# CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2014
Doc 4 Concepts, Trie, Iterator
Feb 4, 2014

# Some Concepts

# Abstraction

“Extracting the essential details about an item or group of items, while ignoring the unessential details.”

Edward Berard

“The process of identifying common patterns that have systematic variations; an abstraction represents the common pattern and provides a means for specifying which variation to use.”

Richard Gabriel

# Encapsulation

Enclosing all parts of an abstraction within a container

# Information Hiding

Hiding of design decisions in a computer program

Hide decisions are most likely to change,
To protect other parts of the program

# Class

Represents an abstraction

Abstraction contains data and operations

Encapsulates data and operations of the abstraction

Hide design decisions/details





Not so much a definition of a class as a goal how we should use a class.

# Metrics for Quality

## Coupling

Strength of interaction between objects in system

## Cohesion

Degree to which the tasks performed by a single module are functionally related

# Coupling

Measure of the interdependence among modules

"Unnecessary object coupling needlessly decreases the reusability of the coupled objects"

"Unnecessary object coupling also increases the chances of system corruption when changes are made to one or more of the coupled objects"

## Design Goal

The interaction or other interrelationship between any two components at the same level of abstraction within the system be as weak as possible

# Disadvantages of Tightly Coupled Systems

A change in one module usually forces a ripple effect of changes in other modules

Assembly of modules might require more effort and/or time due to the increased inter-module dependency

A particular module might be harder to reuse and/or test because dependent modules must be included





Source: http://en.wikipedia.org/wiki/Coupling_(computer_programming)

# Types of Coupling

Nil Coupling

No interaction between two classes

Export Coupling

One class uses the public interface of another

Overt Coupling

One class uses implementation details of another class with permission

Covert Coupling

One class uses implementation details of another class without permission





There are other categories of coupling. See Wikipedia on Coupling

# Polymorphism

Greek - Many Forms

"Providing a single interface to entities of different types"

"Value of different data types can be treated in the same way"

# Types of Polymorphism

Parametric

Inclusion

Overloading

Coercion

# Parametric Polymorphism

Functions or data type written generically
so it can handle values identically without depending on their type

Generics in Java

```
class Node<T> {
    T elem;
    Node<T> next;
}
```

```
boolean add(Node<T> x) {blah}
```

Ruby, Smalltalk, Python, Functional Languages

Don't require the type parameter

Functional programmers call this polymorphism

# Overloading Polymorphism

function or operator overloading

```
public class Node {

    boolean add(Node x) { blah }
    boolean add(String x) { blah }
```

# Coercion Polymorphism

Implicit type conversion

# Inclusion Polymorphism

Also called subtyping or dynamic polymorphism

OO languages call this polymorphism

# Polymorphism

Objects with the same interface can be substituted for each other at run-time

Variables take on many classes of object

Objects will behave according to their type

Code can work with any object that has the right set of methods

In Java polymorphism requires

Inheritance or

Interfaces

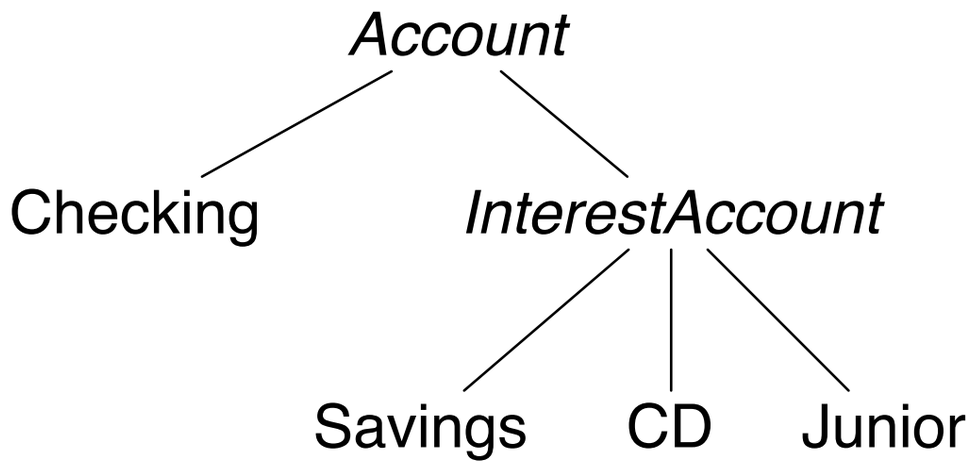
In Smalltalk, Ruby & Objective C polymorphism requires

Objects that implement methods with same name

# Simplistic Example

Bank offers various types of accounts:

- Checking
- Savings
- CD
- Junior savings accounts




Each type has different rules for processing a transaction

# Processing a Transaction

Using Case Statement

```
newCustomer := Bank.createNewAccount(type)


if (newCustomer.isChecking() ) {
     newCustomer.checkTransaction(blah);
}
if (newCustomer.isSavings() ) {
     newCustomer.savingsTransaction(blah);
}
if (newCustomer.isJunior() ) {
     newCustomer.savingsTransaction(blah);
}
etc
```

# Using Polymorphism

```
newCustomer := Bank.createNewAccount(type);
newCustomer.processTransaction(amount);
```

Which processTransaction is called?

Adding new types of accounts to program requires:

- Adding new subclasses
- Changing code that creates objects

Avoid checking the class of an object

# Trie Example

# Trie Assignment - Strawman Solution

Classes with fields

TrieNode

```
char letter;
TrieNode[] childNodes;
boolean isWord;
```

Trie

No Fields

# Trie Assignment - Strawman Solution

Methods TrieNode

None

# Trie Assignment - Strawman Solution

Methods Trie

```
public TrieNode createRoot()
public void insertWord(TrieNode root, String word)
public boolean findWord(TrieNode root, String word)
public void printTrie(TrieNode root)
public void printTrieIngWords(TrieNode root)
```

# Strawman Solution & Abstraction

Data & Operations not together

Fail

# Strawman Solution & Information Hiding

Have to pass data into Trie class

No information hidling

```
public TrieNode createRoot()
public void insertWord(TrieNode root, String word)
public boolean findWord(TrieNode root, String word)
public void printTrie(TrieNode root)
public void printTrieIngWords(TrieNode root)
```

# Trie Assignment - Tinman Solution

Classes with fields

TrieNode

```
char letter;
TrieNode[] childNodes;
boolean isWord;
```

Trie

```
TrieNode root
```

# Trie Assignment - Tinman Solution

Methods TrieNode

Getters & Setters

# Trie Assignment - Tinman Solution

Methods Trie

public void insertWord(String word)
public boolean findWord(String word)
public void printTrie()
public void printTrieIngWords()

# Tinman Solution & Polymorphism

Trie is a collection contains words

Can we replace any Java collection for this Trie class?

No

So fail polymorphism

# How to get Polymorphism

Need to use same method names as in other collection classes

In Java need to implement Collection interface

What should the methods be called?

# Two Issues

```
public void printTrieIngWords() {

    blah
    blah
    blah

        System.out.print(fooBar);
    blah

}
```

# Two Issues

Abstraction

What abstraction does printTrieIngWords belong?

Coupling

printTrieIngWords is coupled to System.out

Not useful

Inflexible

# Solving the Two Issues

Iterators
Visitor Pattern
Strategy Pattern

# Iterator Pattern

Provide a way to access the elements of a collection sequentially without exposing its underlying representation

# Iterator Solution

Java

```
LinkedList<Strings> strings =  new LinkedList<Strings>();

code to add strings

for (String element : strings) {
        if (element.size % 2 == 0)
                System.out.println(element);
}
```

```
Iterator<String> list = strings.iterator();
while (list.hasNext()){
        String element = list.next();
        if (element.size % 2 == 0)
                System.out.println(element);
        }
}
```





This is 1/2 the way to a good solution.

# Ruby Iterator Examples

a = [1, 2, 3, 4]

| | |
|---|---|
| a.each {\|x\| puts x} | 1<br>2<br>3<br>4 |
| result = a.collect {\|x\| x + 10}<br>puts result | 11<br>12<br>13<br>14 |
| result = a.find_all {\|x\| x > 2 }<br>puts result | 3<br>4 |
| puts a.any? {\|x\| x > 2} | true |
| puts a.detect {\|x\| x > 2 } | 3 |





Ruby has a richer set of iterators than Java. Smalltalk, which inspired Ruby's iterators, has a richer set of iterators that Ruby. Perhaps the language that replaces Ruby will match the power that Smalltalk had 20 years ago.

# Pattern Parts

Intent

Motivation

Applicability

Structure

Participants

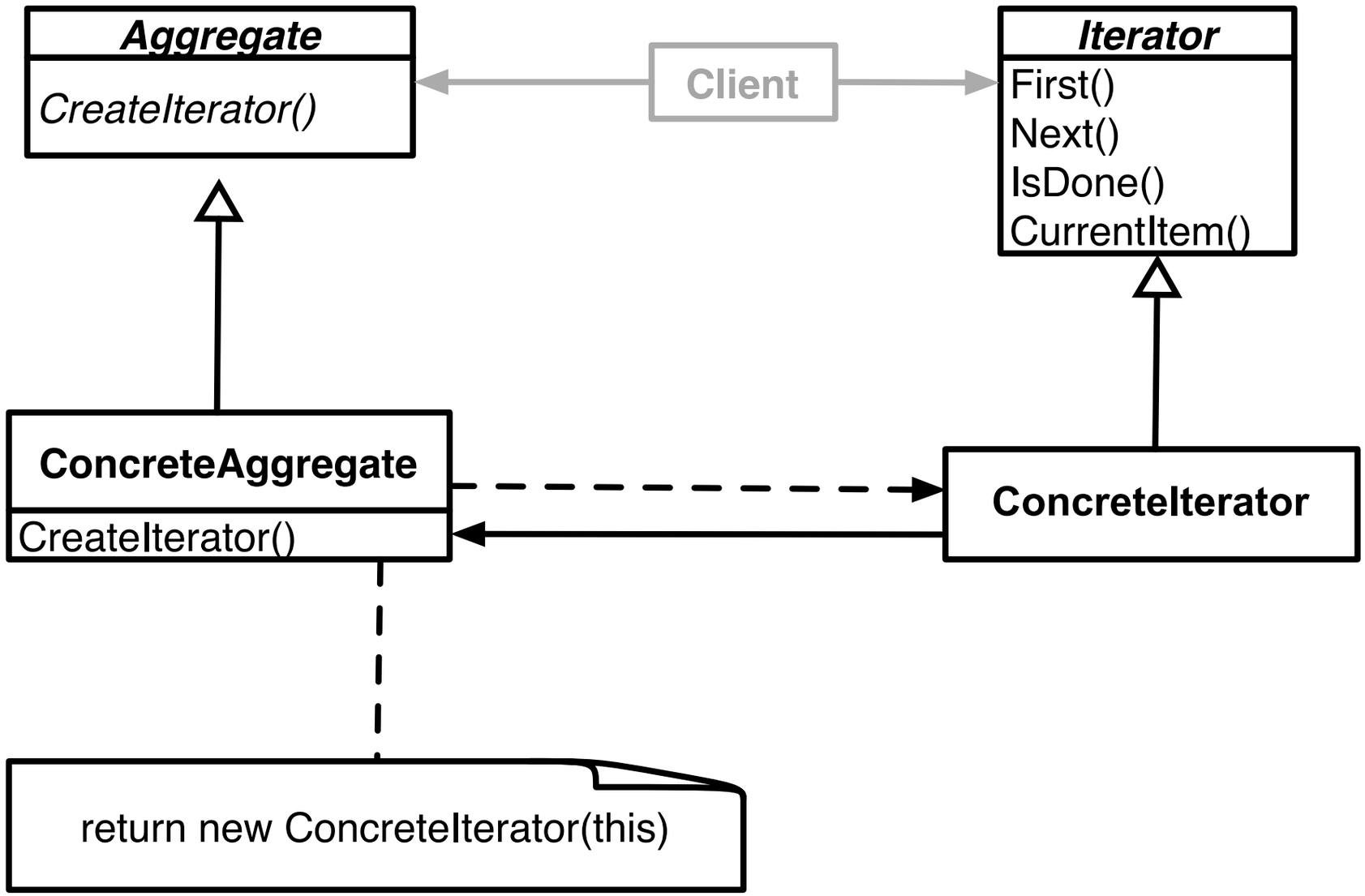
Collaborations

Consequences

Implementation

Sample Code

# Iterator Structure



# Issue - What is the big deal?

```
var numbers = new LinkedList();

code to add numbers

Iterator list = numbers.iterator();
while ( list.hasNext() ) {
        Integer a = (Integer) list.next();
        int    b = a.intValue();
        if ((b % 2) == 0)
                System.out.println( x );
}
```

```
var numbers = new LinkedList();

code to add numbers

for (int k =0; k < numbers.size(); k++ ) {
        Integer a = (Integer) numbers.get(k);
        int    b = a.intValue();
        if ((b % 2) == 0)
                System.out.println( x );
}
```





Java's Enumerations and iterators were awkward to use. C# pushed Sun to add better syntax.

# Issues - Concrete vs. Polymorphic Iterators

Concrete

```
Reader iterator = new StringReader( "cat");
int c;
while (-1 != (c = iterator.read() ))
      System.out.println( (char) c);
```

Polymorphic

```
Vector listOfStudents = new Vector();

// code to add students not shown

Iterator list = listOfStudents.iterator();
while ( list.hasNext() )
      System.out.println( list.next() );
```

Memory leak issue in C++, Why?

# Issue - Who Controls the Iteration?

External (Active)

```
var numbers = new LinkedList();

code to add numbers

Vector evens = new Vector();
Iterator list = numbers.iterator();
      while ( list.hasNext() ) {
            Integer a = (Integer) list.next();
            int    b = a.intValue();
 if ((b % 2) == 0)
                  evens.add(a);
}
```

Internal (Passive)

```
numbers  = LinkedList.new

code to add numbers

evens = numbers.find_all { |element| element.even? }
```

# Issue - Who Defines the Traversal Algorithm

Object being iterated

Iterator

# Issue - Robustness

What happens when items are added/removed from the iteratee while an iterator exists?

```
Vector listOfStudents = new Vector();

// code to add students not shown

Iterator list = listOfStudents.iterator();
listOfStudents.add( new Student( "Roger") );

list.hasNext();              //What happens here?
```

# Are Java's Input Streams & Readers Iterators?

# Pipes and Filters

# Pipes & Filters

ls | grep -i b | wc -l

## Context

Processing data streams

## Problem

Building a system that processes or transforms a stream of data

## Forces

Small processing steps are easier to reuse than large components

Non-adjacent processing steps do not share information

System changes should be possible by exchanging or recombining processing steps, even by users

Final results should be presented or stored in different ways

# Solution

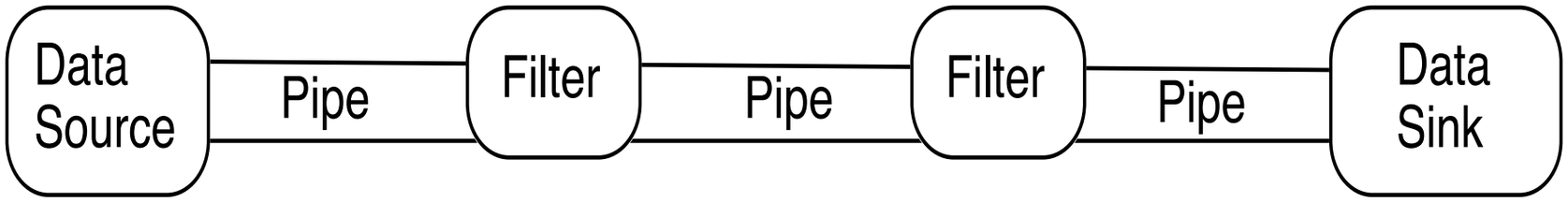
Divide task into multiple sequential processing steps or filter components

Output of one filter is the input of the next filter

Filters process data incrementally

Filter does not wait to get all the data before processing

# Solution Continued




Data source – input to the system

Data sink – output of the system

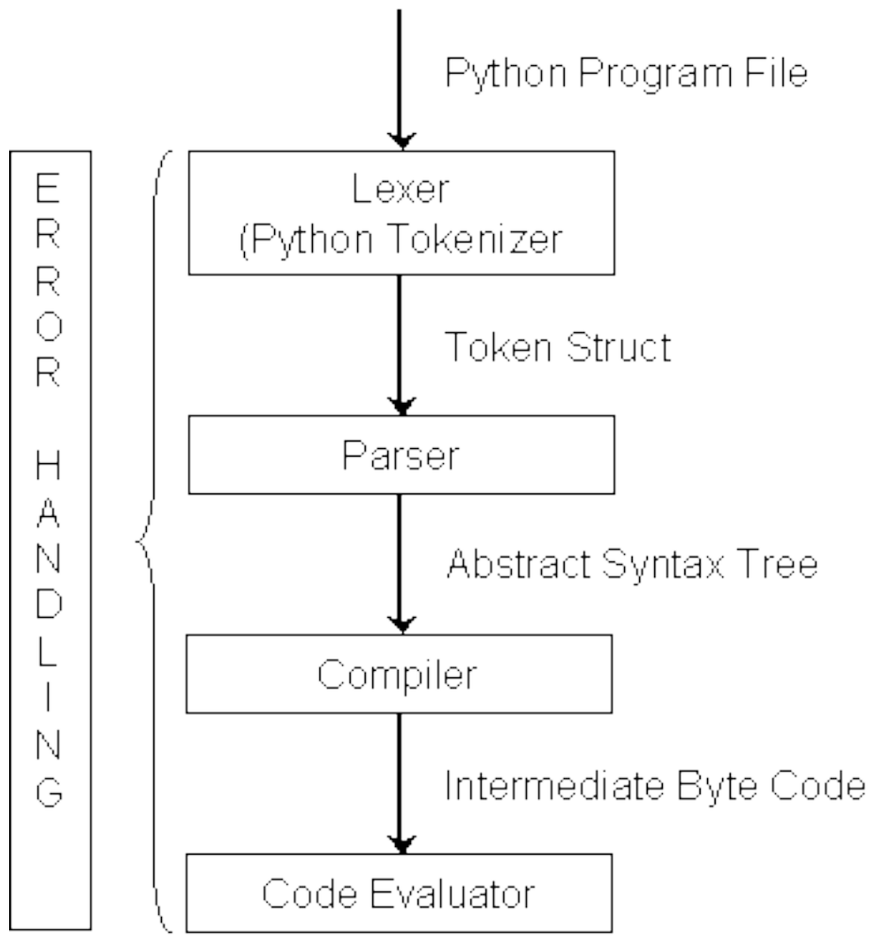
Pipes - connect the data source, filters and data sink

Pipe implements the data flow between adjacent processes steps

Processing pipeline – sequence of filters and pipes

Pipeline can process batches of data

# Python Interpreter




http://wiki.cs.uiuc.edu/cs427/Python+-+Batch+Sequential

# Intercepting Filter - Problem

Preprocessing and post-processing of a client Web request and response

A Web request often must pass several tests prior to the main processing

- Has the client been authenticated?
- Does the client have a valid session?
- Is the client's IP address from a trusted network?
- Does the request path violate any constraints?
- What encoding does the client use to send the data?
- Do we support the browser type of the client?

Nested if statements lead to fragile code

# Intercepting Filter - Forces

Common processing, such as checking the data-encoding scheme or logging information about each request, completes per request.

Centralization of common logic is desired.

Services should be easy to add or remove unobtrusively without affecting existing components, so that they can be used in a variety of combinations, such as

Logging and authentication

Debugging and transformation of output for a specific client

Uncompressing and converting encoding scheme of input