The Singleton Pattern

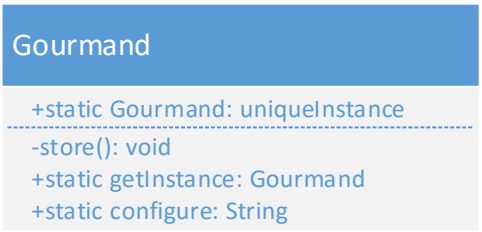
Object Oriented Goal: Ensuring only one instance of the singleton object available to all the classes.

Definition: The Singleton Pattern ensures a class has only one instance, and provides a global point of access to it.

Implemented on classes:

1. Gourmand

Design Snapshot:



Design Explanation:

The Gourmand class is instantiated only once because it is the class used to initiate the app. Thus, after initializing the variable for the first time, that same variable is used as an instance whenever it is required by any other object.

The Singleton Pattern makes a class singleton by providing only one static instance variable to all the objects who wants to access that object. Thus ensuring only one global object. It also defines one static public method which is used to access the unique instance. The implementation of Singleton Pattern in my design is explained as follows:

- The Gourmand class is the singleton class. The variable uniqueInstance is marked as a static variable and thus there is only one copy of that variable. That will be the class variable and anyone who wants to access the instance of Gourmand will get this global instance from the getInstance() method
- The Gourmand can be implemented as follows so as to make the class a singleton class:

```
Public class Gourmand
{
        //declaring the unique instance as a private and static method so that there
        remains only one instance of the class and no one can access the instance directly
        Private static Gourmand uniqueInstance;

        //making the constructor of the class as private, so no one can instantiate the
        class using the constructor
        Private Gourmand() {}
```

```
        //providing the single unique instance of the class by checking if the instance
        already exists or not
        Public static Gourmand getInstance()
        {
                If(uniqueInstance == null)
                        uniqueInstance = new Gourmand();

                //this will return the unique instance to whoever called this method
                return uniqueInstance;
        }
}
```