UML II assignment

Case: Pizzaria

In this assignment, you are going to **redesign** your **Domain model (if needed)**, **Design Class Diagram** and other **UML diagrams** from UML 1 assignment. You are going to implement a new version of the system to Big Mamma Pizzaria.

This version of the system should include:

- Customer administration (to be implemented)
  - The system should be able to make a list of Customers
  - The system should be able to create a customer, delete a customer and update a customer.
  - The system should be able to search for a Customer based on name.

- Pizza menu administration (to be implemented)
  - The system should be able to create a Pizza, delete a Pizza and update a Pizza.
  - The system should be able to print out on the screen a list of Pizzas.
  - The system should be able to search for a Pizza based on relevant and given criteria (Pizza no).

- User dialog (extra)
  - The system should be able to print out on the screen a menu
  - The system should handle errors in the input.

- Order administration (This is NOT part of this asignment)
  - The system should be able to create an order, delete an order and update an order.
  - The system should be able to search for an order based on relevant and given criteria.
  - The system should be able to make a list of Orders

1. User Stories	Update your Product Backlog with new or updated User Stories. The Product Backlog should contain US about orders, customers and more ● Add Acceptance Criteria
2. UML I - Domæne Model and Design Class Diagram	Update your diagrams from the assignment UML 1 regarding the pizza menu, orders and customers.
3. UML II - Sekvens Diagram	Create a sequence diagram for, what happened when you call the SearchPizza method.
4. Implementation	Implement the part of the system related to the customer administration (using a collection of type **List**) and to the pizza menu administration (using a collection of type **Dictionary**).

In parallel with the code, you should update your user stories and UMLdiagrams.

There will be variations in the implementation, depending on the design of your UML diagrams. You are free to implement your own system. There might be more steps than the steps below.

Implement all the 1:* and *:* associations/aggregations/compositions using collections classes. In each case, you should consider which collections class to use e.g., List, Dictionary or another.

Steps to take:

- Implement a class CustomerFile using List (You have a Customer class from UML I)
  - create a class CustomerFile that contains customers (stored in a List),
  - Implement methods for CRUD operations (Create, Read, Update and Delete).
  - Implement a method, PrintMenu, that can print out all the Customers to the screen.

- Implement a class PizzaMenu (You have a Pizza class from UML I)
  - Create a class PizzaMenu that contains pizzas (stored in a Dictionary (key=Pizza Number value = Pizza),
  - Implement methods for CRUD operations (Create, Read, Update and Delete).
  - Implement a method, PrintMenu, that can print out the menu card (to the screen).
  - Implement a method SearchPizza(criteria), that searches for a pizza and returns a Pizza object.

- In the class Store implement code to test the two new implemented classes (CustomerFile and PizzaMenu) and methods
  - Create an object of the CustomerFile
  - Show how to add, delete or update customers in the CustomerFile-object

  - Create an object of PizzaMenu
  - show how to add, delete or update Pizzas in the PizzaMenu -object.
  - printing out the menu (calls the PrintMenu)
  - show how to use the SearchPizza (You do **not** need to read input from the screen)

Extra assignments - If time permits……..

Create a Menu dialog on the console-screen

- Create a user dialog for the system. Create a method menu choice that prints out a list of menu items and reads the users choice. Eg. :

1. Add new to pizza to the menu
2. Delete pizza
3. Update pizza
4. Search pizza
5. Display pizza menu

   ……..

   The method could look like:

```
public int MenuChoice( List<string> menuItems) {
}
```

   The methods must handle errors.
   The method should return the user's choice as a number.

Use the implemented user dialog to test the system.

Really an extra-extra assignment

- Implement a class Order and consider how to represent the items in the order.
- Implement a method that makes it possible to add extra toppings to the pizza in the order.
- Implement the connection to the Customer object.
- Implement methods for CRUD operations for the Order.
- In the class Store implement code to test the implemented classes and methods in step 3.