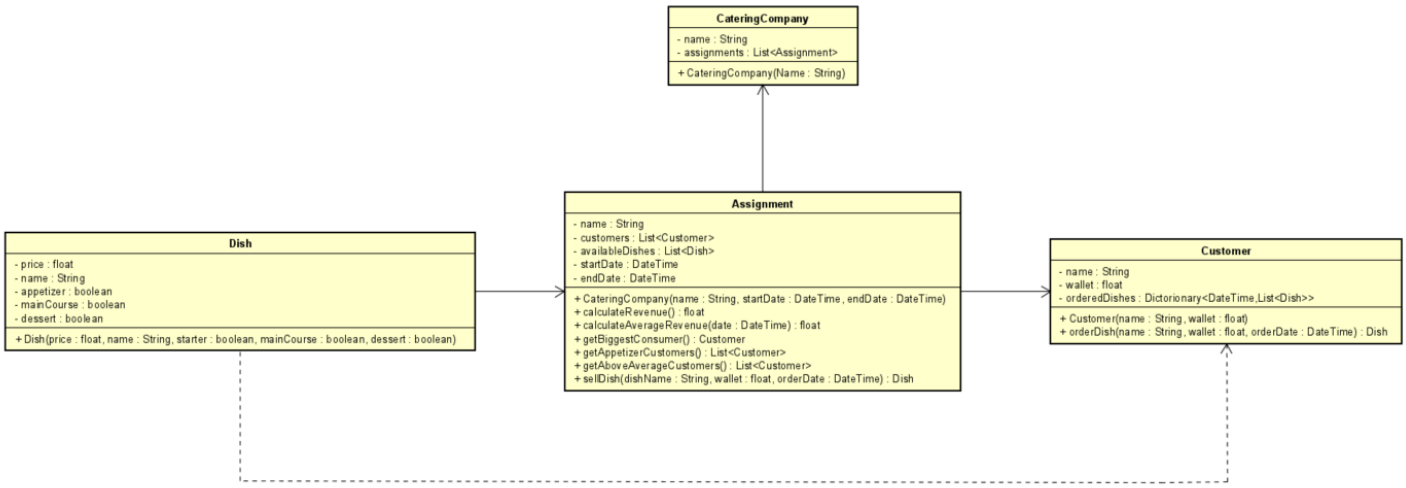
#### Dish

| Case | Type | function/method |
|---|---|---|
| Get the price of a dish | `float` | Price() |
| Get the name of a dish | `String` | Name() |
| Get the result if the dish is a appetizer | `Boolean` | Appetizer() |
| Get the result if the dish is a maincourse | `Boolean` | MainCourse() |
| Get the result if the dish is a dessert | `Boolean` | Dessert() |

#### Customer

| Case | Type | function/method |
|---|---|---|
| Get the name of a customer | `String` | Name() |
| Get the amount of money a customer has in his wallet | `float` | Wallet() |
| Retrieve all dishes that a specific customer has ordered for a specific date | `List<Dish>` | OrderedDishes() |
| Order a dish as a customer | `Dish` | orderDish() |

## Class Diagram




## Testplan

### Customer

#### Ordering a dish

| Customer | Dish name | Cash | Date | Cash after ordering |
|---|---|---|---|---|
| Willem | - Chateau briand<br>- Tomatensoep | 20.00 | 4-6-2022 | 15.00 |
| Myrthe | - Tomatensoep | 10.00 | 4-6-2022 | 8.00 |
| Michel | - Pavlova | 10.00 | 4-6-2022 | 8.00 |

### Assignment

|Revenue total |Revenue day specific |Biggest consumer | Appetizer customers | Customers that ordered above average | |-|-|-|-|-| **9** : `input : button` | **9** : `input : 4-6-2022` | **Willem** `input : button` | **Myrthe** `input : button` | **Willem** `input : button` |