CS1083 Week 11: LinkedList and Iterators.

David Bremner

2018-03-21

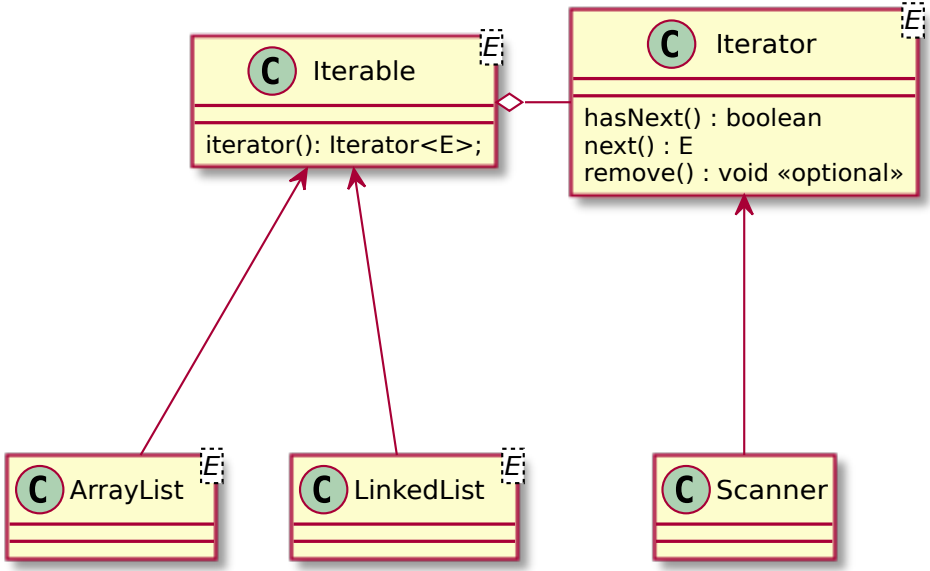
Outline

Iterators

Collections

Using LinkedList

Iterator and Iterable



Using iterators

InsertionSort

```
while ( sc.hasNext() ){
    sortedInsert(sc.next() ,list);
}

ListIterator<String> iterator=list.listIterator();
while (iterator.hasNext()){
  System.out.println(iterator.next());
}
```

Foreach and iterable

- recall Java "foreach" statement

```
for (E element : collection) {...}
```

- The condition to use a `foreach` is exactly implimenting the Iterable interface.

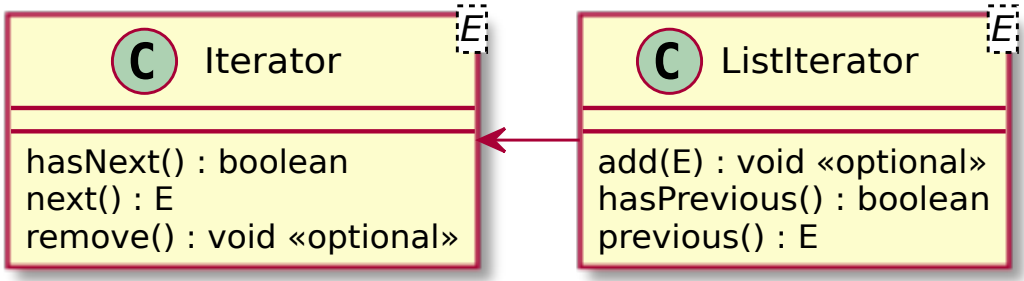
InsertionSort2

```
LinkedList<String> list=new LinkedList<>();
Scanner sc=new Scanner(System.in);
while ( sc.hasNext() )
    sortedInsert(sc.next() ,list);

for (String element : list)
    System.out.println(element);
```

ListIterator

```
                Element(0)   Element(1)    ... Element(n-1)
cursor :   ^              ^              ^      ^
```



Using ListIterator

No direct list operations:

InsertionSort

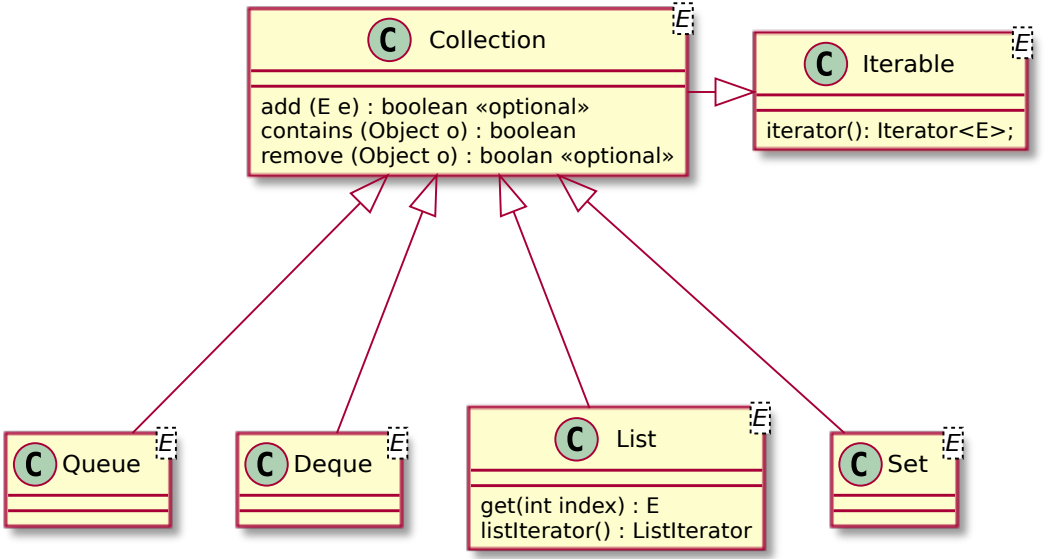
```
ListIterator<String> iterator=list.listIterator();
while (iterator.hasNext() ){
  String element=iterator.next();
  if (element.compareTo(input) >0){
    iterator.previous();
    iterator.add(input);
    return;
  }
}
iterator.add(input);
```

Iterators

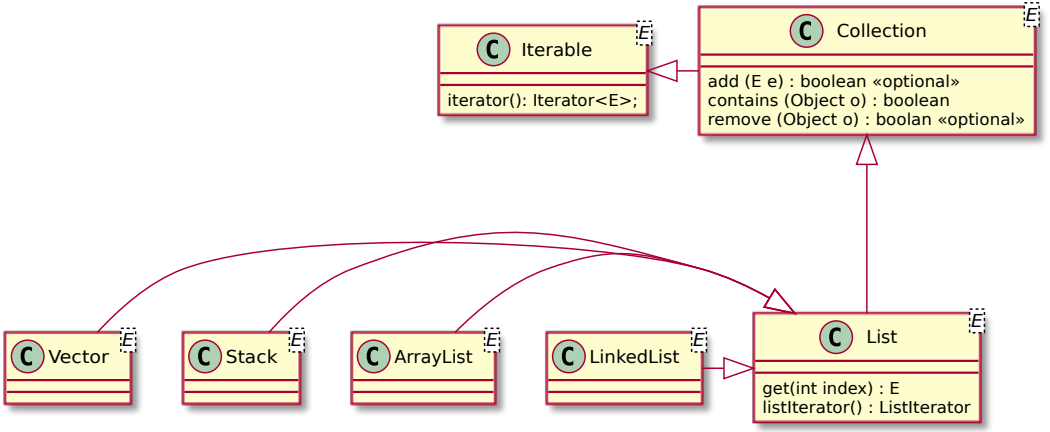
Collections

Using LinkedList

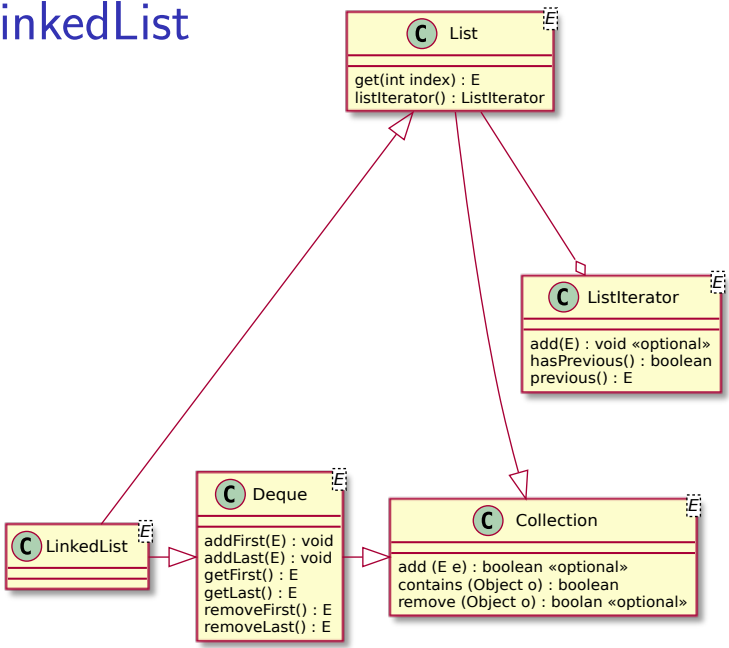
Collection


Collection
E
add (E e) : boolean «optional»
contains (Object o) : boolean
remove (Object o) : boolan «optional»
Iterable
E
iterator(): Iterator<E>;
Queue
E
Deque
E
List
E
get(int index) : E
listIterator() : ListIterator
Set
E

List


Iterable
E
iterator(): Iterator<E>;
Collection
E
add (E e) : boolean «optional»
contains (Object o) : boolean
remove (Object o) : boolan «optional»
Vector
E
Stack
E
ArrayList
E
LinkedList
E
List
E
get(int index) : E
listIterator() : ListIterator

LinkedList



- the familiar linked list operations are actually inherited from the Deque interface

Iterators

Collections

Using LinkedList

Doubly Linked Example, revisited.

DoubleList2

```
DoubleList<String> test=new DoubleList<>();
test.insertLast("hello");
test.insertFirst("goodbye");
test.insertLast("are you still here");

System.out.println("\nForward:");
test.print();

System.out.println("\nReverse:");
test.printReverse();
```

⋮

- re-impliment using LinkedList

DoubleList2 design

- what can we inherit from LinkedList?
- how do we have to extend the class, to match our previous example?