Design Patterns for Iteration

- Iterators
  - purpose
  - example use
  - implementation for pointer-based & array-based data structures
- Concurrent Modification
  - why it's a problem
  - detection with modification counts
- Locators — robust iterators
  - The Valet Parking Problem
  - Markers
  - Trackers
- Visitors

Discuss Lab 2 Handout
Iterators
  • efficient traversal of a structure
  • hide internal rep.

class Foo<T> implements Iterable<T>
  public Iterator<T> iterator()
  }
  return new FooIterator();
}

// other stuff

f = new Foo<String>();
Iterator<String> it = f.iterator();
while (it.hasNext()) {
  String s = it.next();
  doSomethingWith(s);
}

\[ \Rightarrow \]
for (String s : f)
  doSomethingWith(s);
in a LinkedList (doubly linked, dummy/sentinel head & tail)

inner class in the LinkedList class:

```java
class MyListIterator<E> implements Iterator<E>
{
    ListItem currentPosition = head;
    public boolean hasNext() {
        return currentPosition.next != tail;
    }

    public E next() {
        if (hasNext()) {
            currentPosition = currentPosition.next;
            return currentPosition.value;
        } else {
            throw new NoSuchElementException();
        }
    }

    public void remove() {
        if (currentPosition is not head or tail)
            remove (currentPosition);
    }
```

This would splice the list item out of the list (without changing next in the list item)
1. Use previous ptr == null to indicate a deleted list item
2. advance in hasNext until reaching a non-deleted item (or fail)
In an array-based implementation, an iterator keeps an index of the current position.

```java
String[] a = {"a", "b", "c", "d", "e", "f"};
int index = 2;

// Start iterator at index
hasNext: true when index < size - 1
next:
if (hasNext()) {
    index++;
    return a[index];
}

Collection has:
remove(index)

check if not at -1 + that next has been called since last call to remove
remove (index), then index --j
```
index = 2

remove()

next() ⇒
Java's solution to concurrent modifications:

"Fail-fast"

get a ConcurrentModificationException if
try to use an iterator on a collection
that has been modified

Need a detection mechanism

① let the collection keep a modification count
② create iterator — save copy of current
    modification count in iterator
③ when try to use iterator, it
    compares its count to the one in data structure
    if iterator's is less → iterator is stale
    throw ConcurrentModificationException
The Valet Parking Problem:

1. Cars go in and out but don't switch spaces
   - Envelope is a "Marker" — Marks a particular spot (like earlier array-based iterator)
2. Cars get shifted around over time
   - Envelope is a "Tracker" — Keeps track of where the original element is (like earlier linked list iterator)
How would you implement a tracker for an array?

Tracker

ptr to container for its element (container knows its index)

If containers move, update their indices
With iterators, the control flow (loop) is external to the data structure; data structure provides a way to access its internal structure.

With visitors, the control flow is inside the data structure.