OO Design "Part II" unit:

- General introduction to DP; Strategy (**today**)
- State (Wednesday)
- Strategy revisited (lab Thursday)
- Adaptor and Decorator (Friday)
- (Begin pseudo-assembly unit next week Monday)
- State revisited (lab next week Thursday)

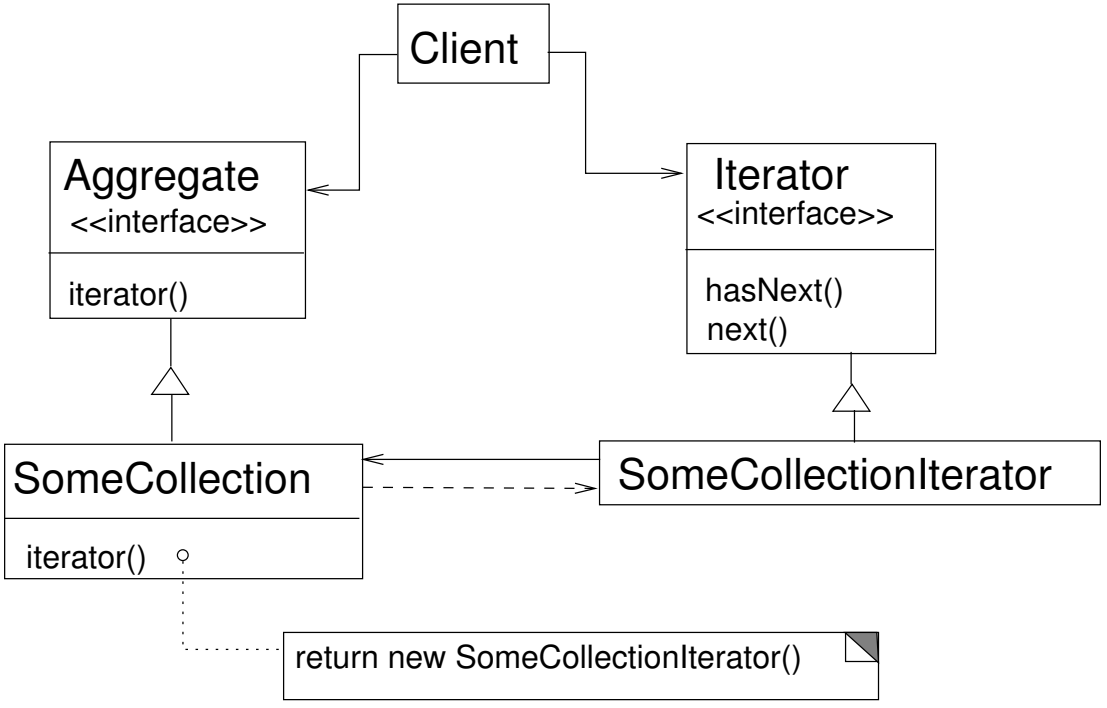
Today:

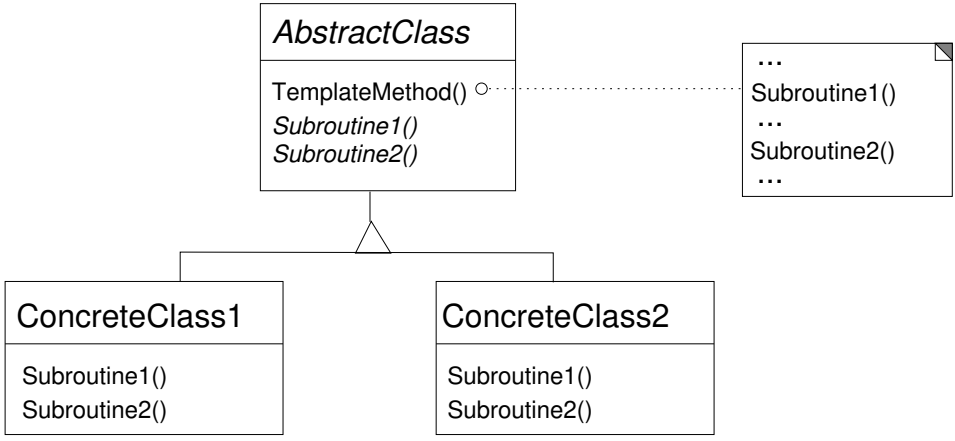
- Design patterns, in general
- Motivation for the Strategy pattern
- Strategy example
- Strategy summary

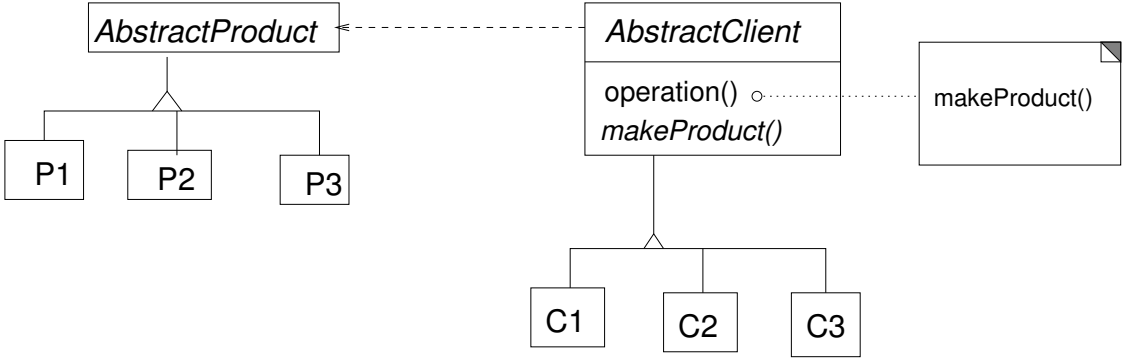
A *pattern* is a solution to a common (recurring) problem. A *design pattern* is a pattern for solving a design problem, where *design* refers to the structure and interaction of the participating classes and objects.

Design patterns we have already seen:

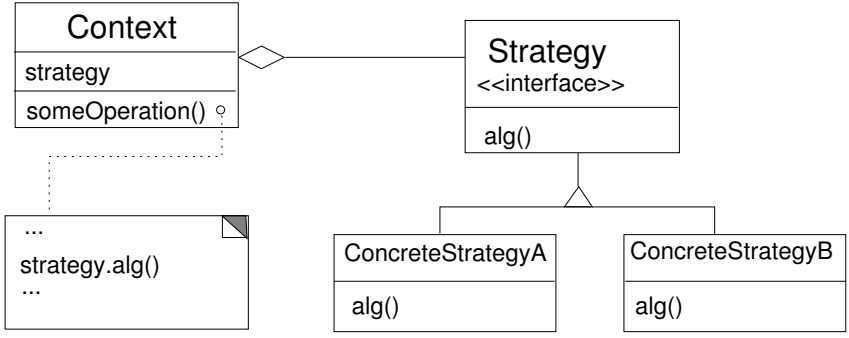
- Iterator
- Template Method
- Factory Method


Client
Aggregate
<<interface>>
iterator()
Iterator
<<interface>>
hasNext()
next()
SomeCollection
iterator()
SomeCollectionIterator
return new SomeCollectionIterator()


AbstractClass
TtemplateMethod()
Subroutine1()
Subroutine2()
...
Subroutine1()
...
Subroutine2()
...
ConcreteClass1
Subroutine1()
Subroutine2()
ConcreteClass2
Subroutine1()
Subroutine2()


AbstractProduct
AbstractClient
operation()
makeProduct()
makeProduct()
P1
P2
P3
C1
C2
C3

Strategy Pattern: Define a family of algorithms, encapsulate each one, and make them interchangeable. The Strategy Pattern lets the algorithm vary independently from the clients that use them.



Applicability:

- You have many related classes differing only in behavior
- You need variants of an algorithm (for space/time tradeoff, e.g.)
- An algorithm uses data that clients don't know about (this allows you to separate/better encapsulate some information from the client)
- A class may have many behaviors (facility for making pseudo-classes)

Consequences

+ Hierarchies exist of algorithms—opportunity for inheritance if they share some code
+ Polymorphic alternative to subclassing the context
+ Polymorphic alternative to conditional/switch
+ Choice of algorithms
- Clients must be aware of different strategies
- Communication overhead between client and strategy
- More objects (possibly)

Coming up:

- **Due Wed, Apr 22.** *Do Project 7, no-ifs calculator*
- **Due Thurs, Apr 16.** *Read prelab and take quiz (Coming soon. . .)*