CSSE 220 Day 26

Linked List Implementation

Checkout *LinkedLists* project from SVN

Questions

Data Structures

- Understanding the engineering trade-offs when storing data

Data Structures

- Efficient ways to store data based on how we'll use it

- The main theme for the rest of the course

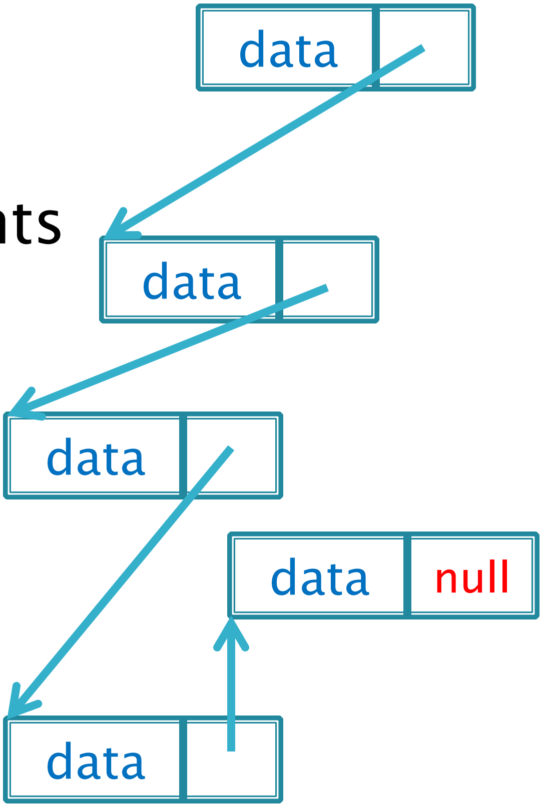
- So far we've seen `ArrayLists`
  - Fast addition **to end of list**
  - Fast access to any existing position
  - Slow inserts to and deletes from middle of list

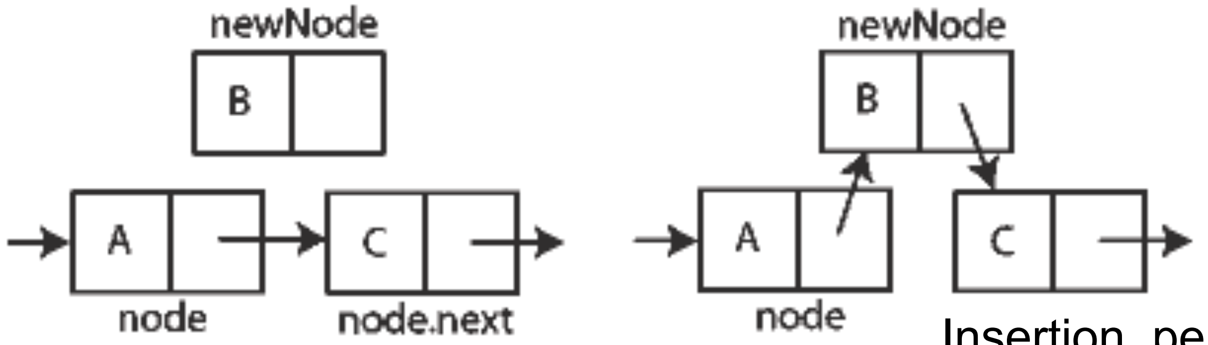
Q1

Another List Data Structure

- What if we have to add/remove data from a list frequently?
- `LinkedLists` support this:
  - Fast insertion and removal of elements
    - Once we know where they go
  - Slow access to arbitrary elements

"random access"





Insertion, per Wikipedia

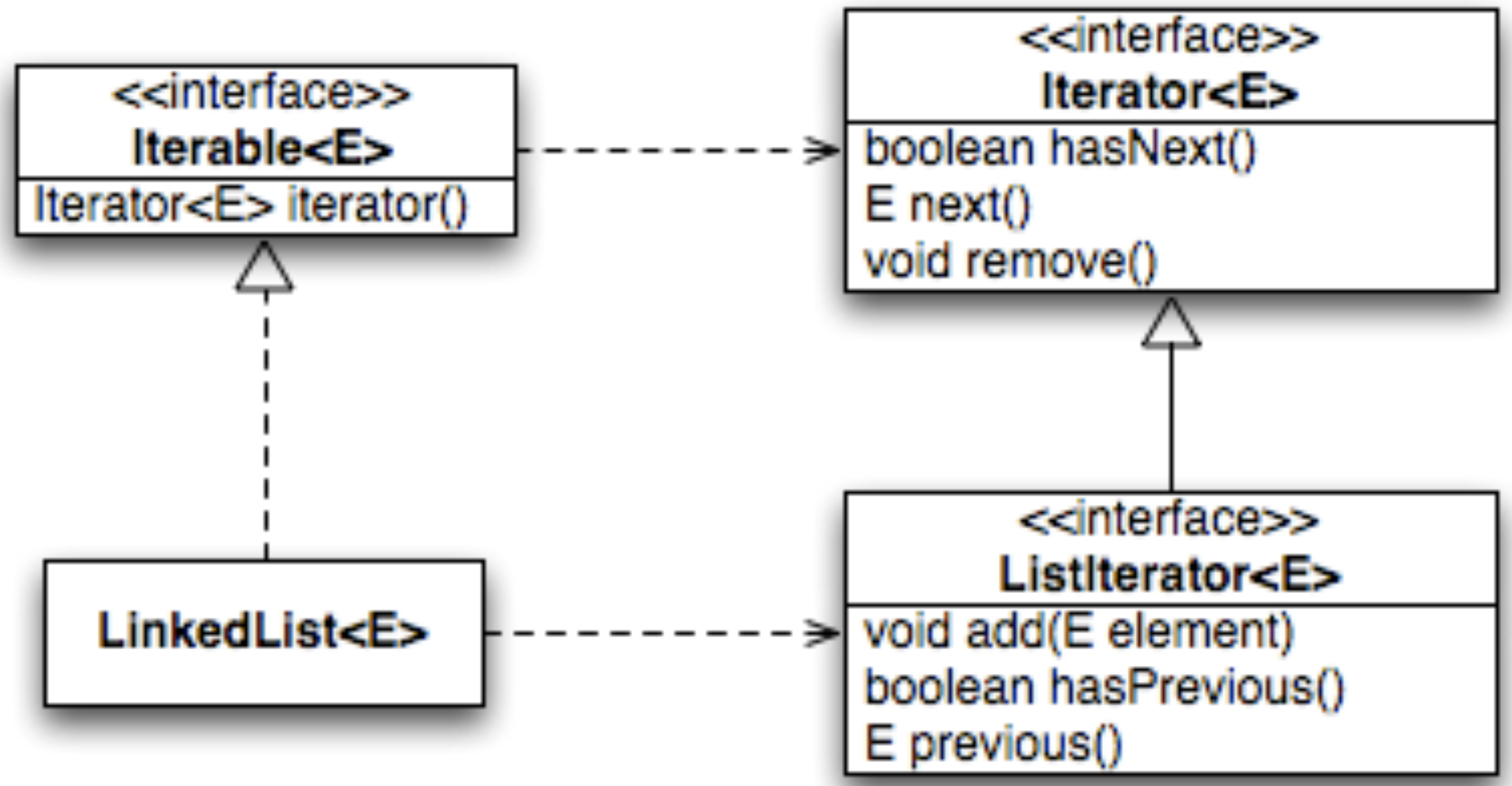
Q2, Q3

LinkedList<E> Methods

- `void addFirst(E element)`
- `void addLast(E element)`
- `E getFirst()`
- `E getLast()`
- `E removeFirst()`
- `E removeLast()`

- What about accessing the middle of the list?
  - `LinkedList<E> implements Iterable<E>`

Accessing the Middle of a LinkedList



An Insider's View

```
for (String s : list) {
  // do something
}
```

```
Iterator<String> iter =
    list.iterator();

while (iter.hasNext()) {
  String s =
iter.next();
  // do something
}
```

Enhanced For Loop

What Compiler Generates

Implementing LinkedList

- A simplified version, with just the essentials
- Won't implement the java.util.List interface
- Will have the usual linked list behavior
  - Fast insertion and removal of elements
    - Once we know where they go
  - Slow random access

Team Project Work Tine

- LodeRunner next cycle due next class