Iterators

October 7 2002

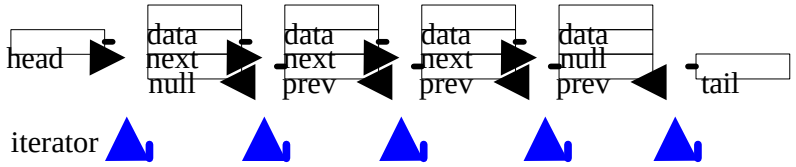
Reading Assignment
Chapter 4

Iterator pattern

- An iterator is an object that allows us to enumerate or go through all the elements of a collection or a data structure
- An iterator object controls iteration of the elements of a collection
- The Java collections API uses the iterator pattern extensively
- What if the collection is modified while an iterator is in use?
  - copy: iterator operates on a snapshot taken at instantiation
  - fail-fast: iterator throws an exception on the next method call
  - flexible: iterator tries to adjust to the changes in a manner that varies with the collection type

Implementation of iterators

- An iterator object holds a pointer into the data structure
- The pointer is always between elements, that is, between the element returned last and the element to be returned next
- Thus, for a list the iterator pointer is initialized with **`head`** or **`tail`** when the iterator is constructed and advanced with the **`next()`** or **`nextElement()`** call
- Since the iterator is defined as a class, multiple iterators can be instantiated from it and hence multiple iterators can be used concurrently on the same data structure



Implementation of iterators

- Iterators can be defined separately from the data structure class or as an inner class of the data structure class
- If an iterator is defined as a separate class, then except for naming conventions and passing the data structure to the constructor as a parameter, it is not clear to which data structure it belongs. Also too much information of the implementation is exposed.
- A better strategy is to define the iterator as an inner class of the data structure. As a result, the iterator is intimately tied to the data structure and the implementation details are nicely hidden (i.e., information hiding software engineering principle is followed).
- We first present the better solution, the inner class solution and then the separate class solution.

Defining an inner class iterator

```
public class LinkedList {
  private class LocalIterator implements Enumeration {
    private Node curNode;
    public LocalIterator() {
      curNode = head;
    }
    public boolean hasMoreElements() {
      return curNode != null;
    }
    public Object nextElement() {
      Object c = curNode;
      curNode = curNode.getNext();
      return ((Node)c).getData();
    }
  }
  public Enumeration iterator() {
    return new LocalIterator();
  }
  // other fields and methods of class LinkedList
}
```


Really nice thing about this is that you can have multiple hidden iterators for a class!

Using the inner class iterator

```
Stack s = new LinkedStack();
s.push(new Integer(4)); s.push(new Integer(7));
s.push(new Integer(9)); s.push(new String("CSc115"));
s.push(new String("CSc160")); s.push(new Double(3.14));
Enumeration litr = ((LinkedList)s).iterator();
while (litr.hasMoreElements()) {
  Object obj = litr.nextElement();
  if (obj instanceof Integer) {
    int k = ((Integer)obj).intValue();
    System.out.println("litr " + k);
  } else if (obj instanceof String) {
    String str = (String)obj;
    System.out.println("litr " + str);
  } else {
    System.out.println("litr unknown");
  }
}
```

Defining a separate class iterator

```
public class LinkedListFwIterator {
  private LinkedList ll;
  private Node curNode;

  public LinkedListFwIterator(LinkedList ll) {
    this.ll = ll;
    this.curNode = ll.getHead();
  }
  public boolean hasNext() {
    return curNode != null;
  }
  public Object next() {
    Object c = curNode;
    curNode = curNode.getNext();
    return ((Node)c).getData();
  }
  public void remove() {
    ll.delete(curNode);
  }
}
```

Using the separate class iterator

```
LinkedListFwIterator gitr =
  new LinkedListFwIterator((LinkedList)s);
while (gitr.hasNext()) {
  Object obj = gitr.next();
  if (obj instanceof Integer) {
    int k = ((Integer)obj).intValue();
    System.out.println("gitr " + k);
  } else if (obj instanceof String) {
    String str = (String)obj;
    System.out.println("gitr " + str);
  } else {
    System.out.println("gitr unknown");
  }
}
```

- In Eclipse: LinkedList.java (look at Dnode.java also)
- Exercise: With a partner read the code just handed out and make sure you understand it. Write on the sheet how you would add another iterator to the class which does a fifo iteration on the list.