CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2001
Doc 6 Iterators

Contents
Iterator ................................................................................................................................. 2
Issues .................................................................................................................................. 5
  Concrete vs. Polymorphic Iterators .................................................................................. 5
  Who Controls the iteration? .............................................................................................. 6
  Who Defines the Traversal Algorithm? ............................................................ ................. 8
  How Robust is the iterator? ............................................................................................. 9
  Iterators and Privileged Access ...................................................................................... 10
  Iterators for Composites ................................................................................................. 11
  Null Iterator ..................................................................................................................... 11
Exercises ................................................................................................................................ 12

References
Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, 1995, pp. 257-271

Reading
Design Patterns: pp. 257-271

Copyright ©, All rights reserved. 2001 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.
Iterator

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

Java Examples

Enumeration, Iterator, and Streams in Java are iterators.

```java
Vector listOfStudents = new Vector();
// code to add students not shown

Iterator list = listOfStudents.iterator();
while ( list.hasNext() )
    Console.println( list.next() );

Hashtable anIndex = new HashMap();
// code to add elements to the hashMap not shown

Iterator list = anIndex.values().iterator();
while ( list.hasNext() )
    Console.println( list.next() );
```
Smalltalk Examples

Streams, do:, select:, reject:, collect:, detect:, inject:into: are iterators in Smalltalk

| sum  |
sum := 0.
#( 1 7 2 3 9 3 50) do: [:each | sum := sum + each squared].
^sum

#( 1 7 2 3 9 3 50) do:
  [:partialSum :number | partialSum + number squared]

'this is an example' select: [:each | each isVowel ]
Sample Implementation of Java Enumerator

class VectorIterator implements Enumeration {
    Vector iteratee;
    int count;

    VectorIterator(Vector v) {
        iteratee = v;
        count = 0;
    }

    public boolean hasMoreElements() {
        return count < iteratee.elementCount;
    }

    public Object nextElement() {
        synchronized (iteratee) {
            if (count < iteratee.elementCount)
                return iteratee.elementData[count++];
            }
            throw new NoSuchElementException("VectorIterator");
        }
    }

    The iterator is using privileged access to Vectors fields
Issues
Concrete vs. Polymorphic Iterators
Concrete

Use Explicit Iterator Type

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

Polymorphic

Actual type of iterator is not known

Vector listOfStudents = new Vector();

// code to add students not shown

Iterator list = listOfStudents.iterator();
while ( list.hasNext() )
    Console.println( list.next() );

Polymorphic iterators can cause problems with memory leaks in C++ because they are on the heap!
Who Controls the iteration?
External (Active)

Vector listOfStudents = new Vector();

// code to add students not shown

Iterator list = listOfStudents.iterator();

while ( list.hasNext() )
    Console.println( list.next() );
**Who Controls the iteration?**

**Internal (Passive)**

'this is an example' select: [:each | each isVowel ]

Vector listOfStudents = new Vector();

// code to add students not shown

while (listOfStudents.hasMoreElements())
    Console.println( listOfStudents.nextElement() );

---

**Fictitious Code**

class Vector implements Cloneable, java.io.Serializable {
    protected Object elementData[];
    protected int elementCount;
    protected int currentPosition;

    public boolean hasMoreElements() {
        return currentPosition < elementCount;
    }

    public Object nextElement() {
        if (currentPosition < elementCount)
            return elementData[currentPosition++];
        throw new NoSuchElementException("VectorIterator");
    }
    // all vector methods not shown
}
Who Defines the Traversal Algorithm?  
Object being Iterated

Iterator can store where we are

In a Vector this could mean the index of the current item

In a tree structure it could mean a pointer to current node and stack of past nodes

BinaryTree searchTree = new BinaryTree();

// code to add items not shown

Iterator aSearch = searchTree.getIterator();
Iterator bSearch = searchTree.getIterator();
Object first = searchTree.nextElement( aSearch );
Object stillFirst = searchTree.nextElement( bSearch );

Iterator

Makes it easier to have multiple iterator algorithms on same type

On Vector class, why not have a reverseliterator which goes backwards?

In a complex structure the iterator may need access to the iteratee's implementation
**How Robust is the iterator?**

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

```java
Vector listOfStudents = new Vector();

// code to add students not shown

Enumeration list = listOfStudents.elements();
Iterator failFastList = listOfStudents.iterator();

listOfStudents.add( new Student( "Roger") );

list.hasMoreElements();
failFastList.hasNext(); //Exception thrown here
```
Additional Iterator Operations

Augmenting basic iteration operations may improve their usefulness

\begin{itemize}
  \item \textbf{previous()}
    \begin{itemize}
      \item back up one location
    \end{itemize}
  \item \textbf{add( Object item)}
    \begin{itemize}
      \item add item to the iteratee at current location
    \end{itemize}
  \item \textbf{remove()}
    \begin{itemize}
      \item remove the current item from the iteratee
    \end{itemize}
  \item \textbf{skipTo( some location, item or condition )}
    \begin{itemize}
      \item go to the location indicated
    \end{itemize}
  \item \textbf{mark()}
    \begin{itemize}
      \item mark current location for future return
    \end{itemize}
\end{itemize}

Iterators and Privileged Access

An iterator may need privileged access to the aggregate structure for traversal
Iterators for Composites

Traversing a complex structure like a graph, tree, or composite can be difficult

An internal iterator can use recursion to keep track of where to go next

For example using a depth-first search algorithm on graph

If each element in the aggregate “knows” how to traverse to the next element and previous element, than an external iterator can be used

Null Iterator

A Null iterator for the empty aggregates can be useful
Exercises

1. Explain why polymorphic iterators in C++ must be on the heap?

2. What are the major consequences of the iterator pattern?

3. Contrast Java and Smalltalk (C++) iterators. What are the strengths and weaknesses of each.