Iterator

- ★ Structural Patterns
  - » strategy
  - » adapter
  - » façade
- ★ Behavioral Patterns
  - » observer
  - » decorator
  - » command
  - » template method
  - » iterator
- ★ Creational Patterns
  - » factory method
  - » abstract factory
  - » singleton

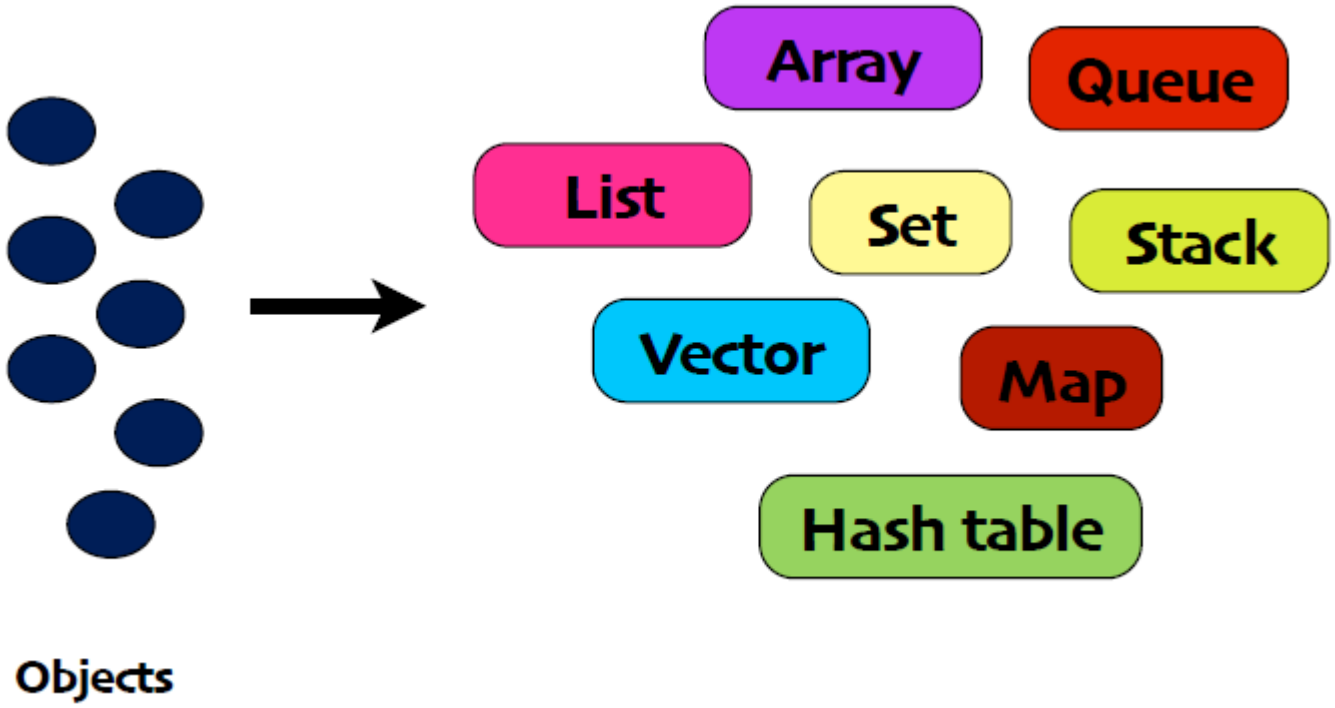
Iterator Pattern Definition

Provides a way to access the elements of an **aggregate** object sequentially without exposing its underlying representation

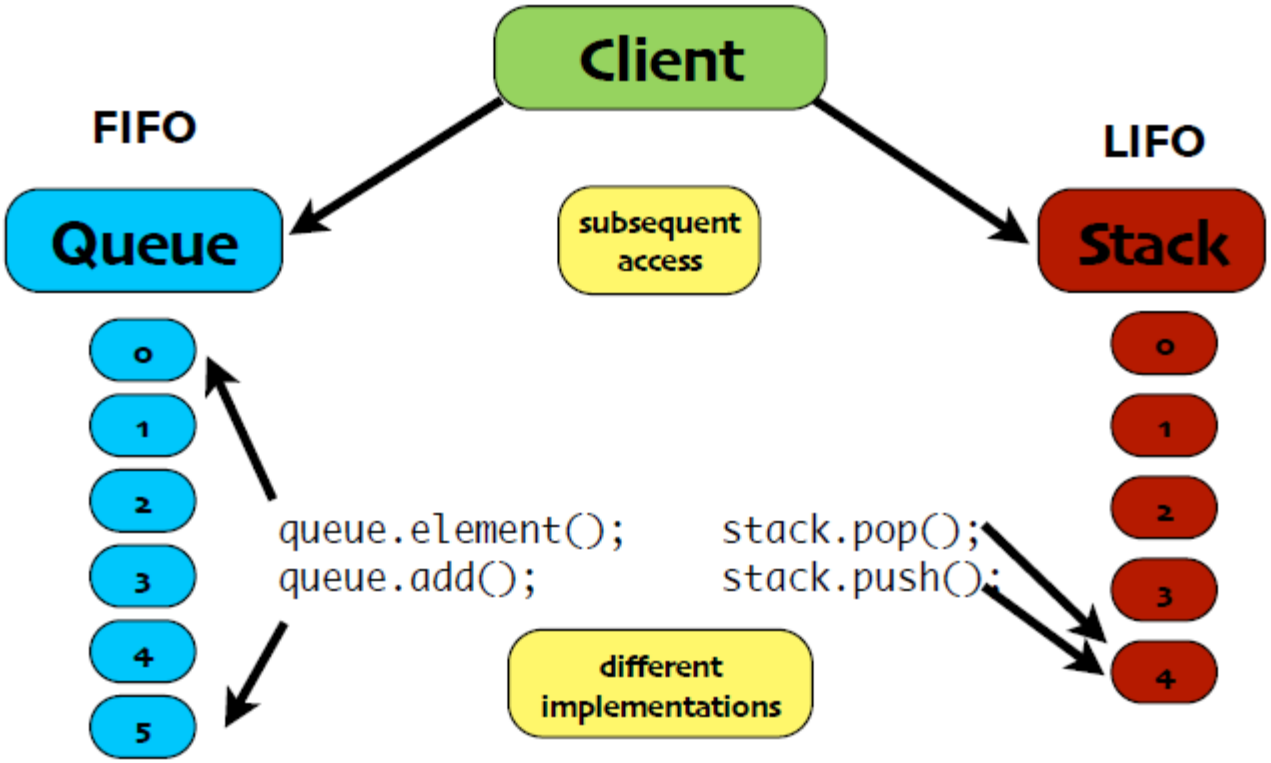
Problem

- Aggregate objects such as something that contains a list internally (or just a list itself!) should allow a way to traverse its elements without exposing its internal structure (encapsulation!)
- We know this kind of thing is supported in Java (and C#) – the enhanced for loop (foreach) takes advantage of this behavior on objects designed with this in mind

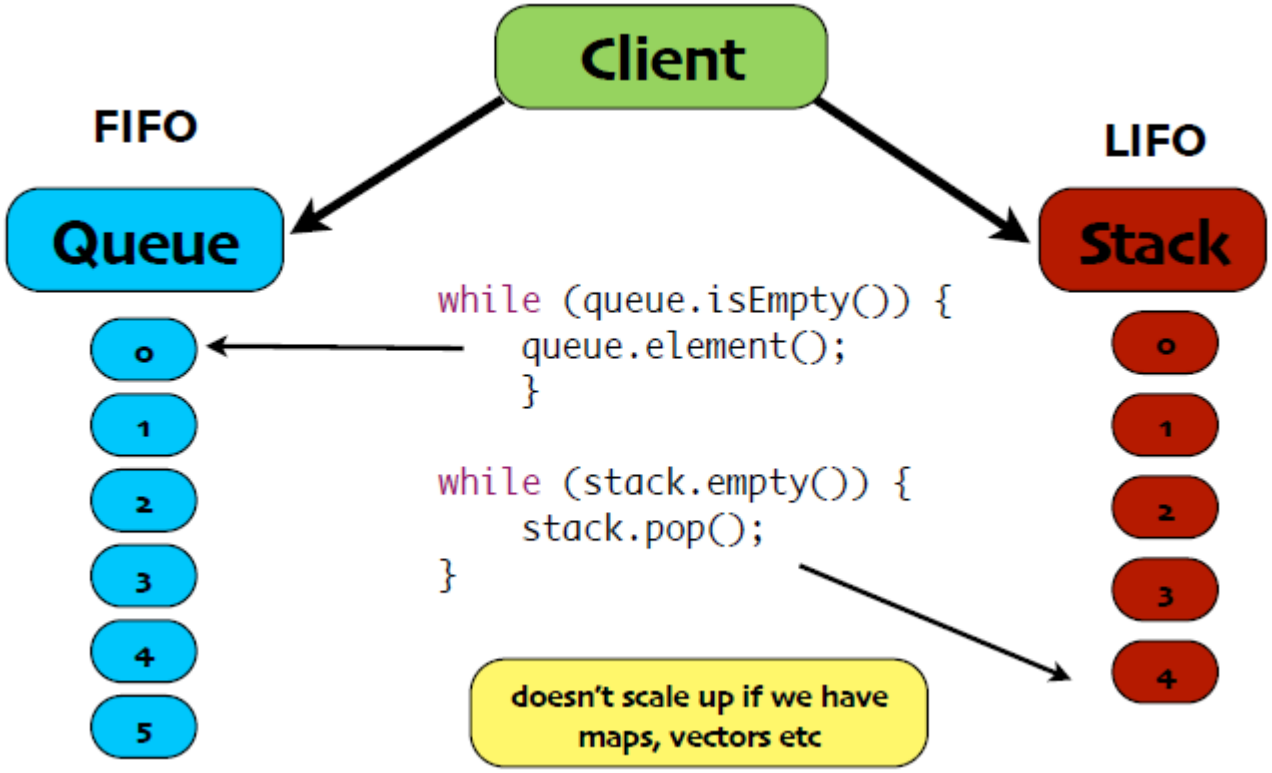
Collection Managers



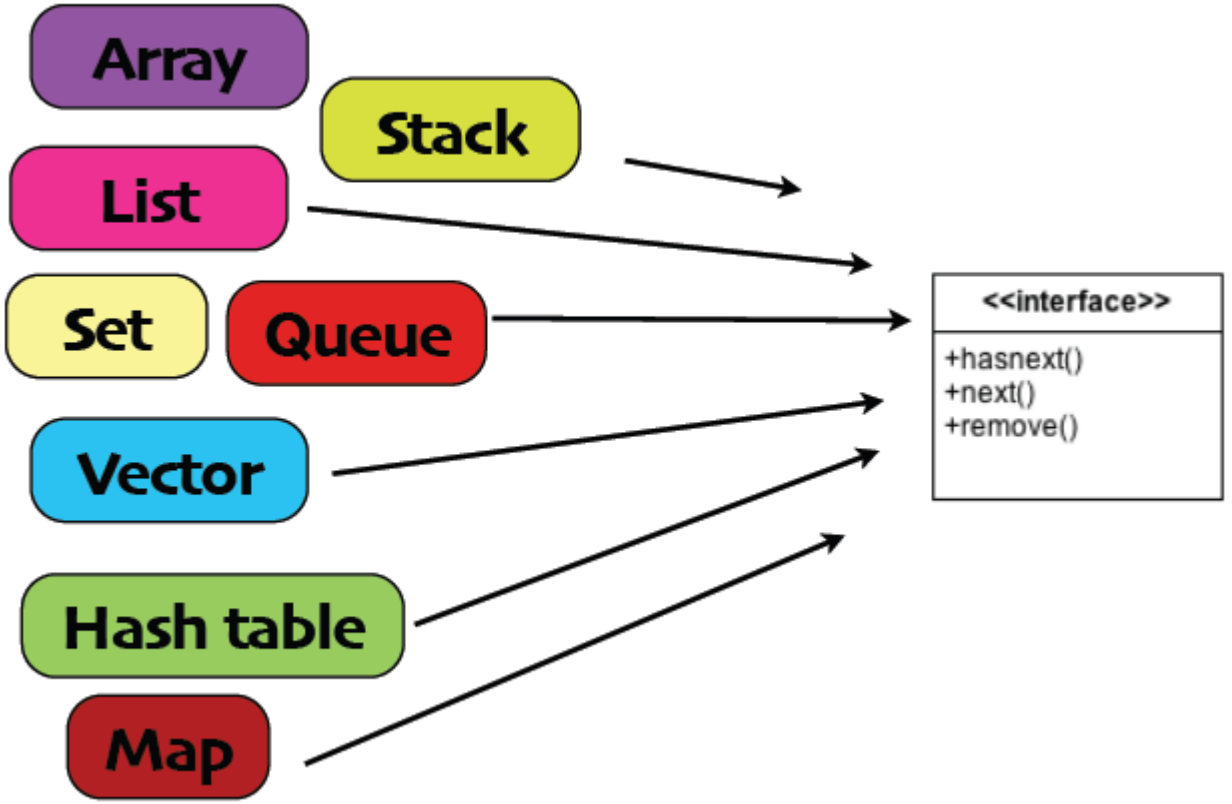
Sample Problem



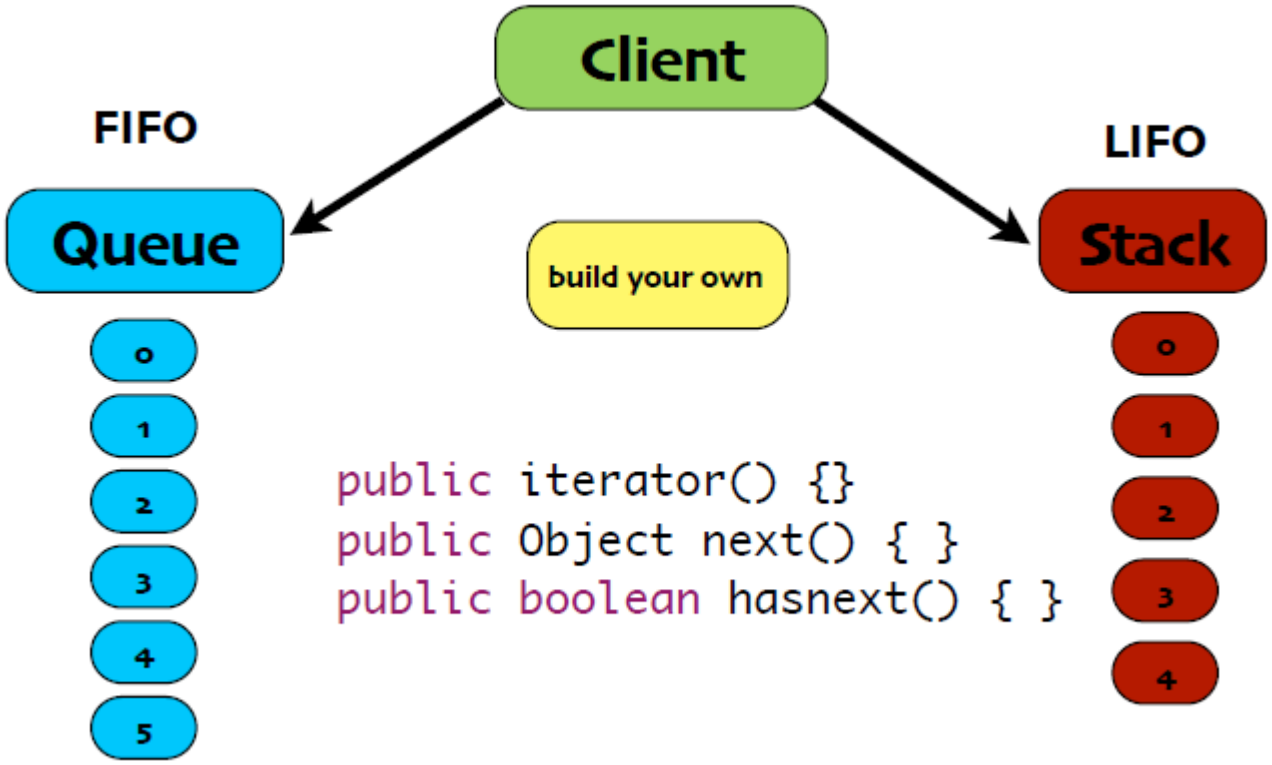
Cumbersome...zzz



One interface to rule them all, one interface to find them, one interface to bring them all and in the API bind them!!!



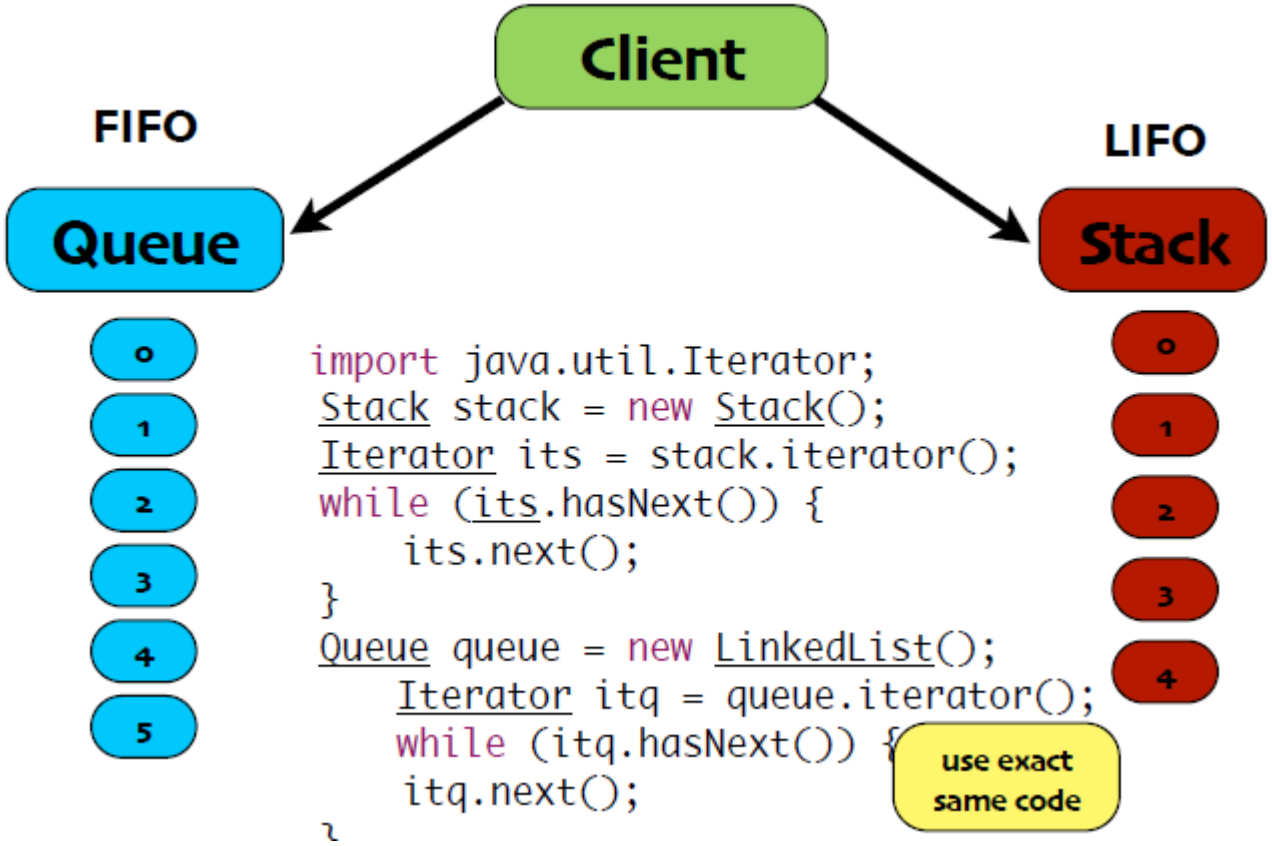
Iterator Approach



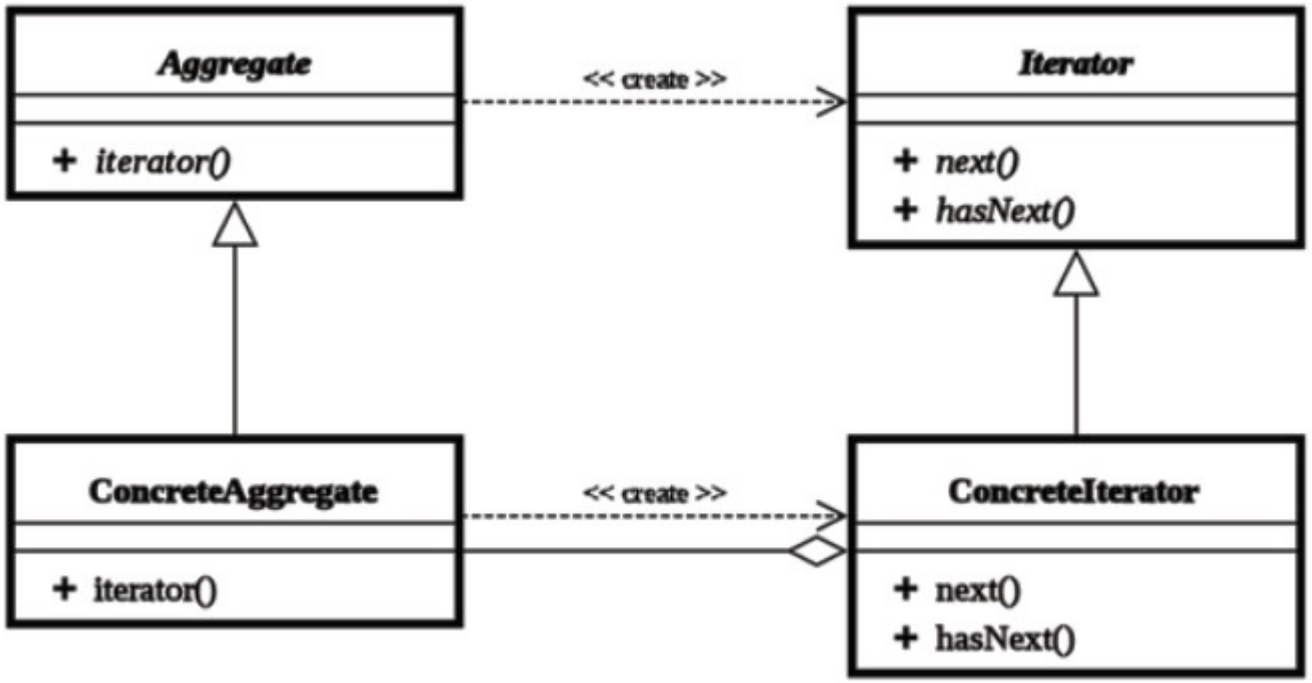
Iterator implementation

- Some class implements Iterator interface (next, hasNext, remove)
- Aggregate class provides a method that generates an iterator object (typically createIterator or iterator)
- Aggregate class can (in Java) implement Iterable interface to guarantee it provides an iterator object for the data it aggregates. This allows an object of the aggregate to be used in a foreach loop
- All classes in Collection (in Java) have the ability to create an iterator

Built in, in Java (and C#)



Class Diagram



Iterator enforces Single Responsibility Principle (SRP)

- SRP: A class should have only one reason to change
- Task of iteration is on the iterator object, not the aggregate, which simplifies the aggregate interface **and** implementation, and places responsibility where it should be

Just Don't Do It!

In Class Team Exercise

- Design a simple class called Course that contains a list of Students (the type of list is up to you :-) – make a **simple** Student class as well
- Course should implement the Iterable interface
- Write an iterator for Course called CourseIterator that implements Iterator
- Write a tester that uses a foreach loop to demonstrate your iterator works
- This assignment may be done using Java or C#
- Submit a zip file with your source files to Blackboard by tomorrow (Wednesday) before noon