CSE214 Data Structures
Iterator

YoungMin Kwon

Iterator

- Iterator
  - A software design pattern that abstracts the process of scanning through a sequence of elements
- java.until.Iterator interface
  - hasNext(): returns true iff there is at least one additional element
  - next(): returns the next element
- java.lang.Iterable interface for a collection
  - iterator(): returns an iterator of the elements

Copying Data Structures

- Object class has clone() method

```
protected Object clone() throws
                          CloneNotSupportedException
```

  - If a class did not implement Cloneable interface, the clone() method will throw the exception

- By convention,
  - Implement Cloneable interface
  - Override clone() with public access modifier

Equivalence Testing with Linked Lists

- Two lists are equivalent
  - If they have the same size
  - The contents are element-by-element equivalent
- Implementing equals for SinglyLinkedList
  - While simultaneously traversing two lists, test x.equals(y) for each pair of corresponding elements x and y

```
public class IterableSinglyLinkedList<E>
             implements Cloneable, Iterable<E> {
...
    private class ListIterator implements Iterator<E> {
        private Node<E> cur;
        public ListIterator() {
            cur = head;
        }

        public boolean hasNext() {
            //TODO: implement hasNext
        }

        public E next() {
            //TODO: implement next
        }
    }

    public Iterator<E> iterator() {
        //TODO: implement iterator
    }

...
```

```
public IterableSinglyLinkedList<E> clone()
                                    throws CloneNotSupportedException {
    //TODO: implement clone

    //1. use Object.clone() to create the initial copy

    //2. deep copy using iterator
}

public boolean equals(Object o) {
    if(o == null)                    //nothing equalst to null
        return false;
    if(getClass() != o.getClass())  //classes should be the same
        return false;
    IterableSinglyLinkedList<E> that = (IterableSinglyLinkedList<E>) o;
    if(size() != that.size())       //size should be the same
        return false;

    //TODO: finish implement equals using iterator
    //element-wise equivalence
    Iterator i = that.iterator();
}
```

```
public static void main(String[] args) {
    IterableSinglyLinkedList<Integer> list =
                                new IterableSinglyLinkedList<Integer>();
    list.addLast(2);
    list.addLast(3);
    list.addLast(4);
    list.addFirst(1);
    //test iterator
    int i = 1;
    for(int j : list)
        onFalseThrow(j == i++);
    try {
        //test clone
        IterableSinglyLinkedList<Integer> l2 = list.clone();
        //test equals
        onFalseThrow(list.equals(l2));
    } catch(CloneNotSupportedException e) {
        e.printStackTrace();
    }
    onFalseThrow(list.removeLast() == 4);
    onFalseThrow(list.removeLast() == 3);
    onFalseThrow(list.removeFirst() == 1);
    onFalseThrow(list.removeLast() == 2);
    System.out.println("Success!");
}
```