CS367 Announcements
Wednesday, July 3, 2013

- P1 due Wed 11:59pm
- H2 grades today
- H3 posted yesterday, due Mon July 8th
- P2 posted today, due Wed July 17th
- no class July 4th (tomorrow)
- Midterm Thursday July 11th in class (through today)

Last Time

- Access Control
- Linked Lists (cont.)

Today

- Comparable interface
- Linked Lists (cont.)
- LinkedList variations

Comparable Interface

```
Interface Comparable<T>

    int compareTo(T o)
    // Compares this object with the specified object for order. Returns a
    // negative integer, zero, or a positive integer as this object is
    // less than, equal to, or greater than the specified object.
```

Recall Chain of Linked Nodes Data Structure

Conceptual picture

Listnode class constructors and methods

```
Listnode(E d)
Listnode(E d, Listnode<E> n)
E getData()
Listnode<E> getNext()
void setData(E ob)
void setNext(Listnode<E> n)
```

Chain of linked nodes use

Implementing List ADT using Chains of Linked Nodes - the `LinkedList`

```
public class LinkedList<E> implements ListADT<E> {
    private Listnode<E> head;
    private int numItems;

    public LinkedList() {
        head = null;
        numItems = 0;
    }

    public void add(E item) {
        // add at the end of the list
        add(numItems,item);
    }

    public void add(int pos, E item) {
        // detect an invalid requested position
        if  (pos < 0 || pos > numItems)
            throw new IndexOutOfBoundsException();

        // if the list is empty, add a new item with null pointer
        if (head == null) {
            head = new Listnode<E>(item);
        // if adding to beginning of list
        } else if ( pos == 0 ) {
            head = new Listnode<E>(item, head.getNext());
        // otherwise, add at requested position
        } else {
            Listnode<E> n = head;
            //NOTE: increment to pos-1, pointing at node currently at pos
            for (int i = 0; i < pos-1; i++) {
                n = n.getNext();
            }
            n.setNext(new Listnode<E>(item,n.getNext()));
        }

        // always increment numItems
        numItems++;
    }

    ...
}
```

Implementing LinkedList - with tail

```
public class LinkedList<E> implements ListADT<E> {
    private Listnode<E> head;
    private Listnode<E> tail;
    private int numItems;

    public LinkedList() {
        head = tail = null;
        numItems = 0;
    }

    public void add(E item) {
        if (head == null) {
          head = new Listnode<E>(item);
          tail = head;
        } else {
          tail.setNext(new Listnode<E>(item));
          tail = tail.getNext();
        }
        numItems++;
    }

    public void add(int pos, E item) {
        if  (pos < 0 || pos > numItems)
            throw new IndexOutOfBoundsException();

        if (head == null || pos == numItems) {
            add(item);
        } else if ( pos == 0 ) {
            head = new Listnode<E>(item, head.getNext());
            numitems++;
        } else {
            Listnode<E> n = head;
            for (int i = 0; i < pos-1; i++) {
                n = n.getNext();
            }
            n.setNext(new Listnode<E>(item,n.getNext()));
            numitems++;
        }
    }
    ...
}
```

Implementing `LinkedList` - with header node

```
public class LinkedList<E> implements ListADT<E> {
    private Listnode<E> head;
    private Listnode<E> tail; //optional
    private int numItems;

    public LinkedList() {
        head = new Listnode<E>(null,null);
        tail = head;
        numItems = 0;
    }

    public void add(E item) {
        tail.setNext(new Listnode<E>(item));
        tail = tail.getNext();
        numItems++;
    }

    public void add(int pos, E item) {
        if  (pos < 0 || pos > numItems)
            throw new IndexOutOfBoundsException();

        if (pos == numItems) {
            add(item);
        } else {
            Listnode<E> n = head;
            //NOTE: increment to pos, taking header node into account
            for (int i = 0; i < pos; i++) {
                n = n.getNext();
            }
            n.setNext(new Listnode<E>(item,n.getNext()));
            numitems++;
        }
    }

    ...

}
```

Linked List Variations

Singly-linked chains of nodes

with tail reference

with header node

Doubly-linked chains of nodes

Circular singly-linked chains of nodes

Circular doubly-linked chains of nodes