HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INFORMATION TECHNOLOGY

FINAL REPORT OOSD

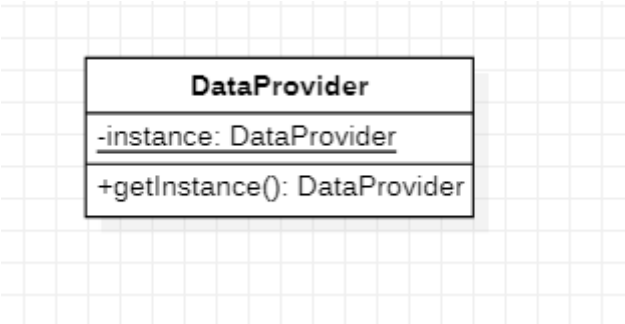
Object-oriented software design (design pattern)

Topic: BUILDING BOOKING TABLE APP
Subject: Object-oriented software design_02 CLC
Lecturer: Huỳnh Xuân Phụng
Group: 13
Members: 18110018 Lê Quang Huy
18110063 Nguyễn Thị Cẩm Tú
Link github:
https://github.com/DevLeeHuy/FinalProjectOOSD

Thành phố Hồ Chí Minh, thứ năm ngày 28 tháng 5 năm 2021

❖ Singleton

- **Create only one DataProvider for communicating with database.**
- **Why do you select this design pattern?**
  Because when interacting with data we have to create a DataProvider which can easily cause the problem that there are many connections at the same time so a singleton is needed to create only one DataProvider in the whole process.
- **The characteristic of this design pattern:**
  The Singleton pattern ensures that a class has only one instance and provides a global point of access to that instance. It is named after the singleton set, which is defined to be a set containing one element.
- **Class diagram:**



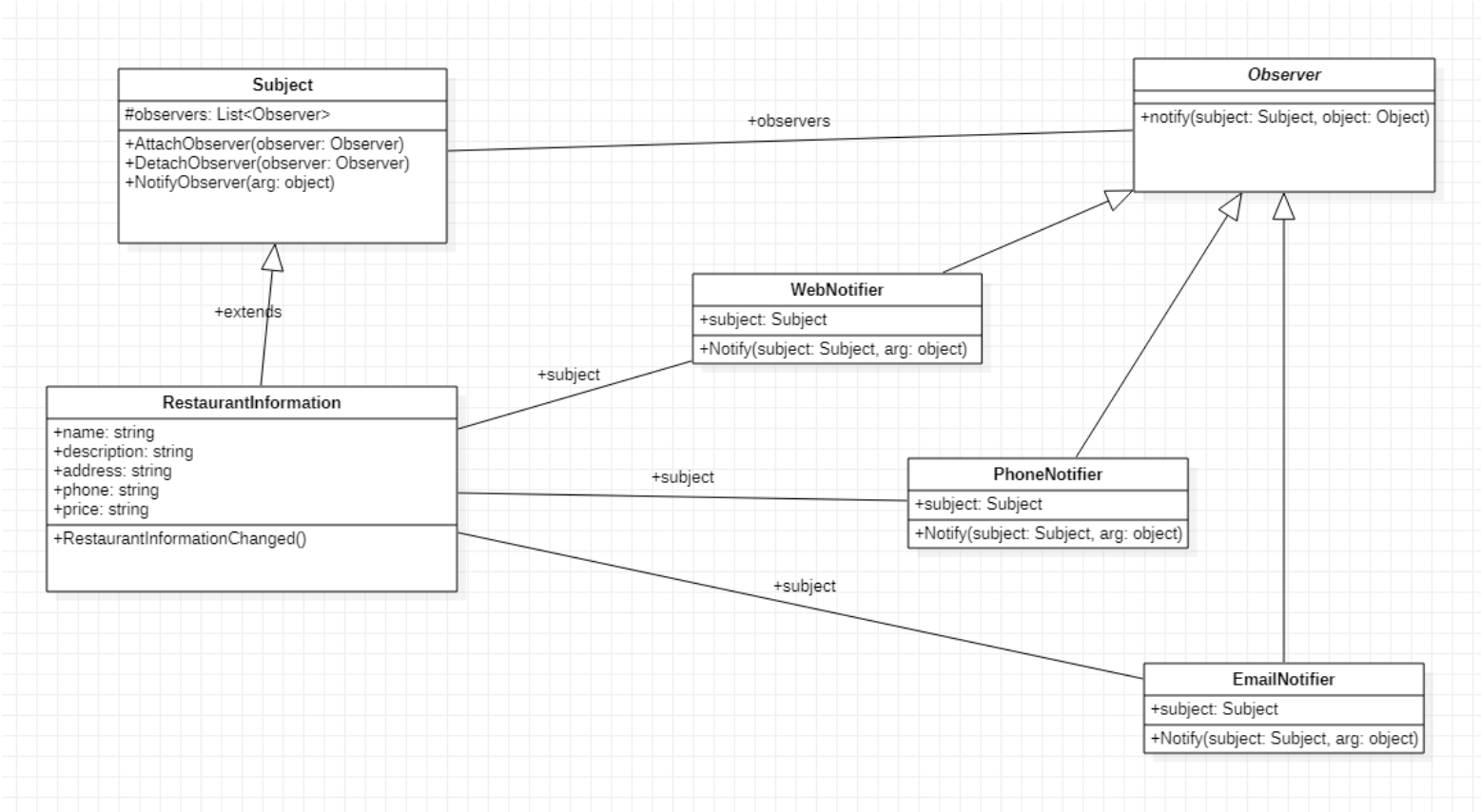
- **Code:**

https://github.com/DevLeeHuy/FinalProjectOOSD/blob/main/Design_Pattern/SingletonPattern/Program.cs

❖ Observer

- **Create an Observer Notifier for notifying when some information of a restaurant was changed to diners by many notifier (such as email, facebook, website,…).**
- **Why do you select this design pattern?**
  This observer gives us more flexibility in switching notification states for diners and restaurant changes through the notifiers Can add or remove notifiers dynamically as shown in the code example.
- **The characteristic of this design pattern:**
  The observer pattern is a software design pattern in which an object, named the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods.

- **Class diagram:**

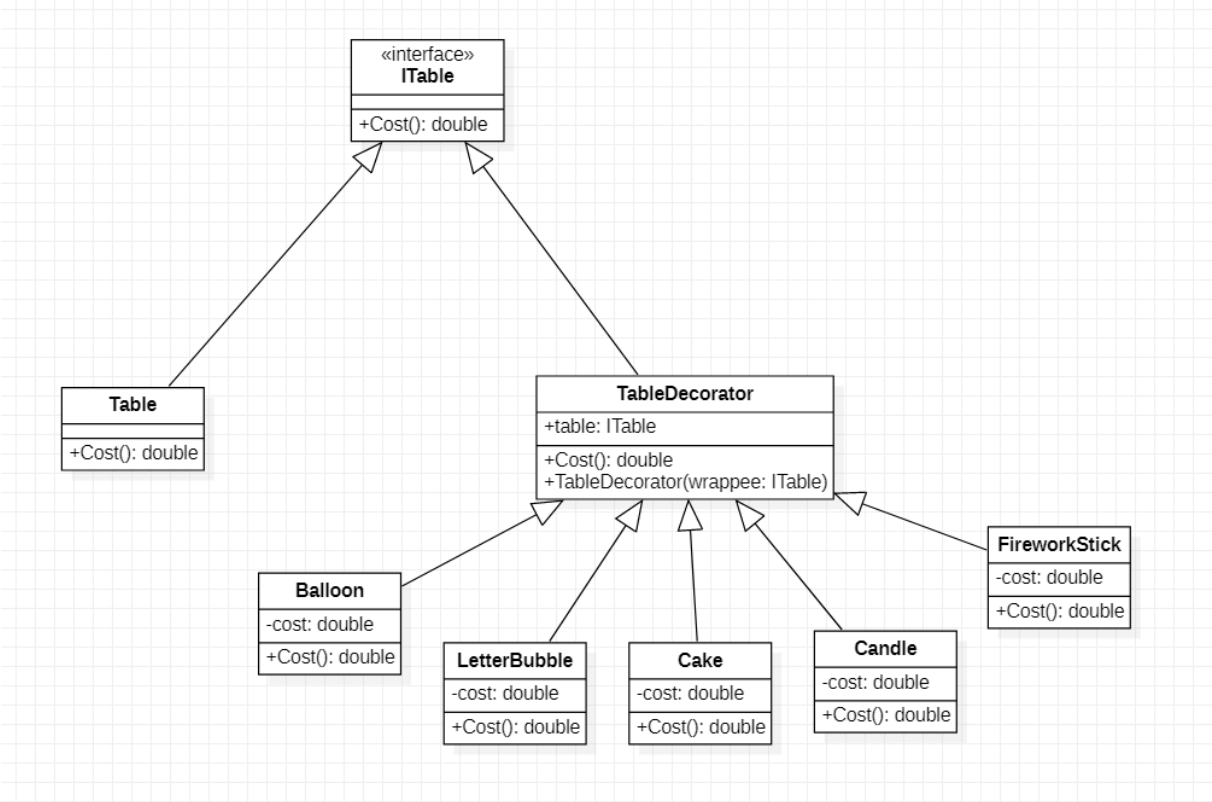


- **Code:**

  https://github.com/DevLeeHuy/FinalProjectOOSD/blob/main/Design_Pattern/Observer/Program.cs

❖ Decorator

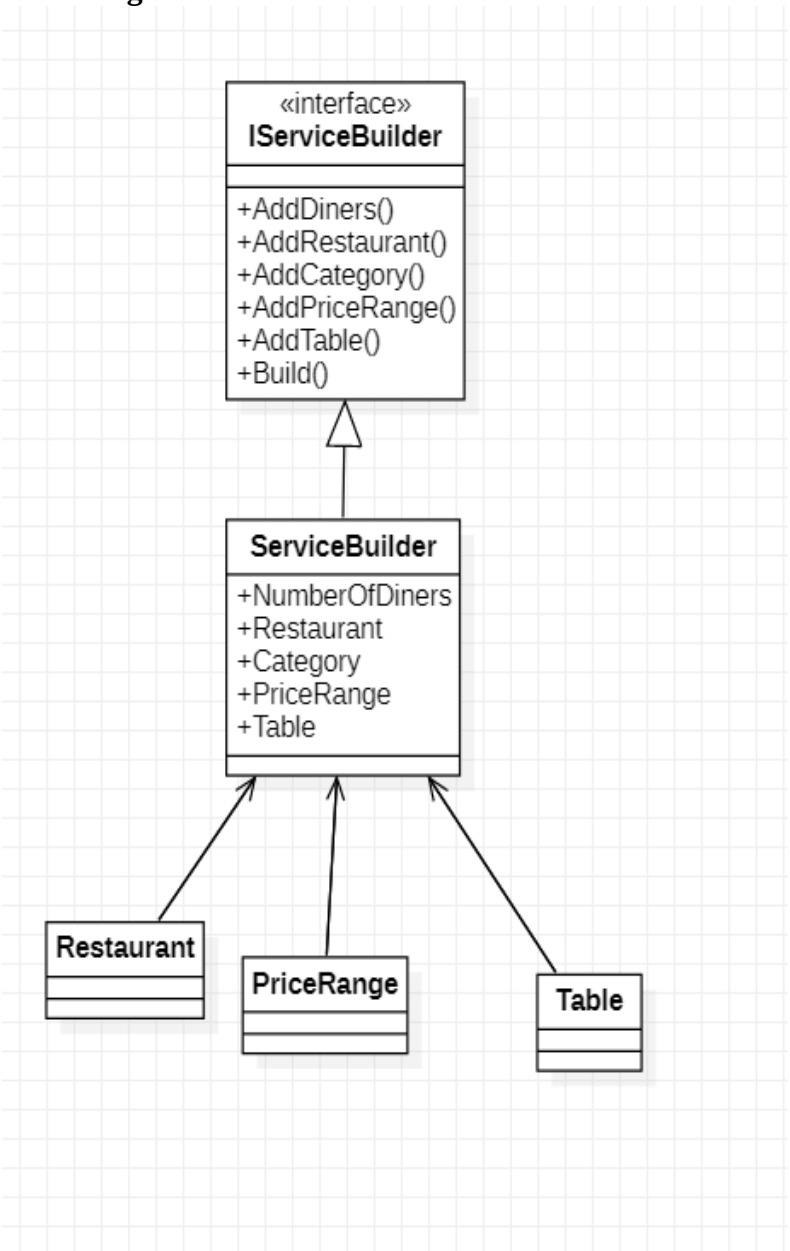
- **For diners can decorate table they booking for some kinds of party like Birthday.**
- **Why do you select this design pattern?**
  By use decorator when user decorate their table the cost automatically calculated by wrapping all decorators. It makes decorating more flexible without being bound by certain patterns.
- **The characteristic of this design pattern:**
  The decorator pattern is a design pattern that allows behavior to be added to an individual object, dynamically, without affecting the behavior of other objects from the same class.
- **The characteristic of this design pattern:**



- **Code:**
  https://github.com/DevLeeHuy/FinalProjectOOSD/blob/main/Design_Pattern/Decorator/Program.cs

❖ Builder

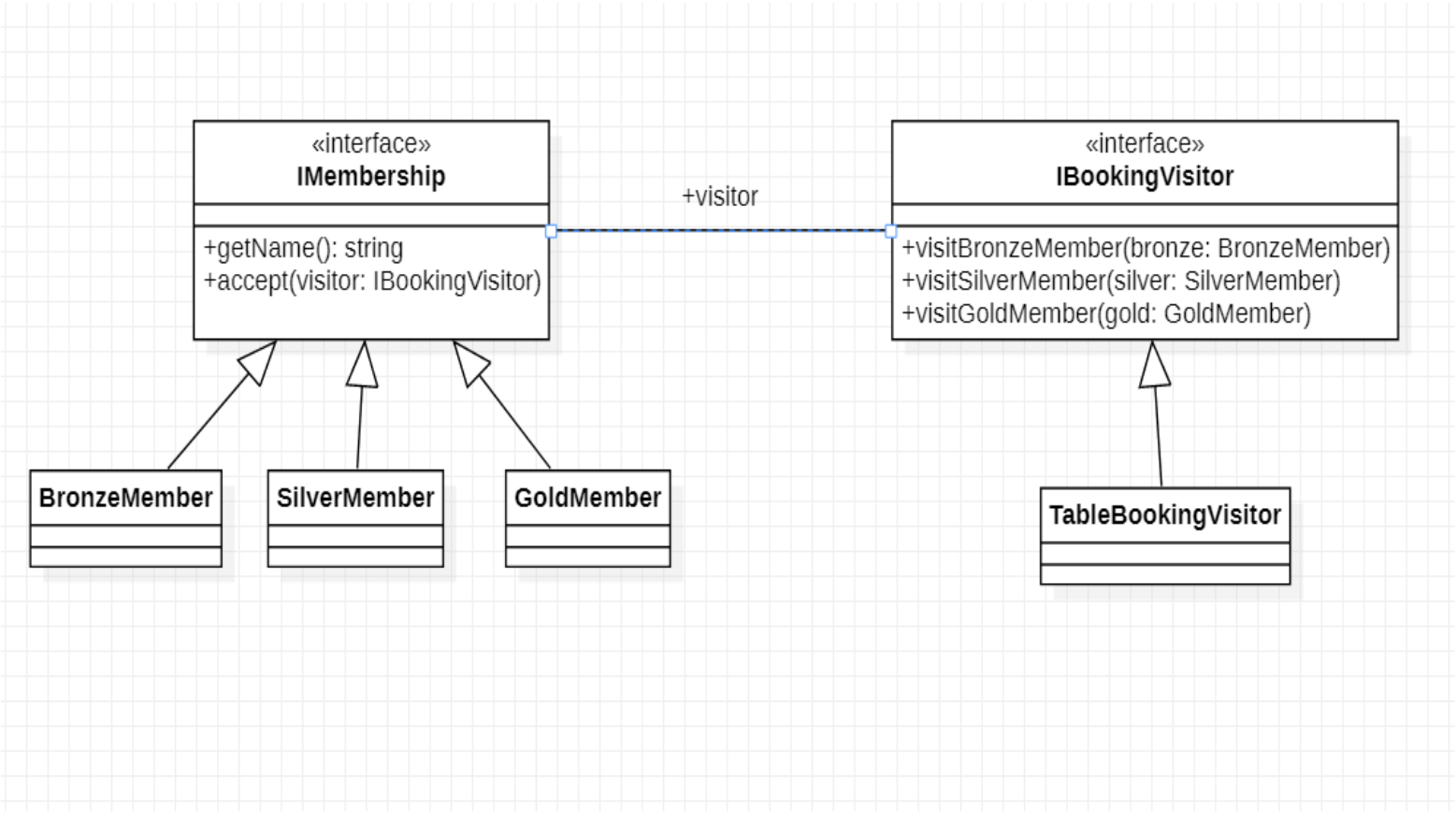
- **Create a service with many properties.**
- **Why do you select this design pattern?**
  Creating objects with too many properties (includes separate object) using conventional constructors easily leads to confusion and lack of transparency. Therefore, using builder complex objects step by step with concrete methods.
- **The characteristic of this design pattern:**
  The Builder pattern suggests that you extract the object construction code out of its own class and move it to separate objects called builders. The Builder pattern lets you construct complex objects step by step. The Builder doesn't allow other objects to access the product while it's being built.
- **Class diagram:**



- **Code:**
  https://github.com/DevLeeHuy/FinalProjectOOSD/blob/main/Design_Pattern/Builder/Program.cs

❖ Visitor:

- **Booking table by different memberships.**
- **Why do you select this design pattern?**
- When we have to perform an operation on a group of similar object types like memberships(bronze, silver, gold...). We can move behavioral logic from objects to another class.
- **The characteristic of this design pattern:**
  The visitor pattern or visitor design pattern is a pattern that will separate an algorithm from the object structure on which it operates. It describes a way to add new operations to existing object structures without modifying the structures themselves.
- **Class diagram:**

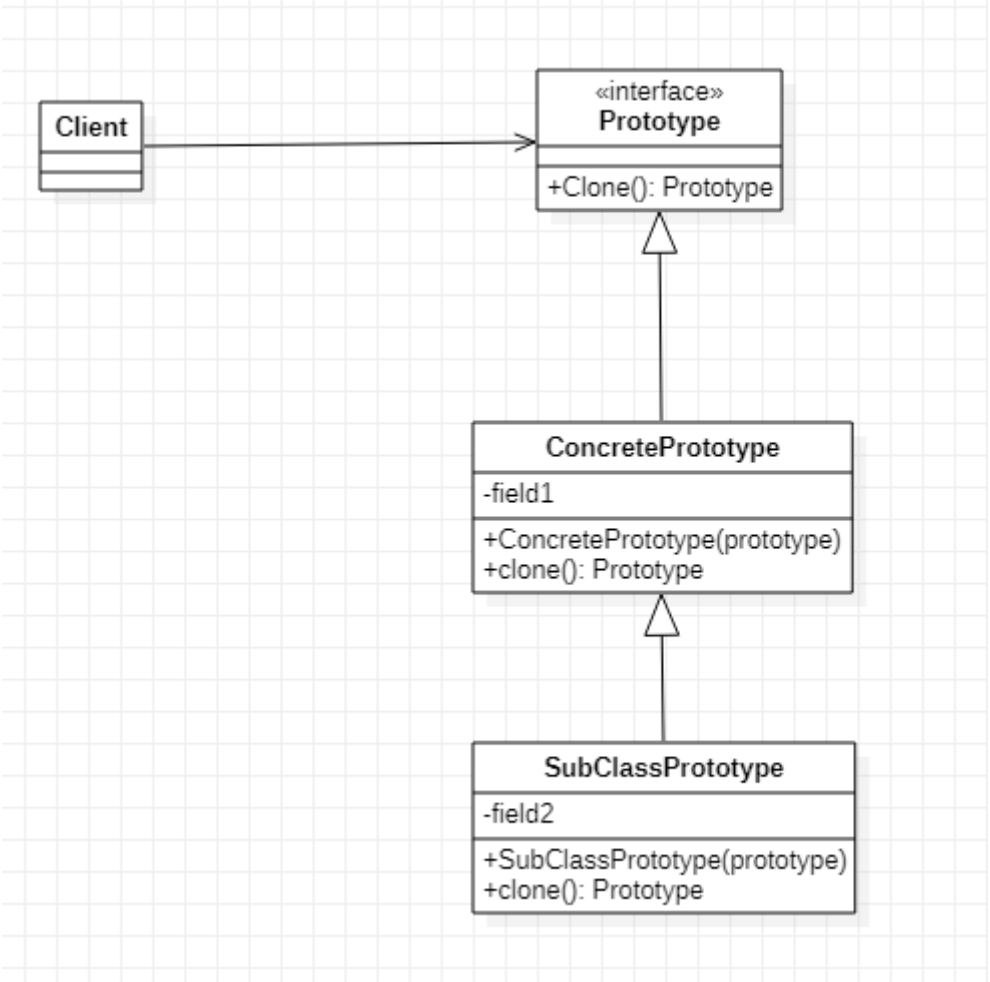


- **Code:**

  https://github.com/DevLeeHuy/FinalProjectOOSD/blob/main/Design_Pattern/VisitorPattern/Program.cs

❖ Prototype

- **Create & edit a user.**
- **Why do you select this design pattern?**
  When edit a user, create a clone user will help the main object isn't affected by changing some fields. Therefore, if you cancel changes, main object would not be changed just delete the clone object.
  And when create a new user, system can clone from exist user (like default user) and do not declare new object.
- **The characteristic of this design pattern:**
  The Prototype pattern delegates the cloning process to the actual objects that are being cloned. The pattern declares a common interface for all objects that support cloning. This interface lets you clone an object without coupling your code to the class of that object. Usually, such an interface contains just a single clone method.
- **Class diagram:**



- **Code:**

  https://github.com/DevLeeHuy/FinalProjectOOSD/blob/main/Design_Pattern/PrototypePattern/Program.cs