CSCI 136
Data Structures &
Advanced Programming

Jeannie Albrecht
Lecture 19
April 7, 2014

Administrative Details

- Lab 6 due tomorrow
- At least one TA will be around tonight (probably around 8ish)
- 134 will be in the main lab
- Handout: Lab 7
- You’ll get Midterm 1 back in a bit

Looking ahead
- Labs 8 and 9 are the most challenging (but fun!) labs of the semester
- Midterm 2 is April 30
- Check for conflicts and let me know!

Last Time

- Finished discussing queues
- Talked about how queues are used in network routers for buffering packets

Today’s Outline

- Begin discussing iterators (Ch 8)
- Maybe begin thinking about ordered structures (Ch 11)
- FYI, we have now covered Chapters 1-11

Review:
Common Structure Operations

- size()
- isEmpty()
- add()
- remove()
- clear()
- contains()

- What’s missing?
  - Method for efficient data traversal
  - iterator()

Visiting Data from Structure

- Write a method (numOccurs) that counts the number of times a particular Object appears in a structure

```java
public int numOccurs (List data, Object o) {
    int count = 0;
    for (int i=0; i<data.size(); i++) {
        Object obj = data.get(i);
        if (obj.equals(o)) count++;
    }
    return count;
}
```

- Does this work on all structures (that we have studied so far)?
Problems

- get() not defined on Linear structures (i.e., stacks and queues)
- get() is “slow” on some structures
  - O(n) on SLL (and DLL)
  - So numOccurs = O(n^2)
- How do we process data in structures in a general, efficient way?
  - Must be data structure-specific for efficiency
  - Must always use some interface to make general

Iterators

- **Iterators** provide us with a way to efficiently cycle through elements of a data structure
- An Iterator:
  - Provides generic methods to traverse elements
  - Abstracts away details of how to access structure
  - Uses different implementations for each structure
  - As usual, we use both an Iterator interface and an AbstractIterator class

Implementations

- Iterator interface defines next(), hasNext(), and reset() (remove() is actually optional)
  - Works for all structures!
- All specific implementations in structure5 extend AbstractIterator (which implements Iterator)
  - We need to define the methods labeled “abstract” for each data structure (i.e., get(), next(), hasNext(), and reset())
  - Methods are specialized for specific data structures
  - Example: SLL

Rewriting numOccurs

```java
public int numOccurs (List data, Object o) {
    int count = 0;
    Iterator iter = data.iterator();
    while (iter.hasNext()) {
        if (o.equals(iter.next())) count++;
    }
    return count;
}
```

More Iterator Examples

- How would we implement VectorIterator?
- How about StackArrayIterator?
  - Do we go from bottom to top, or top to bottom?
  - Doesn’t matter! We just have to be consistent…
- We can also make “specialized iterators” (we’ll look at these next time…)
  - Another SLL Example (SpecialIterator.java)
  - TestIterator.java
General Rules for Iterators

1. Understand order of data structure
2. **Always call hasNext() before calling next()!!!**
3. Never change underlying data structure while iterating over it

- Take away messages:
  - Iterator objects capture state of traversal
  - They have access to internal data representations
  - Should be fast and easy to use