Software Design Pattern
Singleton & Flyweight

2014/2/8

YodaLee

讀書會#6

* Singleton
* Flyweight

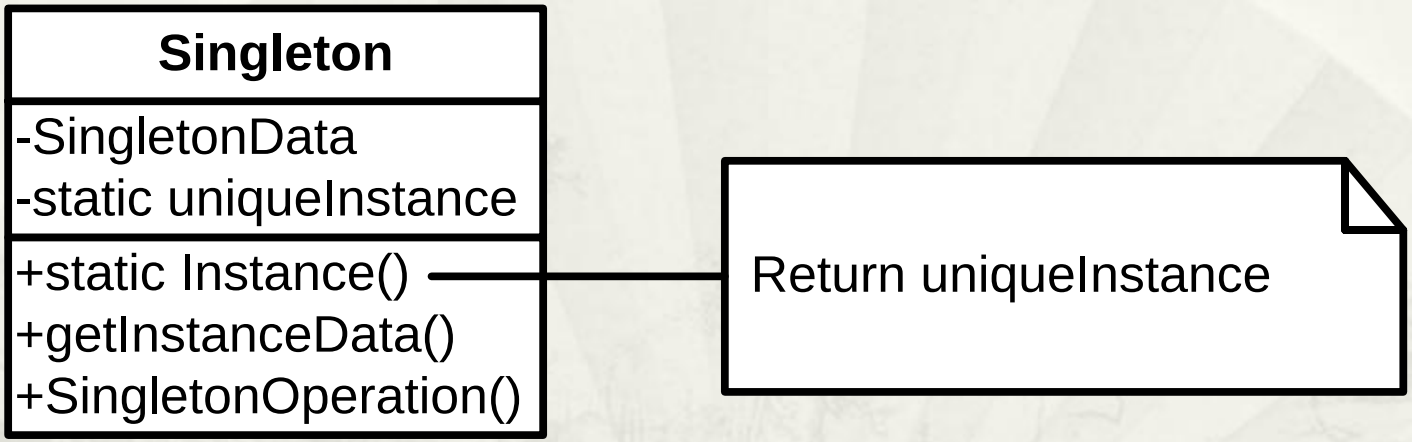
Singleton Pattern

Definition

* Intent in GoF
  * Ensure a class only has one instance, and provide a global point of access to it.

僅此一家，絕無分店

UML diagram


Singleton
-SingletonData
-static uniqueInstance
+static Instance()
+getInstanceData()
+SingletonOperation()
Return uniqueInstance

Benefit of Singleton 1

- Controlled access to sole instance
- Reduced name space
  - 相對於global 的好處

EX:

Struct QCL	writeHomework(QCL->homework)
Singleton QCL	writeHomework(QCL->homework()) //safety, prevent plagiarism

Benefit of Singleton (cont.)

- Permits refinement of operations and representation
- Permits a variable number of instances
- More flexible than class operations

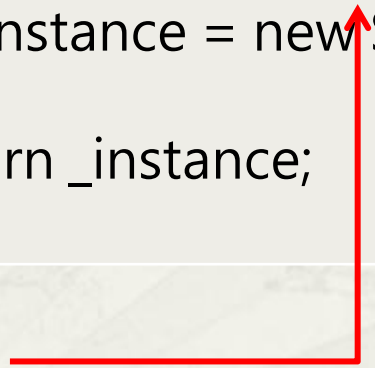
Implementation 1

```
class Singleton {
    public:
        static Singleton* Instance();
    protected:
        Singleton();
    private:
        static Singleton* _instance;
    };
```

```
 Singleton* Singleton::_instance =
0;

    Singleton* Singleton::Instance ()
{
        if (_instance == 0) {
            _instance = new Singleton;
        }
        return _instance;
    }
```

Here should apply mutex lock



Implementation 2 subclass

```
class Singleton {
  public:
    static void Register(const char* name, Singleton*);
    static Singleton* Instance();
  protected:
    static Singleton* Lookup(const char* name);
  private:
    static Singleton* _instance;
    static List<NameSingletonPair>* _registry;
  };
```

Implementation 2 subclass

```
Singleton* Singleton::Instance () {
        if (_instance == 0) {
            const char* singletonName = getenv("SINGLETON");
            // user or environment supplies this at startup

            _instance = Lookup(singletonName);
            // Lookup returns 0 if there's no such singleton
        }
        return _instance;
    }
```

```
MySingleton::MySingleton() {
        // ...
        Singleton::Register("MySingleton", this);
    }
```

Other issue

* Metaclass in many language.
* Relation to other pattern:
  * Factory, Builder, Prototype

Flyweight Pattern

Definition

- Intent in GoF
  - Use sharing to support large numbers of fine-grained objects efficiently.

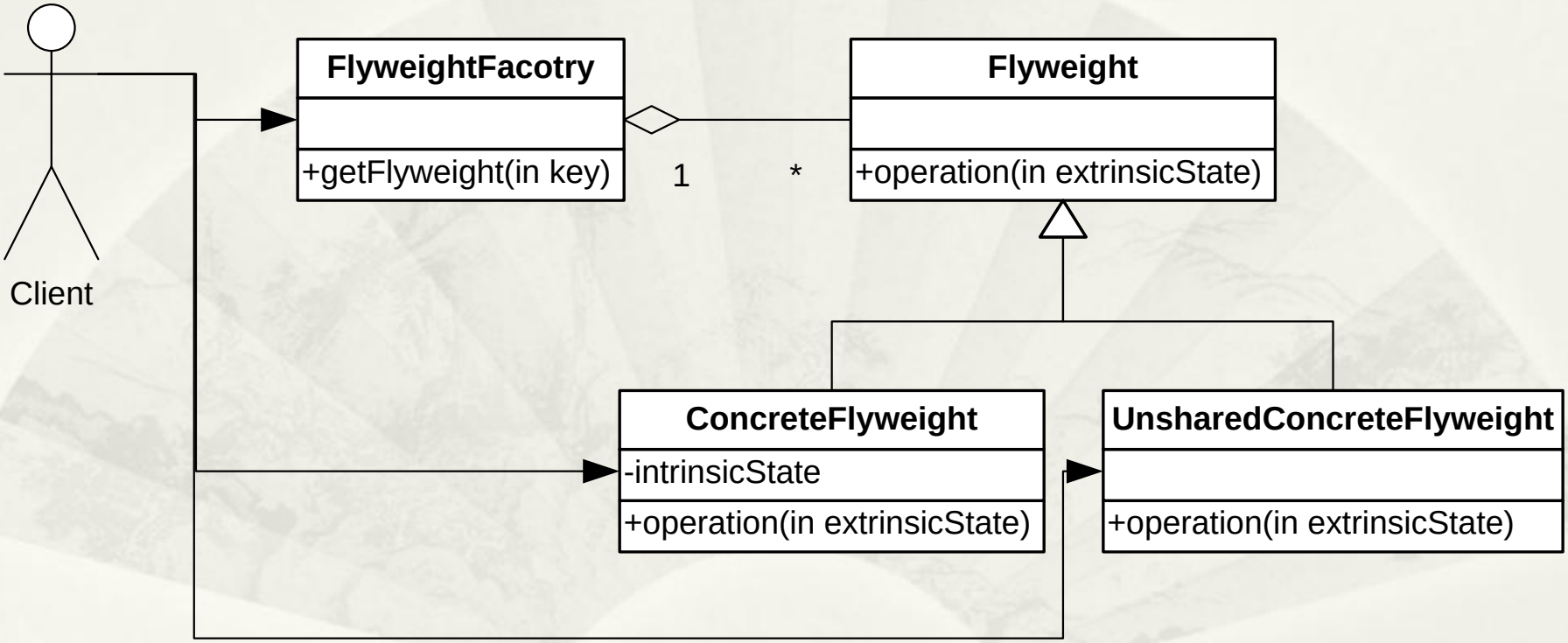
- 十萬青年十萬肝，GG**輪班**救台灣(誤)
  - 這是一個邪惡的pattern

何時用之？

Condition	Example
large number of objects.	員工很多
Storage costs are high	22K * 300000 = (OAO)
Most object state can be made extrinsic	生產線的狀態，不用員工來記
groups of objects may be replaced by relatively few shared objects	你是「勞工」
application doesn't depend on object identity	派遣工的概念.....

身寸☆惹ル--By obov

UML diagram


Client
FlyweightFacotry
+getFlyweight(in key)
1
*
Flyweight
+operation(in extrinsicState)
ConcreteFlyweight
-intrinsicState
+operation(in extrinsicState)
UnsharedConcreteFlyweight
+operation(in extrinsicState)

Implementation step

1. Removing extrinsic state
2. Managing shared objects

Related Pattern

- Composite Pattern
  - Manage composite by flyweight
- State, Strategy Pattern
  - Recommend implement by Flyweght

Other issue

* .NET pattern: use frequency: low