CS 247 Final Review

CS 247

University of Waterloo

cs247@student.cs.uwaterloo.ca

August 7, 2014

Overview

1. Pre-Midterm Material
   - Pre-Midterm Design Patterns
2. Post-Midterm Design Patterns
3. Templates
4. C++ Standard Template Library

Pre-Midterm Material

Important topics covered before the midterm include:

- ADTs
- Polymorphism
- Exceptions
- Unified Modelling Language
- Object Oriented Design Principles
- Some Design Patterns

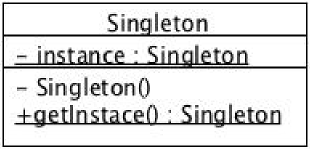
These are all covered in detail in the Midterm review session. Feel free to ask questions about them.

Design Patterns

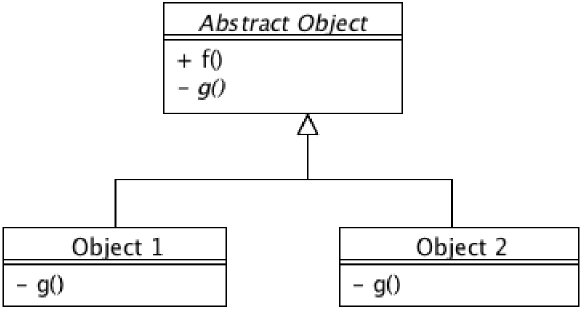
The design patterns covered before the midterm were :

- Singleton
- Template Method
- Facade
- Adapter
- Strategy
- Observer
- Model-View-Controller

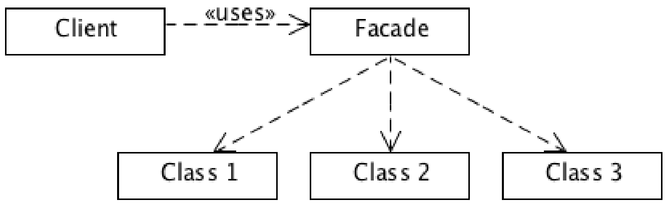
Singleton


Singleton
- instance : Singleton
- Singleton()
+getInstace() : Singleton

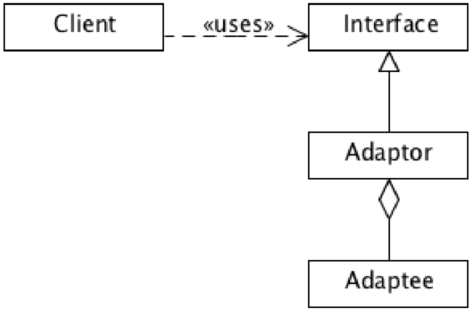
Template Method



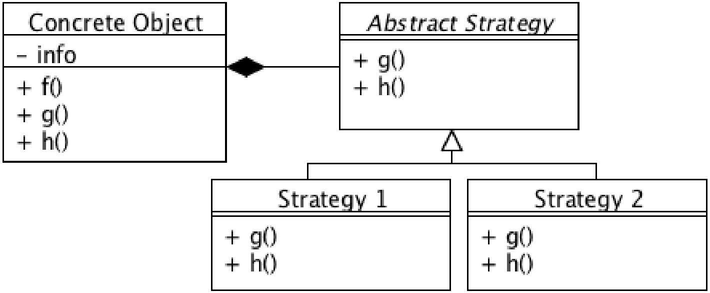
Facade



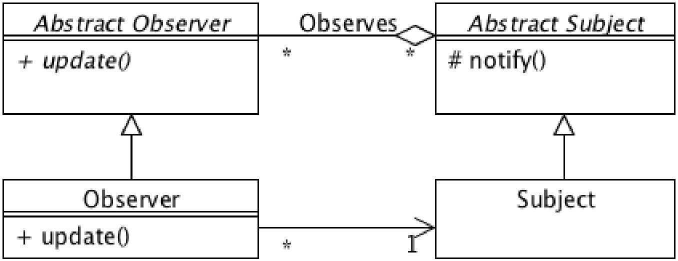
Adapter



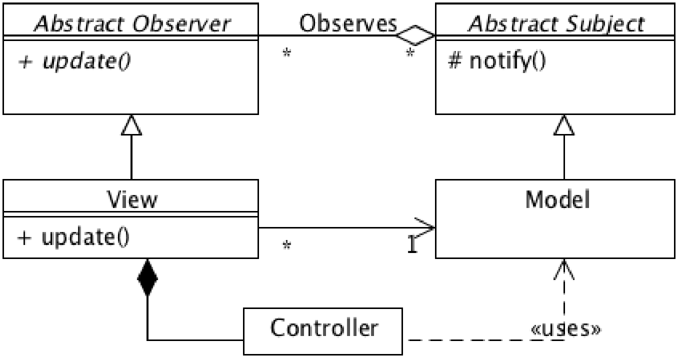
Strategy


Concrete Object
- info
+ f()
+ g()
+ h()
Abstract Strategy
+ g()
+ h()
Strategy 1
+ g()
+ h()
Strategy 2
+ g()
+ h()

Observer


Abstract Observer
+ update()
Observes
*
*
Abstract Subject
# notify()
Observer
+ update()
*
1
Subject

Model-View-Controller

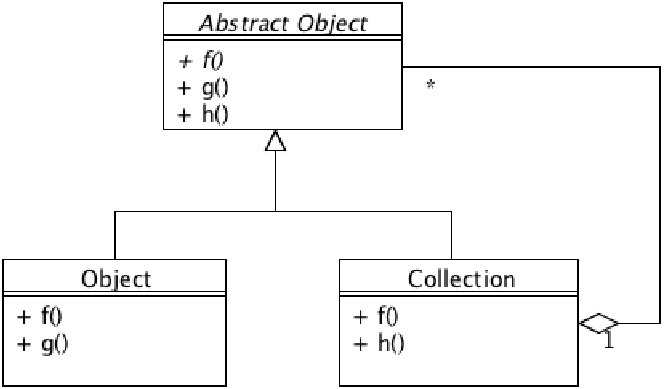


Design Patterns

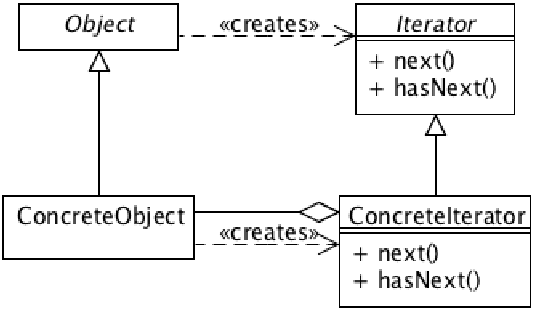
The design patterns covered after the midterm are:

- Composite
- Iterator
- Decorator
- Factory Method

Composite



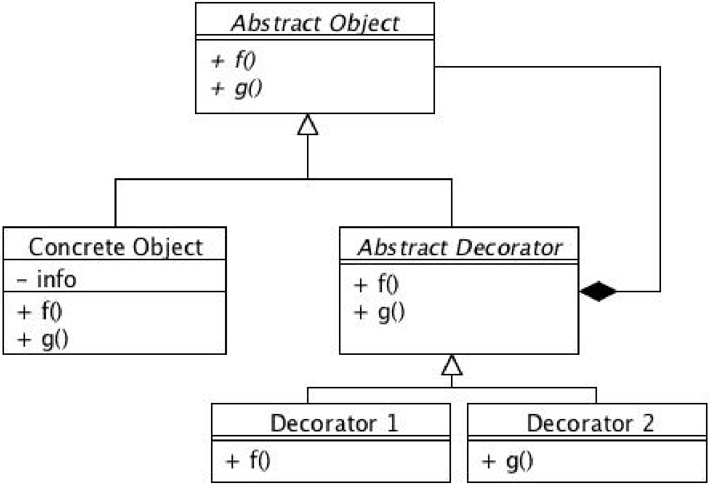
Iterator


Object
«creates»
Iterator
+ next()
+ hasNext()
ConcreteObject
«creates»
ConcreteIterator
+ next()
+ hasNext()

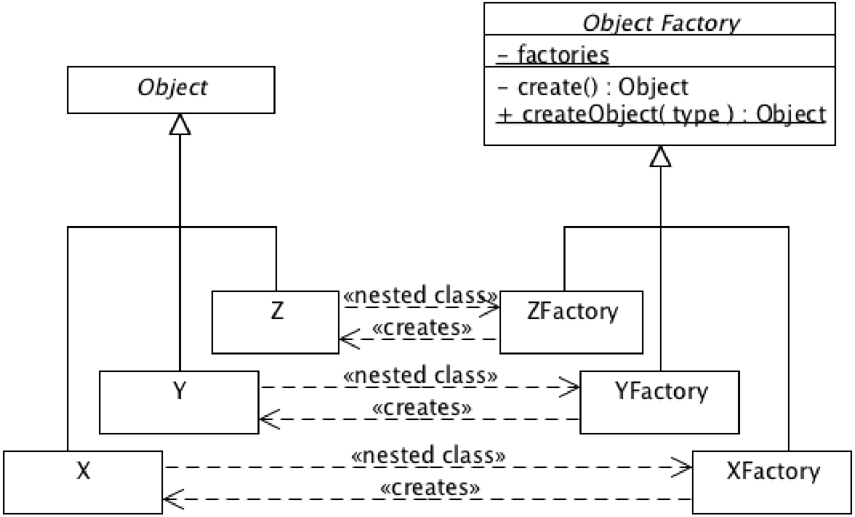
Composite Iterator

How does should the UML be extended to iterate over a composite?

Decorator


Abstract Object
+ f()
+ g()
Concrete Object
- info
+ f()
+ g()
Abstract Decorator
+ f()
+ g()
Decorator 1
+ f()
Decorator 2
+ g()

Factory Method



Templates

Templates are how we make generic functions and classes in C++.

Example Function Template

```
template <typename T>
bool operator>(T a, T b) {
    return b < a;
}
```

Example Class Template

```
template <typename T>
class Triple {
    T val[3];
  public:
    T first();
    T second();
    T third();
};
```

Template Exercise

Write the function foldl which takes in a binary functor f, an initial value i, and a list L. First, apply f to i and L_1 to produce L'_1, then apply f to L'_1 and L_2 to produce L'_2, and so on to eventually return L'_n. This function exists in the STL under the name accumulate. It is not in algorithm but in numeric.

foldl

```
template<typename it, typename T, typename proc>
T foldl(it first, it last, T init, proc f) {
    while (first!=last) {
        init = f(init,  *first);
        ++first;
    }

    return init;
}
```

Template Exercise

Write a function object which is constructed with a unary predicate that takes an integer and value and produces the integer plus one if the predicate is true for the value and the integer unchanged otherwise.

plusIf

```
template<typename pred>
class plusIf {
    pred f;
  public:
    plusIf(pred fun) : f(fun) {}

    template<typename T>
    int operator() (int n, T t) {
        return f(t) ? n + 1 : n;
    }
};
```

Templates

foldl (accumulate) can be used with this function object to implement the C++11 algorithm count_if.

```
template<typename it, typename pred>
int countIf(it first, it last, pred f) {
    plusIf<pred> fun(f);
    return foldl(first, last, 0, fun);
}
```

STL

The STL provides useful general purpose containers, iterators and algorithms.

STL Exercises

Write a program that reads in a sequence of numbers, sorts them, then writes them back to standard output

STL Exercises

```
istream_iterator<int> eos;
istream_iterator<int> iit(cin);
ostream_iterator<int> out(cout, "\n");
vector<int> num;

copy(iit, eos, back_inserter(num));
sort(num.begin(), num.end());
copy(num.begin(), num.end(), out);
```

STL Exercises

Write a program to iterate through integers in a container and count the number greater than 99.

STL Exercises

```
int n;
vector<int> num;
vector<int>::iterator nit;

nit = remove_if(num.begin(), num.end(),
                bind2nd(less<int>(), 99));
n = nit - num.begin();
```

The End