# Intro to Data Structures

Lecture #15 – Iterators
October 21, 2014

Mark Stehlik

# Outline for Today

- HW4 due Wed (early bonus due tonight)
- HW5 (and Quiz 4) on Thursday
- ArrayLists/LinkedLists and the *List* interface
- Iterators
- Demo HW5

# Interface Review

- An *interface* specifies a set of (public abstract) methods that define an abstract object
- As a result, an interface contains no method implementations (not even constructors)
- Other classes implement the interface; if a class advertises that it implements an interface, it MUST implement all methods that are in the interface (and can implement other methods as well)
- Classes that implement the interface need to advertise that fact (and can implement multiple interfaces)

# ArrayLists and LinkedLists

- Java provides two flavors of *List* (an interface)
  - ArrayList
  - LinkedList (just like the one you've been working on, except…)
  - Lists are part of the Java Collections hierarchy (API)
- How are ArrayList/LinkedList similar/different?
  - operations
  - running time

# Efficiency question

```
public static void print(List<String> l)
{
    for (int i = 0; i < l.size(); i++)
        System.out.print(l.get(i) + " ");  //hidden cost!
    System.out.println();
}
```

Running time?  O(??)

# Efficiency check

- So that's a problem…
- There has to be a better way…
- Iterators
  - use (the API)
  - implementation

# Iterator implementation

- Let's look at two interfaces, *Iterable* and *Iterator*
- Implementing *Iterable* means that the class needs to provide an instance of an *Iterator*
- It also means an object of that class can be the target of an enhanced for loop (we'll talk about that more in a little bit)
- Let's see how an iterator can impact our "printing problem"

# Efficiency check (using an iterator)

```
public static void print(List<String> list)
{
    Iterator<String> it = list.iterator();
    while (it.hasNext())
        System.out.print(it.next() + " ");
    System.out.println();
}
```

Running time?

# Efficiency check (enhanced for loop)

```
public static void print(List<String> list)
{
    for (String s : list)  // enhanced for loop or "for-each"
        System.out.print(s + " ");
    System.out.println();
}
```

Running time?

# Iterator implementation

- Let's make this concrete and implement one
- I've got a Roster class that holds students
- I can print the roster with *toString()*, but what if I wanted just a list of andrew id's?
- Let's let the Roster implement *Iterable* which will give us an iterator
- We can use the iterator (or a for-each loop) to give us each student in turn; then I can print whatever I want as each comes up