Software Design Patterns

讀書會#2

2013.10.02

qcl

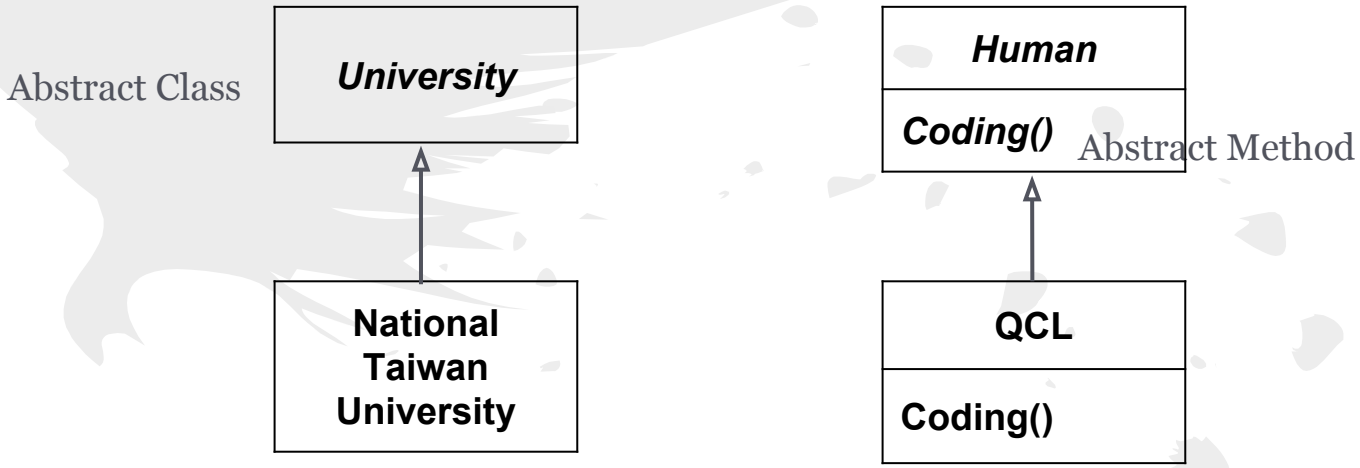
- 國立臺灣大學資訊工程學學士(2012)
- 土木所「設計模式與軟體開發」批改郎(2012)
- qcl wants a girlfriend (2013)

讀書會#2

- Adapter
- Bridge
- Façade

Class Diagram

- *Abstract / Virtual*



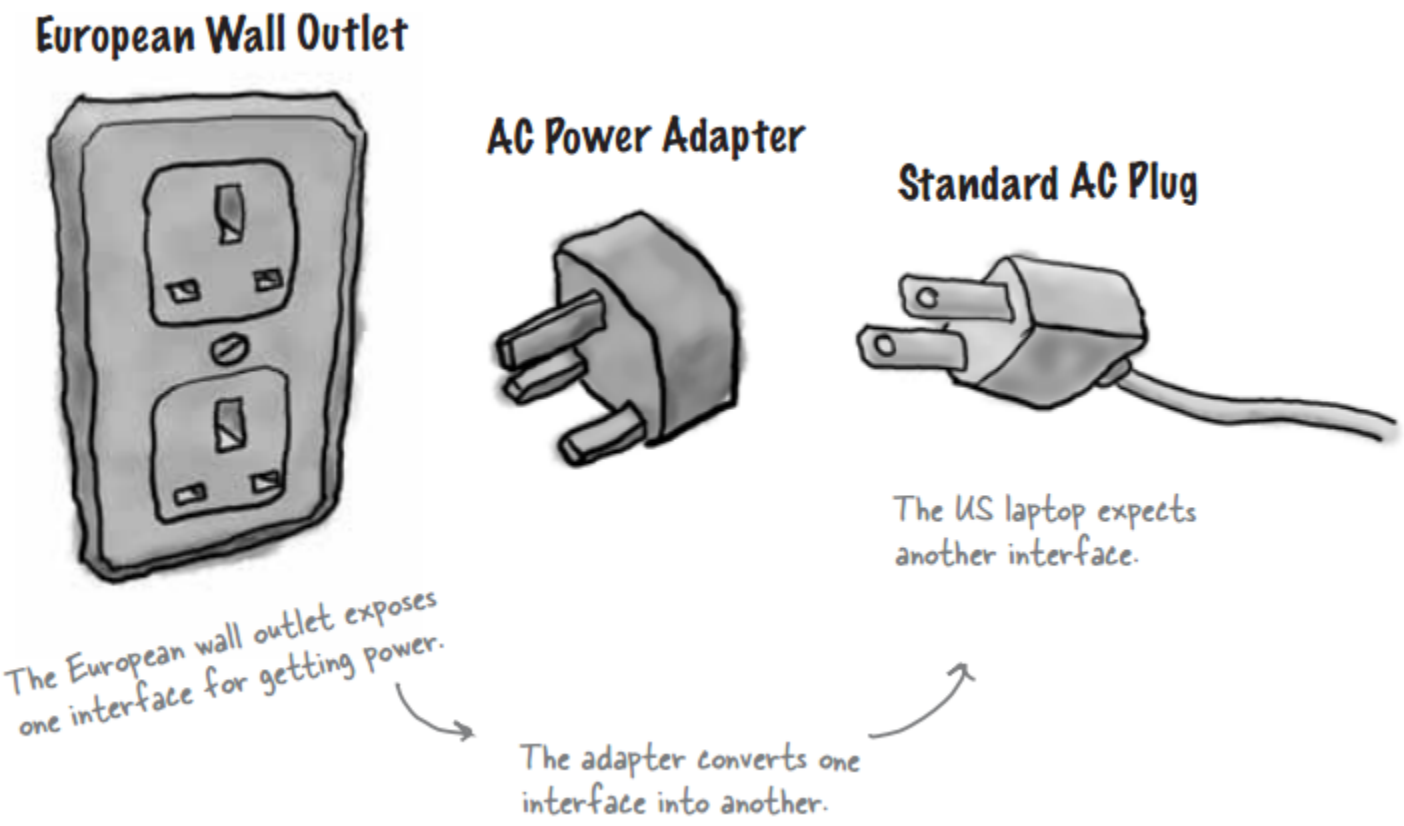
Adapter

Class/Object Structural Pattern

- GoF Structural Pattern
- Head First Chapter 7

Adapter pattern

Adapter


European Wall Outlet
AC Power Adapter
Standard AC Plug
The US laptop expects another interface.
The European wall outlet exposes one interface for getting power.
The adapter converts one interface into another.

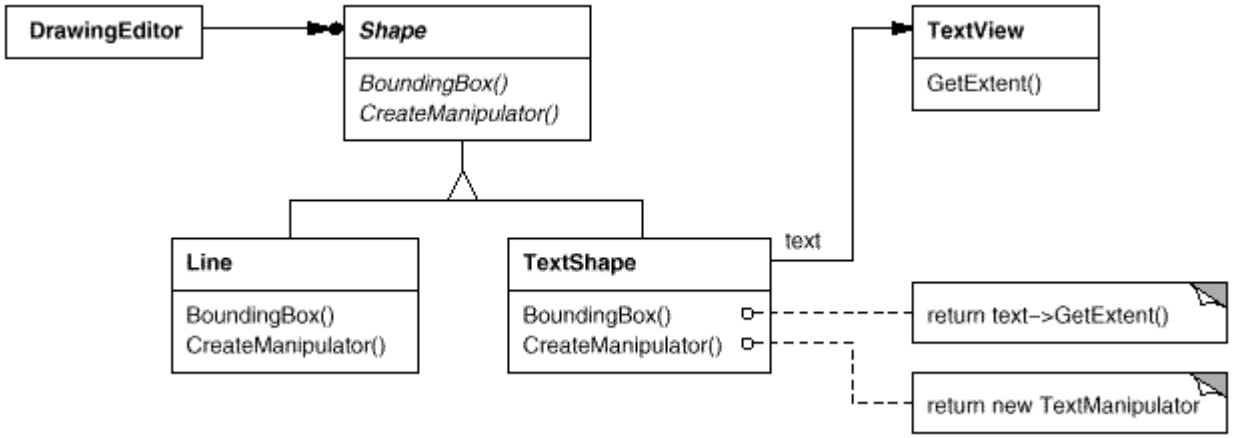
Another example (?)

Intent

- Convert the interface of a class into another clients except.
- Adapter lets classes work together that couldn’t otherwise because of incompatiable interfaces.

Motivation

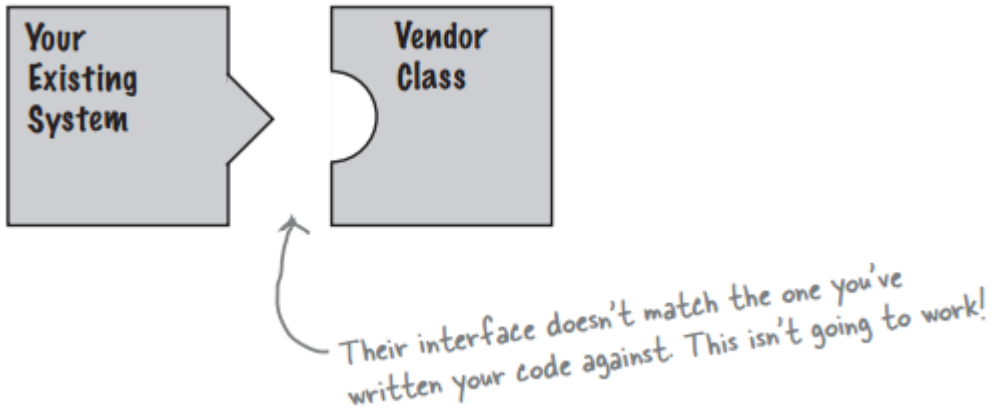
- Sometimes a toolkit class that's designed for reuse isn't reusable only because its interface don't match the domain-specific interface an application requires.



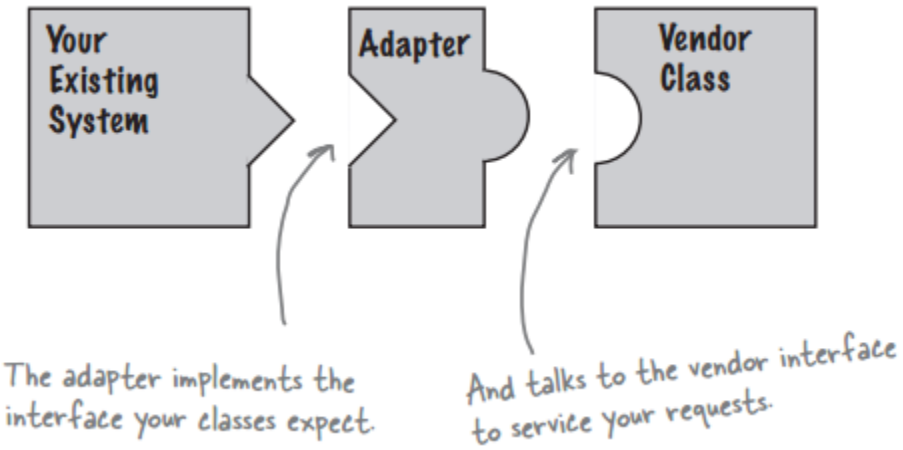
Applicability

- Use Adapter when:
  - you want to use an existing class, and its interface does not match the one you need.
  - you want ot create a reusable class that cooperates with unrelated or unforessen classes, that is, classes don’t necessarily have compatible interface.
  - (object only) you need to use serveral existing subclasses, but it’s impractical to adapt their interface by subclassing every one. An object adapter can adapt the interface of its parent class.

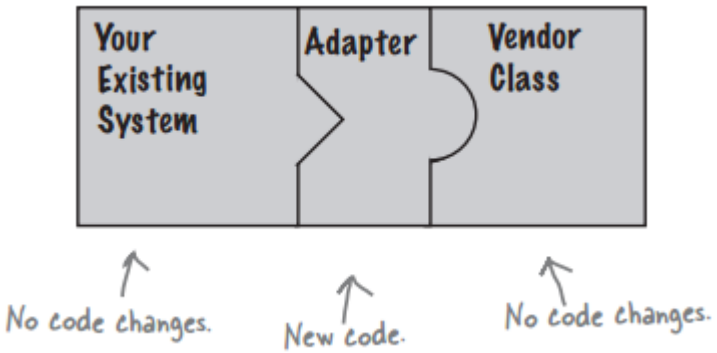
Object oriented adapters



Object oriented adapters

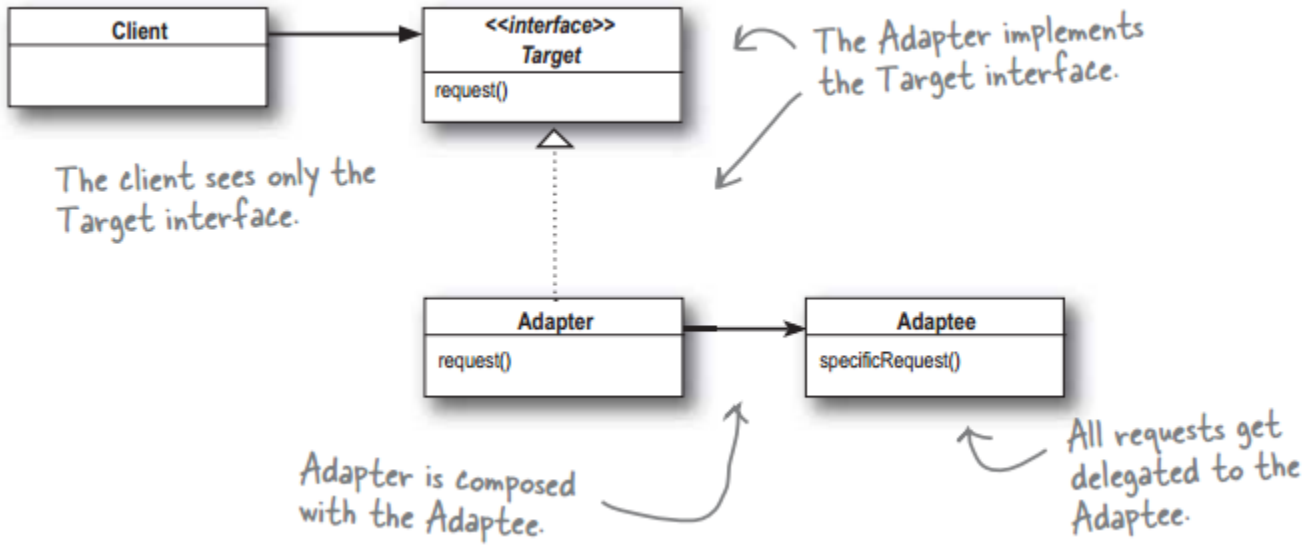


Object oriented adapters



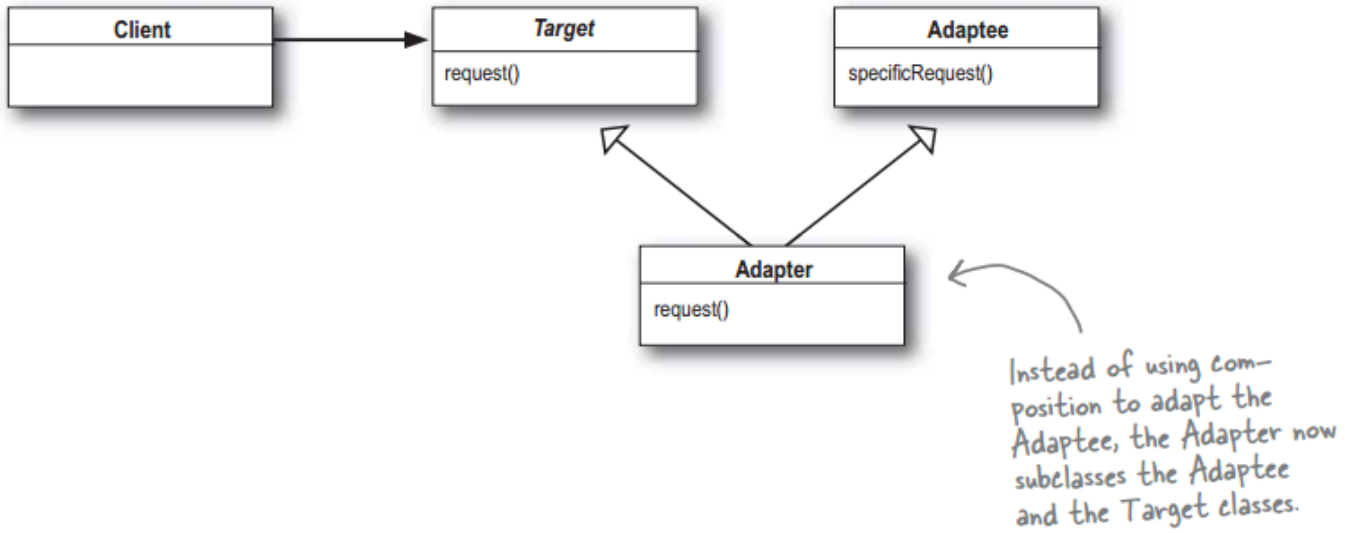
Structure

- Object Adapter
  - relies on object composition
- Adapter **has a** adaptee



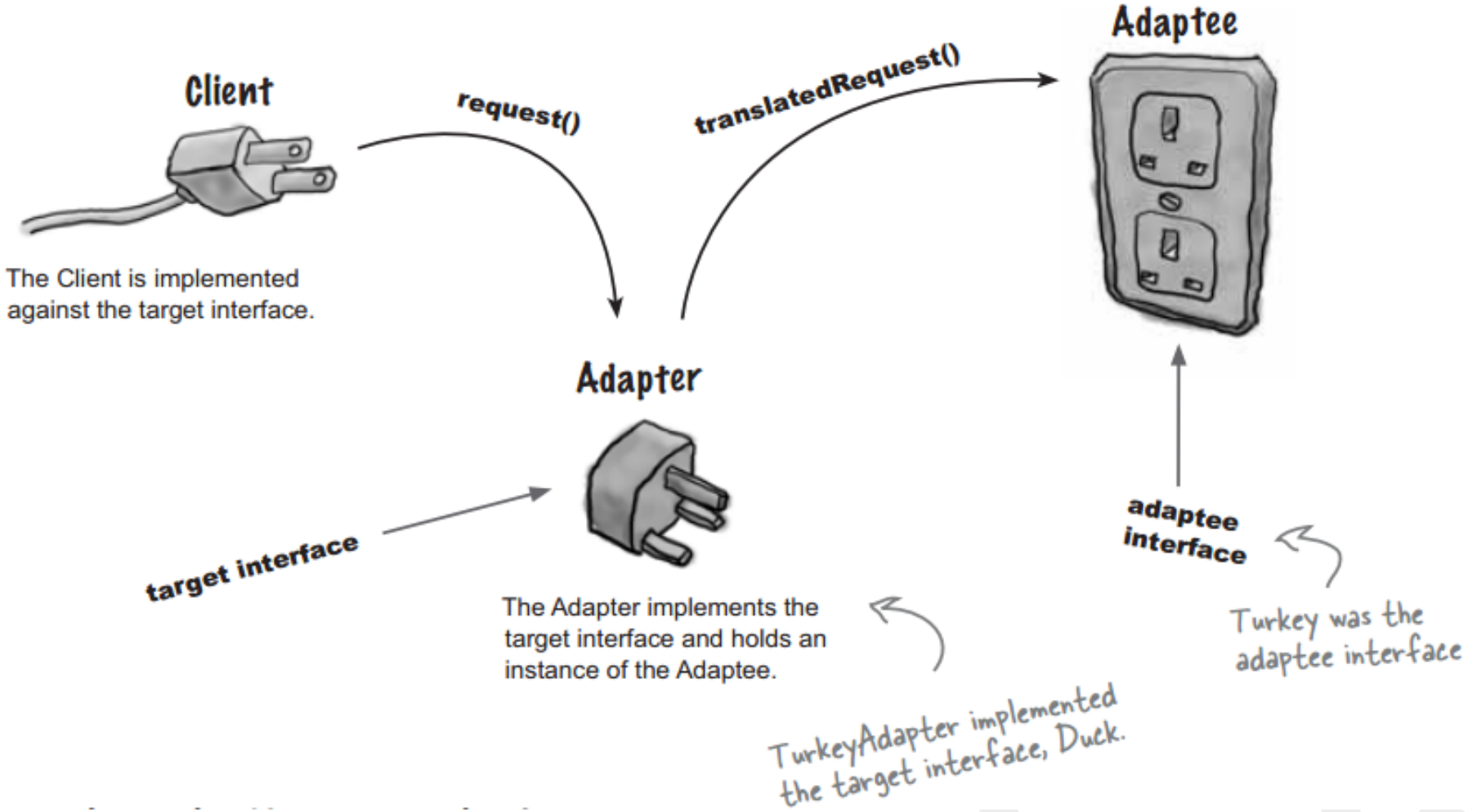
Structure

- Class Adapter
  - uses multiple inheritance to adapt one inferface to another.
- Adapter **is a** adaptee
  - private 繼承 adaptee



Participants

- Target
  - define the domain-specific interface that Client uses.
- Client
  - collaborates with objects conforming to the Target inferface.
- Adaptee
  - define an existing interface that needs adapting.
- Adapter
  - adapts the interface of Adaptee to Target interface.



participants of Adapter pattern

後果

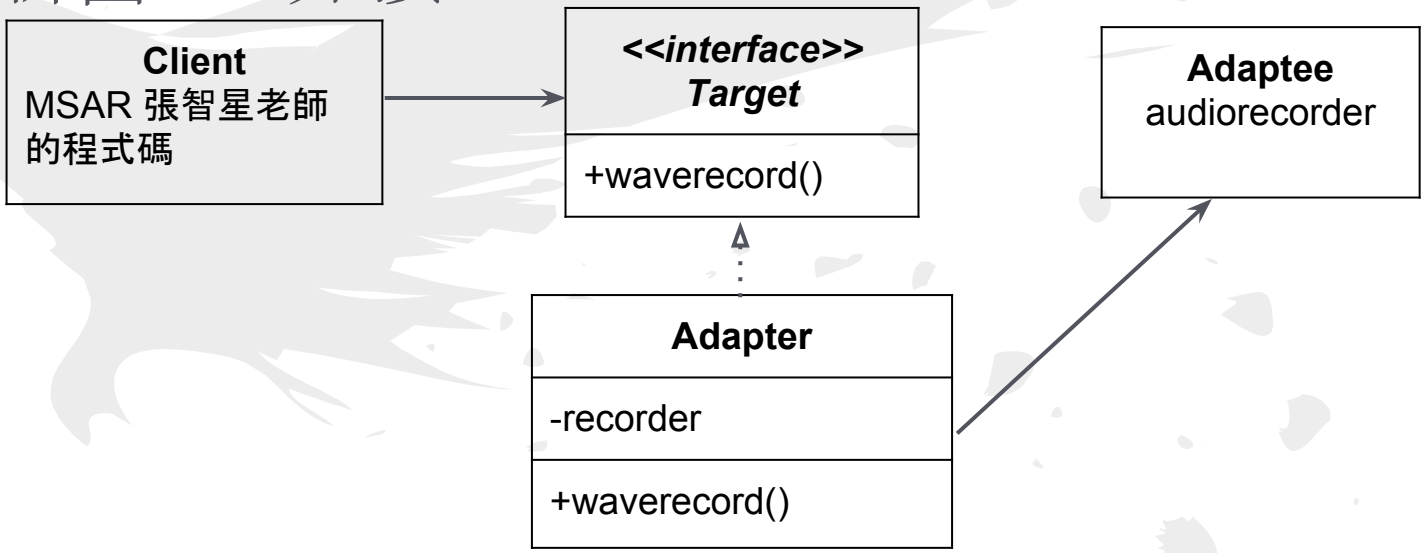
- Object Adapter
  - 一個Adapter可以和多個Adaptee一起玩(以及他們的孩子們)
  - 沒法override Adaptee的行為
- Class Adapter
  - 可以override Adaptee的行為
  - 沒辦法跟Adaptee的孩子們玩耍
  - 就是個物件, 不需要而外的指標去指Adaptee

一些考慮

- Adapter到底轉換了多少東西？
- Pluggable adapter
- Two-way adapter
  - 多重繼承Target

例子

- 新舊API介接




```
function y = wavrecord(n,Fs,ch,dtype)
%略
recorder = audiorecorder(Fs,nbits,ch);
recordblocking(recorder,n/Fs);
y = getaudiodata(recorder,dtype);
%略
```

https://github.com/qcl/qcl-matlab-adapter

- Bridge
  - 結構很像Adapter, 目的不同。
  - Adapter想轉換現有物件的界面, Bridge是要將實作體系與抽象體系分離。
- Decorator
  - 可在不改變物件介面的狀態下增靜功能, 比Adapter更句通透性。
- Proxy
  - 替其他物件預留空位, 但不會改變介面。

Questions?

Bridge

Object Structural Pattern

- GoF Sturctural Patterns

Bridge pattern

目的

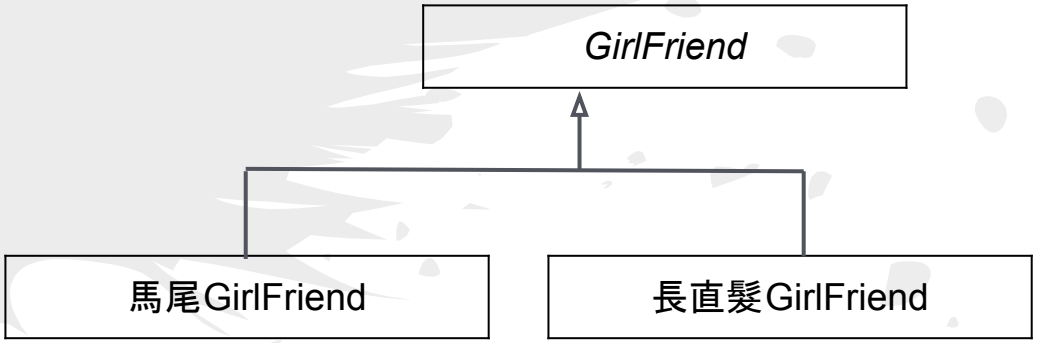
- **Decouple** an abstraction from its implementation so that the two can vary independently.

動機

- 如果某個抽象體有好幾種可能的實作方式, 我們常使用繼承的方式來處理。
- 定義抽象體的介面, 然後用各種具象的子類別來實作。
- 這種做法彈性未必足夠。

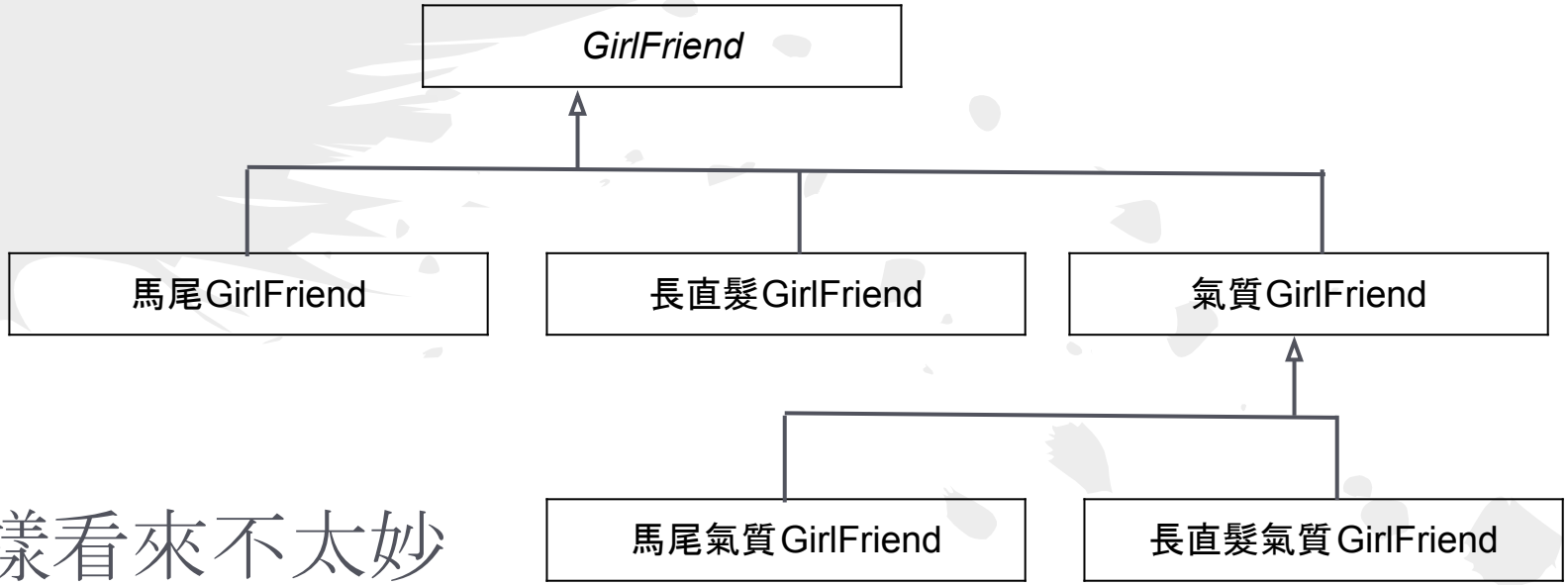
舉例來說

- 我希望我的女朋友是綁馬尾或是長直髮

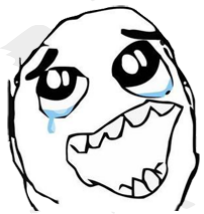


舉例來說，雖然不是個好例子

- 我希望我的女朋友同時也是是氣質妹

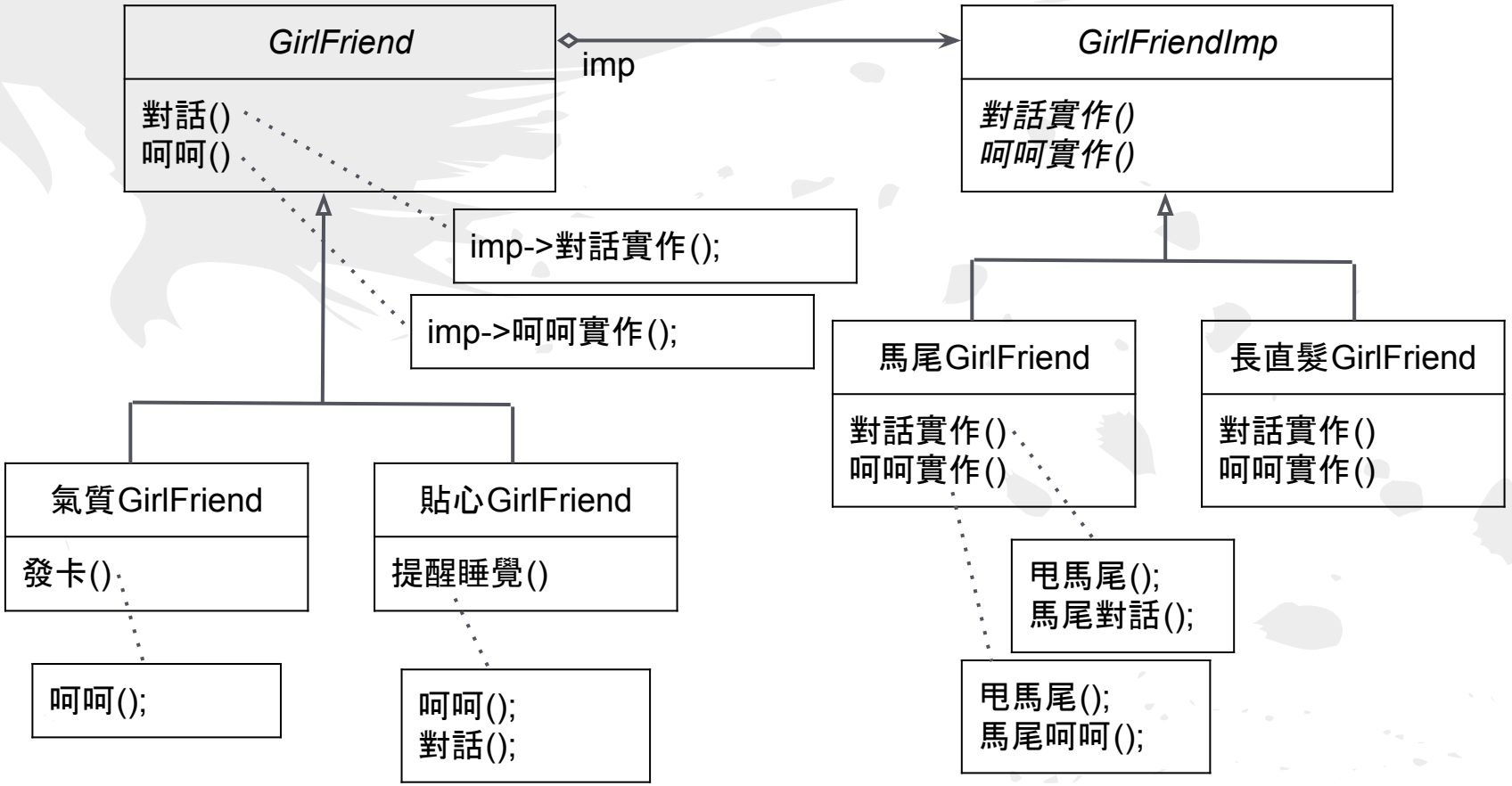


- 這樣看來不太妙
  - 特質與髮型不再保持獨立 ~~好多女朋友(yay)~~
  - 每多一種女友就會多很多subGFclasses
    - ~~長/短裙女友、黑褲襪女友...etc~~



Decoupling!

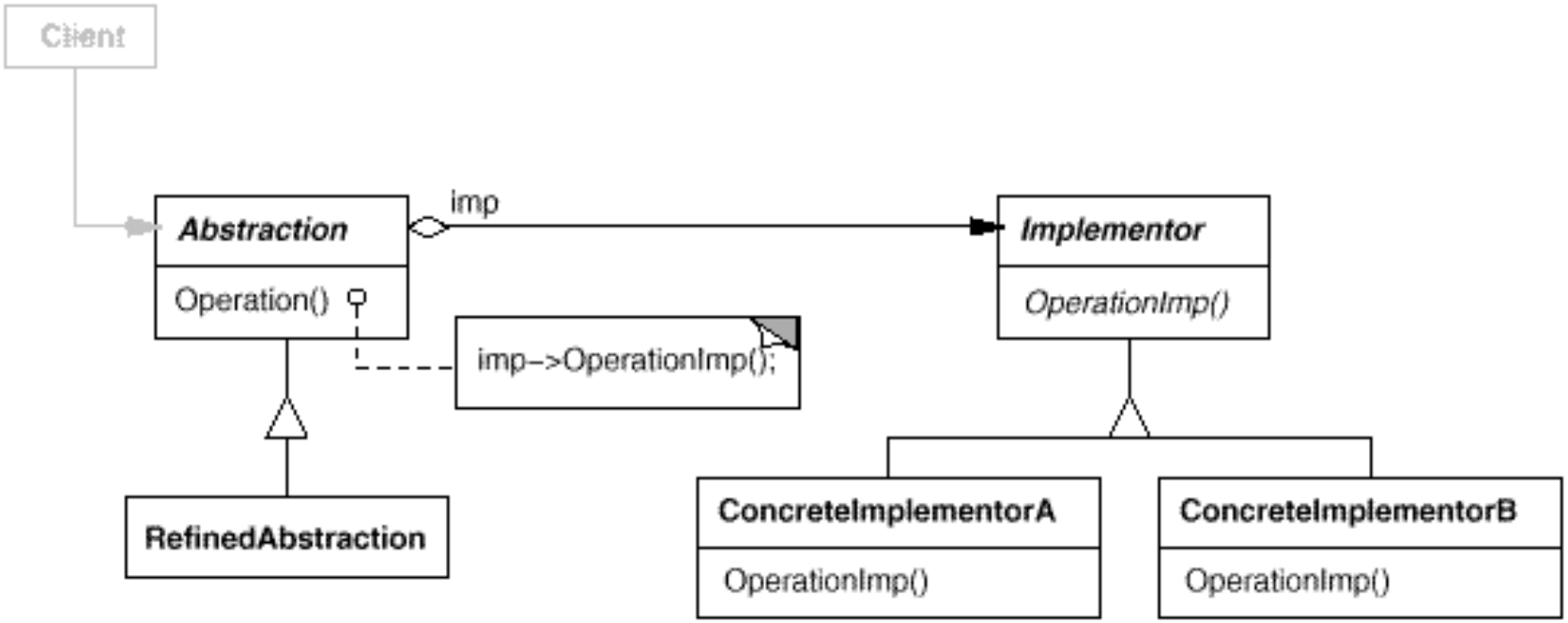
- 將抽象與實作分離



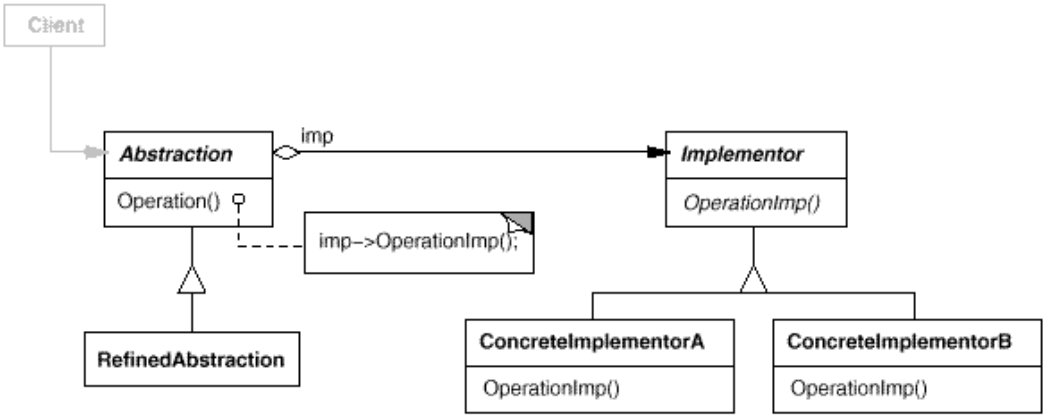
應用時機

- 避免將抽象體與實作體綁在一起
  - 執行期仍能選擇或切換
  - ~~執行期馬尾女友也可以換成長直髮女友~~
- 希望能用子類別個別擴充抽象體與具象體
- 如果只有實作體有變、但抽象體沒變時，就不應該波及客戶碼
  - 客戶碼不必重新編譯
  - ~~QCL無痛交女友(？)~~
- 將抽象體的實作方式藏起來
  - ~~不想讓你們知道女朋友是誰(？)~~

結構


Client
Abstraction
Operation()
imp
Implementor
OperationImp()
imp->OperationImp();
RefinedAbstraction
ConcreteImplementorA
OperationImp()
ConcreteImplementorB
OperationImp()

參與者



- Abstraction
  - 抽象體的介面
  - 維護指向Implementor的reference
- RefinedAbstraction
  - 擴充Abstraction所訂定的介面
- Implementor
  - 定義實作類別的共同介面(未必與Abstraction相同)
- ConcreteImplementor
  - 具體實作Implementor所訂定的介面

效果~~非常顯著~~!

- 將介面與實作隔離
- 易於擴充
- 隱藏內部實作細節

例子

- 據說C++的STL有 (GoF中譯本)
- Sequential containers (like implementor)
  - vector, list, deque
- Container adaptors (like abstraction)
  - stack, queue, priority_queue

- stack<int, vector<int> >
  - using vector<int> to imlement stack

- Adapter
  - Adapter pattern通常是等到系統已經設計好了裁施行。
  - Bridge則是在系統設計之初就已經決定採用了。
- Abstract Factory
  - 可以建立特定的Bridge並設定組態

Questions?

Façade

Object Structural Pattern

- GoF Sturctural Patterns
- Head First Chapter 7

Façade pattern

Façade

Intent

- Provide a unified interface to a set of interfaces in a subsystem. Façade defines a high-level interface that makes the subsystem easier to use.

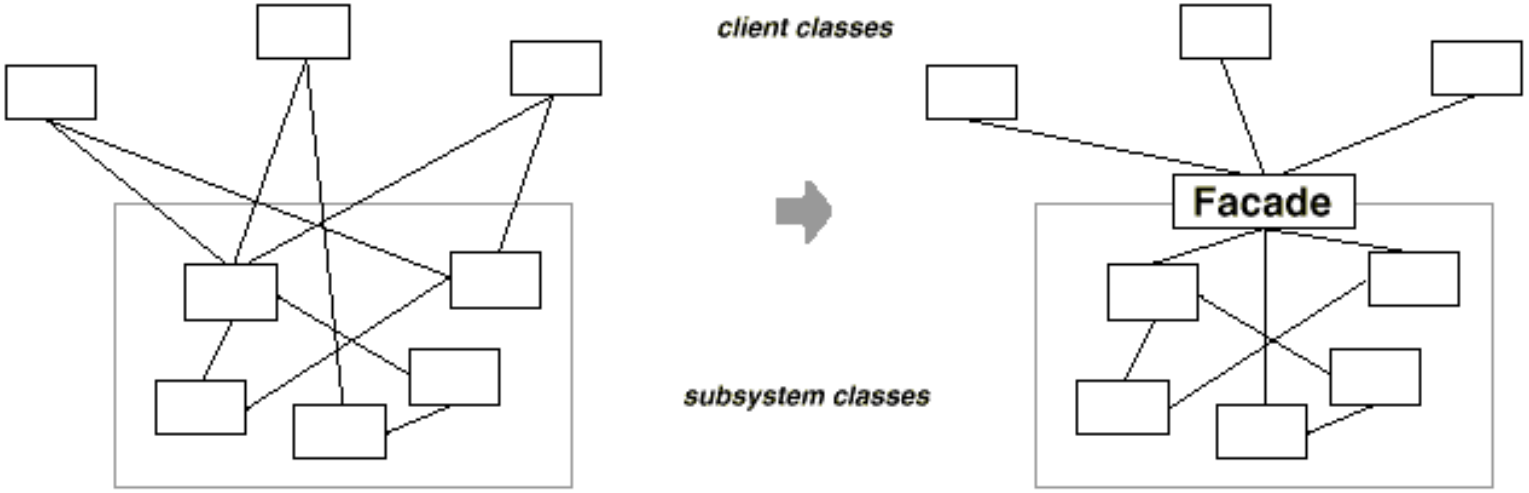
Motivation

- Structuring a system into subsystems helps reduce complexity. A common design goal is to minimize the communication and dependencies bewteen subsystems.
- One way to achieve this goal is to introduce a façade object that provides a single, simplified interface to the more general facilities of a subsystem.

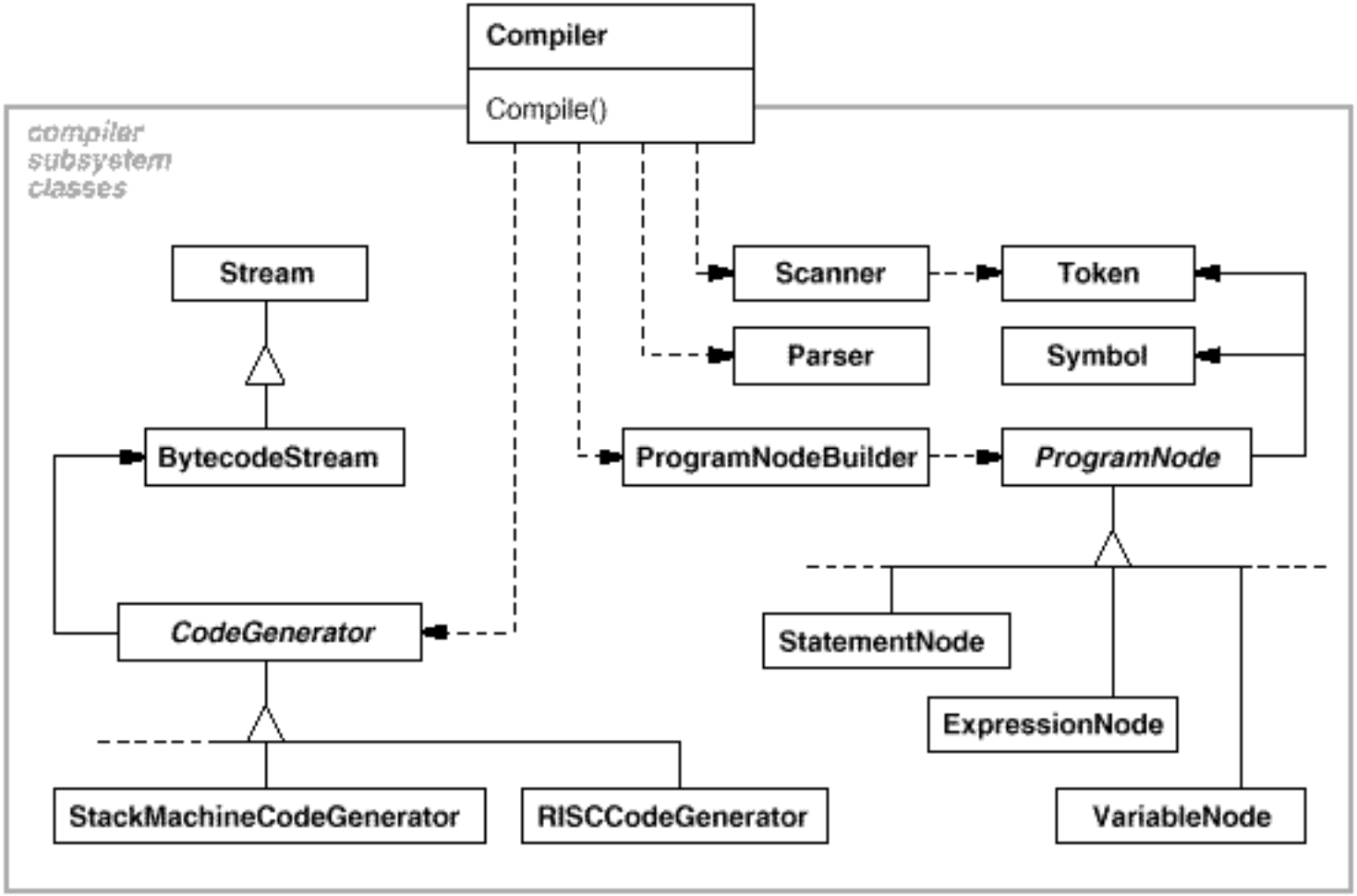
MyNTU?

#wtf

MyNTU



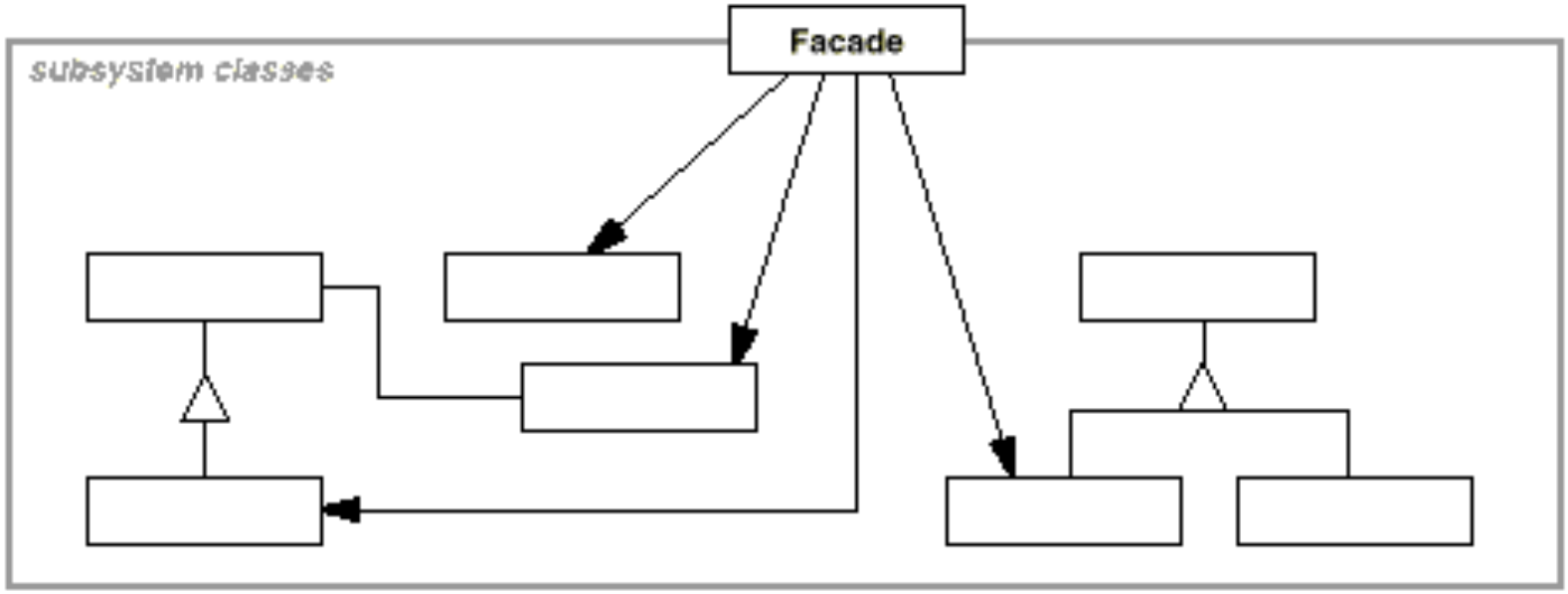
Compiler


Compiler
Compile()
compiler subsystem classes
Stream
Scanner
Token
Parser
Symbol
BytecodeStream
ProgramNodeBuilder
ProgramNode
CodeGenerator
StatementNode
ExpressionNode
StackMachineCodeGenerator
RISCCodeGenerator
VariableNode

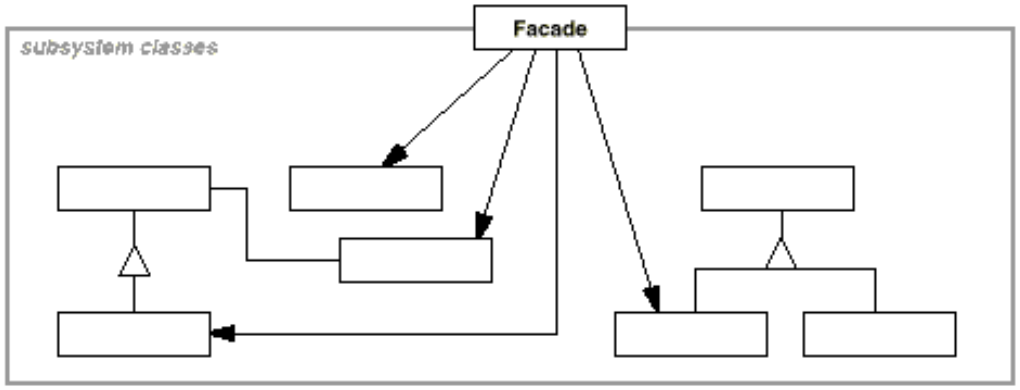
Applicability

- want to provide a simple interface to a complex subsystem.
- there are many dependencies between clients and the implementation classes of an abstraction.
- want to layer subsystems.

Structure



Participants



- Façade
  - knows which subsystem classes are responsible for request
  - delegates client requests to appropriate subsystem objects
- subsystem classes
  - implement subsystem functionality
  - handle work assigned by the Façade object.
  - have no knowledge of the façade
    - keep no references to façade

Consequences

- 避免外界觸及子系統的元件
  - 降低外界所需要面對的物件數量
- 外界與子系統低度耦合
- 如有必要, 外界仍可直接碰觸子系統

- Abstract Factory
- Mediator
- Singleton
  - 通常只要一個Façade，可以做成Singleton

Related patterns

Questions?

Thanks!

qcl
qingcheng.li @ qcl.tw
qc.linux @ gmail.com

@qcl (plurk/ptt/ptt2/github)
@qingchengli (twitter)