PARIS PANTHÉON-ASSAS UNIVERSITÉ

Project – MOM

Integration System - Application

Nathan DOLY
Sylvain MIGEON
Promotion 2024 – DT Group

M. ALLAM Diana

2022 – 2023

Summary

UML

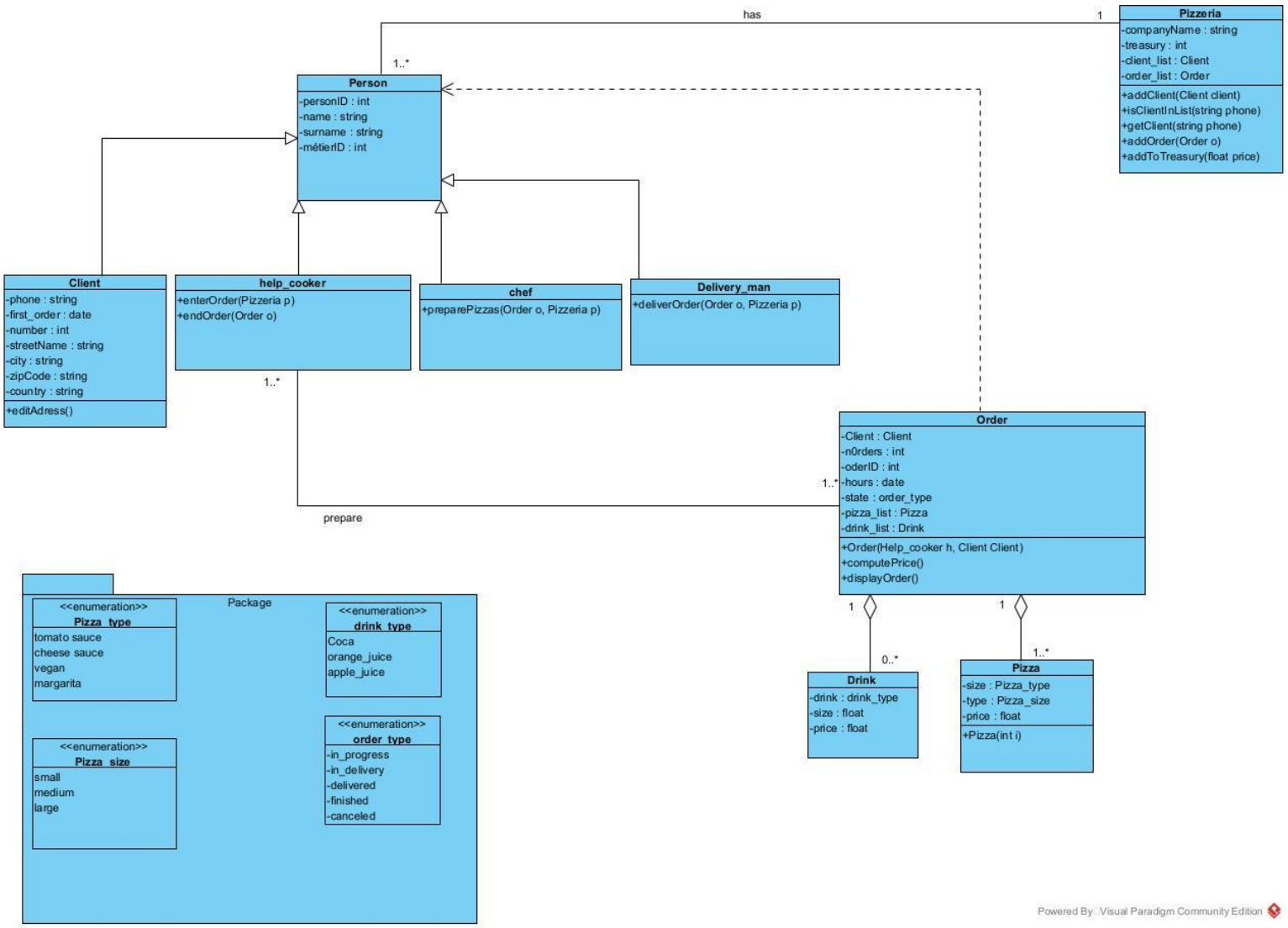
In this UML schema, we have 3 main entities which are Pizzeria, Person and Order.

Pizzeria contains all the attributes necessary to his identification but also the different lists such as the client list and the order list.

Person contains all the basics attributes, but this entity is the base class. Client, help_cooker, chef and Delivery_man inherit from Person. Each entity according to their name will have their own methods.

Order contains the attributes and there are aggregations for pizza and pizza because they are part of the order.

And finally, different enumerations to assign constant values to make it easier to put state on the different objects.



Code

Program.cs – Launching of the program

```
static async Task Main(string[] args)
{
    Console.Clear();

    Pizzeria pizzeria = new Pizzeria();

    Client c = new Client(1, "John", "Doe", 1, "1234567890", DateTime.Now, 1, "rue de la paix", "Paris", "75000", "France");

    pizzeria.addClient(c);

    //set h as Help_Cooker
    pizzeria.Help_cooker = new Help_cooker(9, "Pierre", "Chabrier", 2);

    // set chef to pizzeria
    pizzeria.Chef = new Chef(1, "Jean", "Dupont", 3);

    //set delivery_man to pizzeria
    pizzeria.Delivery_Man = new Delivery_man();

    // Run the Task
    Task t1 = Task.Run(() => pizzeria.Help_cooker.enterOrder(pizzeria));
    await t1;
}
```

In this main program we just have instantiate some classes such as the pizzeria, a client (here just to have at least one object in the client list), a help cooker, a chef, and a deliver person.

Pizzeria – Contain all data

```
class Pizzeria
{
    // ATTRIBUTES
    2 references
    private string companyName = "O'pizza";
    4 references
    private float treasury = 0;

    4 references
    private List<Client> client_list = new List<Client>();
    5 references
    private List<Order> order_list = new List<Order>();
```

```
public bool isClientInList(string phone) {
    foreach(Client c in this.client_list) {
        if(c.Phone == phone) {
            Console.WriteLine("find " + c.Name + c.Surname);
            return true;
        }
    }
    return false;
}
```

The pizzeria class is the one of the important classes because it is our data storage. In this class we have our client list and our order list. These lists will be use through all the programs.

Then the **isClientInlist** which is not asynchronous but it the start of our scenario. In this function, we verify if a client phone is associate to a client and according to the result different thing will happen.

Help_cooker

```
Console.WriteLine("<======Take the order======>");
Order order = new Order(this, c);
p.addOrder(order); // Add the order to the list of orders

//Order sent to the Chef
p.Chef.preparePizzas(order, p);

//Take another client's order
Task t2 = Task.Run(() => p.Help_cooker.enterOrder(p));
t2.Wait();
```

In help_cooker, we have a function **enterOrder** which will simply launch the order constructor and take the order of the client, add it to the order_list and send it to the chef. While this the help cooker will be able to take another order to another client while the chef and deliver execute their tasks.

NB: In this part we suppose the client already exists. According to the response a complementary step will occur.

Chef

```
public void preparePizzas(Order o, Pizzeria pizzeria) {
    string pizza_list_string = "";

    // Check if the order has been properly taken by the help_cooker (in_pro
    if(o.State == order_type.in_progress){
        Console.WriteLine("Receiving the pizza of the order " + o.OrderID);

        // Run a task for each Pizza in the Order to prepare it asynchronous
        foreach(Pizza pizza in o.Pizza_list) {
            pizza_list_string += pizza.pizza_Type + " ";
            Task.Run( () => makingPizza(pizza, o));
        }

    }
    // Once all the pizzas are prepared, the order is ready to be delivered
    o.State = order_type.in_delivery;
    pizzeria.Delivery_Man.deliverOrder(o, pizzeria);
}
```

Once the help cooker sends the order to the chef, we will check the state of the order then start pizza preparation. For each pizza of the order, we start a new task asynchronously.

Execution

Help cooker

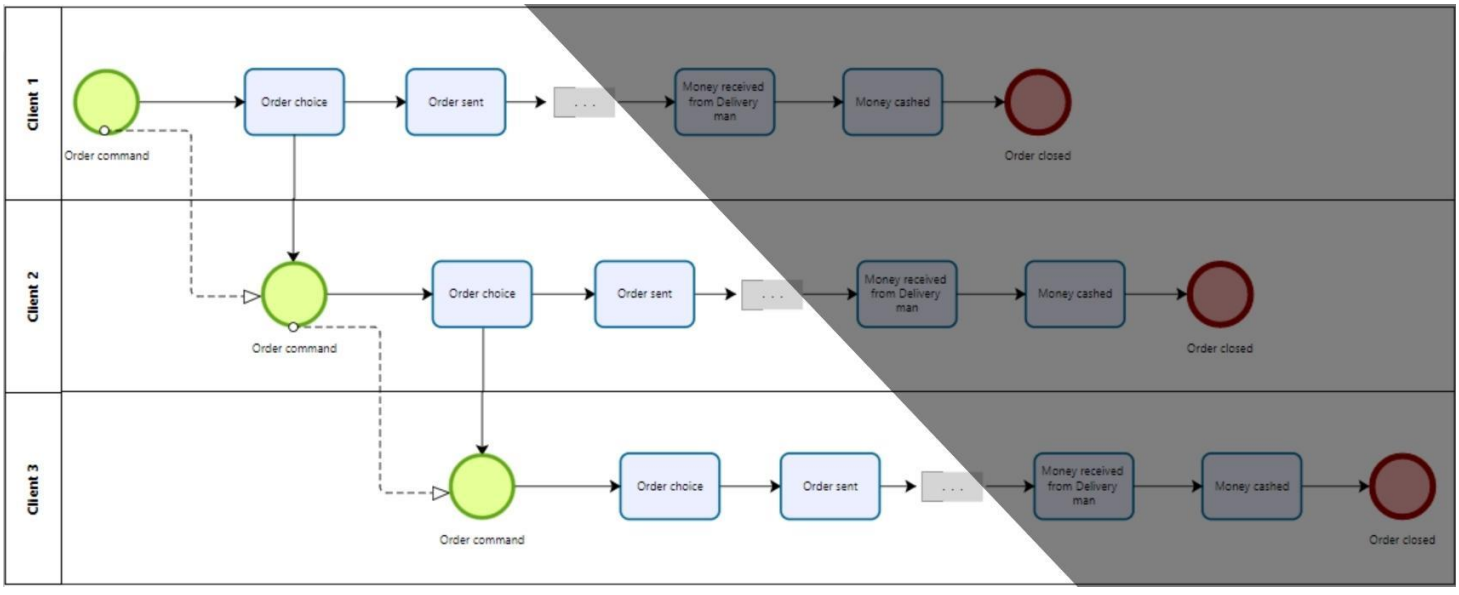
To begin the process of pizza ordering, the help_cooker (the person who answer at the client's call) ask to the client if it is his first-time ordering at the Pizzeria. Depending on the answer it will either use the client data from the List of Client from the pizzeria or create a new Client and adding it to the List of Client.

Then the client had to confirm his address, if the address is correct the process continues if not, all the information of the client is collected to create a new Client to add to the List of Client.

After that, if the client is ready to order, the help cooker will take his command then take the order of another client. If the client is not ready to command, the help cooker will directly begin a new process of ordering with another client.

```
-----------------------------------
Is it the first time you order? (y/n)
-----------------------------------
n
Not the first time
Verification of the phone number
1234567890
find JohnDoe
Client is in the list
-------------------------
Is it your address ? [y/n]
-------------------------
y
Good Address
-------------------------
Are you ready to command ?
-------------------------
Enter [y/n]
y
```

The orders are taken asynchronously (see the schema below).



Help cooker - Executive Schema

Legend :

——→ Classical process – The client is ready to order

- - - - - - - → Fast process – The Client is not ready to order

Still with the help cooker, he will take the order of the client beginning by the pizza choice then the drink choice.

```
<======Take the order======>
------------
Order n°1
------------
How many pizzas ?
Enter a number [max 5]
2
<======= PIZZA CHOICE =======>
Which pizza do you want :
--------------------
1 margherita  [5?]
2 tomato      [7?]
3 cheese      [9?]
4 vegetarian  [11?]
--------------------
1
Which size :
-------------
1 Small  [1?]
2 Medium [2?]
3 Large  [3?]
-------------
1
Which pizza do you want :
--------------------
1 margherita  [5?]
2 tomato      [7?]
3 cheese      [9?]
4 vegetarian  [11?]
--------------------
3
Which size :
-------------
1 Small  [1?]
2 Medium [2?]
3 Large  [3?]
-------------
3
```

```
How many drinks ?
Enter a number [max 5]
1
<======= DRINK CHOICE =======>
Which drink do you want :
--------------------
1 Coca-Cola     [1?]
2 Orange juice  [2?]
3 Apple juice   [3?]
--------------------
1
Volume :
-------------
1 25cL  [1?]
2 33cL  [2?]
-------------
2
---------------------------------
Order take at 22/10/2022 10:22:42
---------------------------------
```

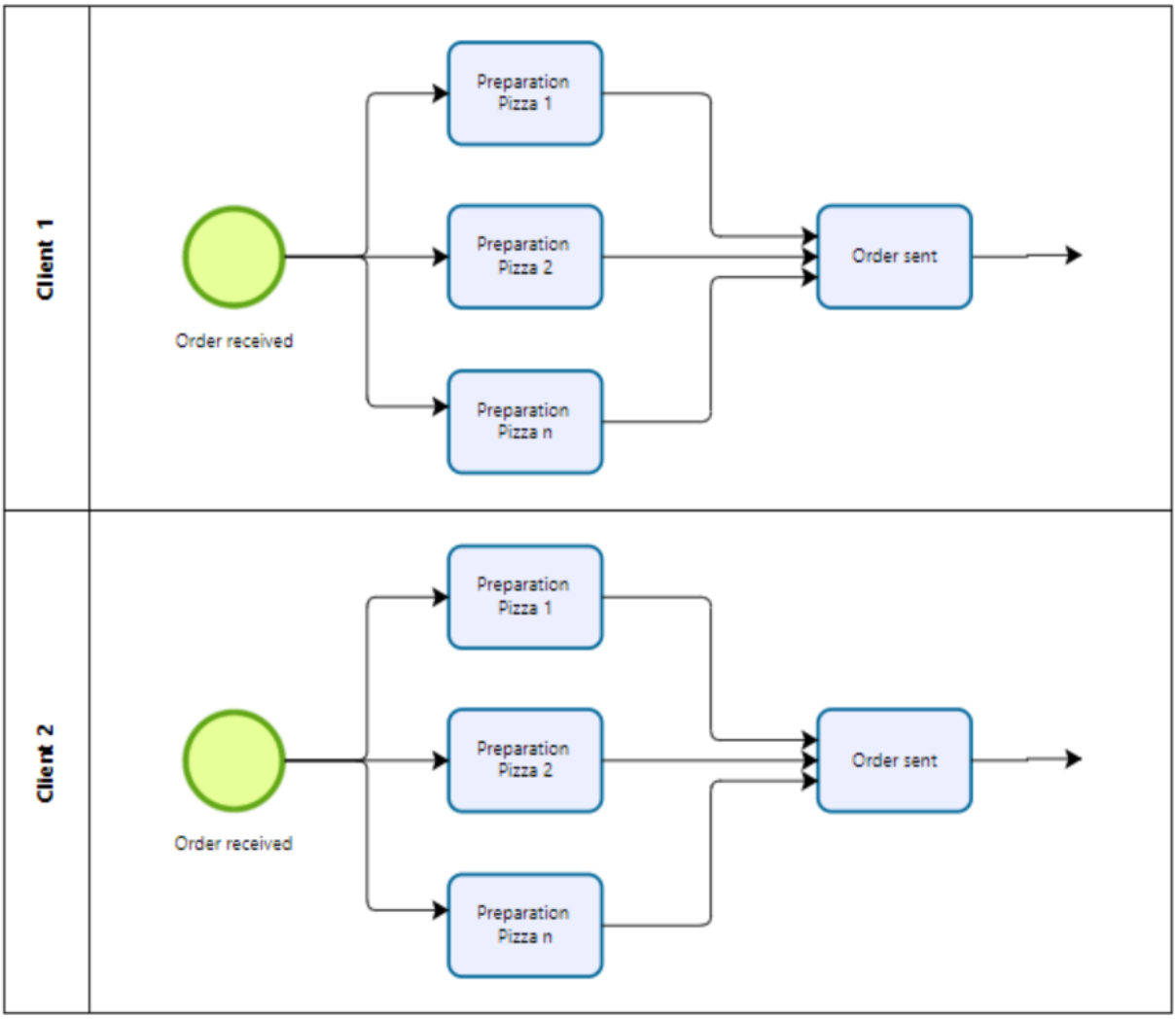
Chef

Once the order is taken by the help cooker, the chef will prepare asynchronously the pizza.

The pizza time cooking depends on the size of the pizza. (See the schema below)

When all the pizza are ready, there is an announcement.

```
State of the order : in progress
Receiving the pizza of the order 1
Making the pizza cheese_sauce
Making the pizza margherita
-----------------------------------
Is it the first time you order? (y/n)
-----------------------------------
Pizza margherita is ready
Pizza cheese_sauce is ready
-----------------------------------
 Order 1 - All the pizza are ready !
-----------------------------------
```



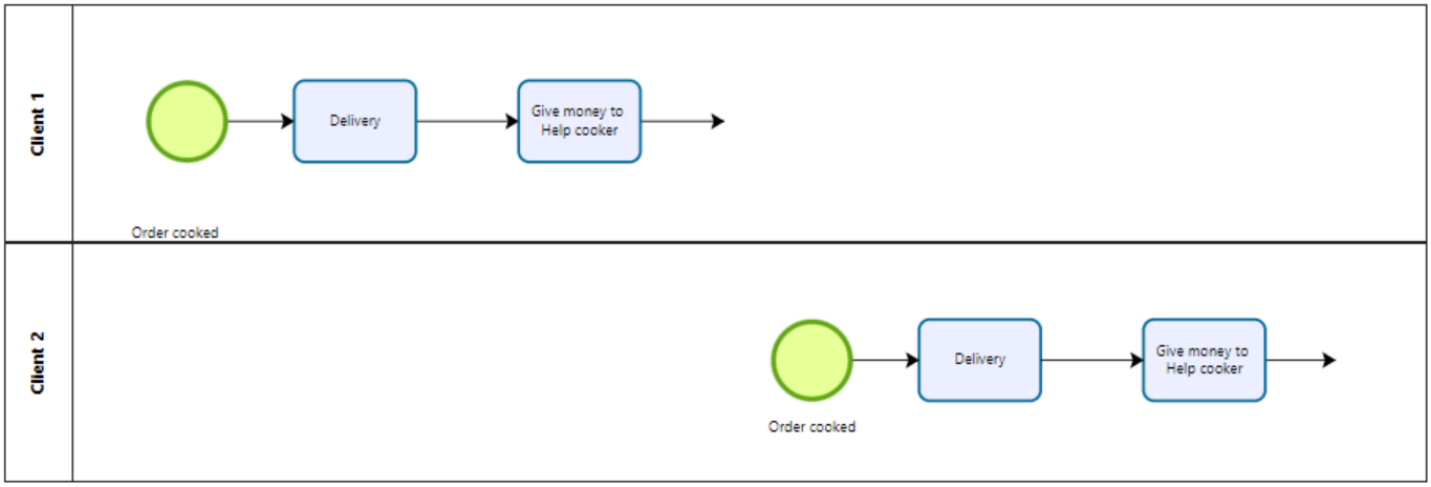
Chef - Executive Schema

Delivery man

Once all the pizza are prepared, the delivery man will deliver the pizza to the client and receive money from him. (see the schema below)

After that the delivery man give the money to the help cooker to end the order.

```
------------
Order in delivery
Sending the order 1 to the client
The address was found
------------------------------
 The order 1 is delivered
------------------------------
Deliver received : 20?
```



Delivery man - Executive Schema

Help cooker

Finally, the help cooker cashed the money receive by the delivery man. The money is directly add to the treasury.

The order process end with the help cooker who close the order and a summary of the order.

```
Money cashed : 20?
Treasury : 20?
------------------------
 Order 1 finished
------------------------
Order ID : 1
State of the order : finished
Pizzas ordered :
        pizza 1 : margherita
        pizza 2 : cheese_sauce
Drinks ordered :
        drink : coca
Please answer y or n
```