CS 367 Announcements
Tuesday, January 27, 2015

Course Website - http://pages.cs.wisc.edu/~cs367-1/
Sign Up for Piazza

Last Time

Course Topics
Implementing the Bag ADT
- casting when using Object
- using Java generics for generality

List ADT
- coding the ListADT as a Java interface
- using lists via the ListADT

Today

Coursework
Lists
- implementing the ListADT using an array (SimpleArrayList)

Java API Lists
Iterators
- concept
- iterators and the Java API
- using iterators

Next Time

Read: finish *Lists*, start *Exceptions*
Iterators
- options for implementing iterators
- making a class iterable

Exceptions Review
- throwing
- catching/handling

Course Work

TA Consulting office: 1366 CS

Exams (55%)

- Midterm 1 (17%): Tuesday, March 3rd, 5:00 pm to 7:00 pm
- Midterm 2 (17%): Tuesday, April 14th, 5:00 pm to 7:00 pm
- Final (21%): Wednesday, May 13th, 5:05 pm to 7:05 pm

Programming Assignments (25%)

- 5 programs, 5% each
- about two weeks to complete each
- pair programming is allowed
- accepted late only if
  - your extenuating circumstances are beyond your control
  - you notify me at least 3 days prior to due date
  - the work can be completed in a few extra days

Homework Assignments (20%)

- 10 homeworks, 2% each
- about one week to complete each
- no collaboration is allowed
- not accepted late

Email skrentny@cs.wisc.edu by this Friday if you:

- Have a conflict with any of the exam dates.
  Email with subject "CS 367 Exam # Conflict" as in "CS 367 Exams 1 & 2 Conflict"
  and include an appropriate explanation:
  - course/exam – include course name/number, time, instructor name and email
  - VISA – include any accommodation(s) requested
  - other – include concise explanation
- Participate in religious observances that may interfere with course requirements.
  Email with subject put "CS 367 Religious Observance" and include a date and explanation.
- Have a VISA from the McBurney Disability Resource Center.
  Email with subject "CS 367 VISA" and request an appointment.

Recall the List ADT

Memorize it!

Concept

A List is a general container storing a contiguous collection of items, that is position-oriented using zero-based indexing and where duplicates are allowed.

Operations

```
void add(E item);
void add(int pos, E item);
E get(int pos);
E remove(int pos);
boolean contains(E item);
int size();
boolean isEmpty();
```

add end

add to a particular postion

Issues

Null item – detect then signal with `IllegalArgumentException`
Bad position – detect then signal with `IndexOutOfBoundsException`
Empty list – handle as a bad position

Implementation - `ListADT` using a Generic Array

generics Interface

```
public class SimpleArrayList<E> implements ListADT<E> {

    private E[] items;       //the items in the List
    private int numItems;    //the # of items in the List

    public SimpleArrayList() {

        //items = new E [100]; // WRONG!
        items = ( E [ ] ) ( new Object [100] );
        numitems = 0;


   }

  //*** required ListADT methods ***

    public void add(E item) { ... }

    public void add(int pos, E item) { ... }
    public E remove(int pos) { ... }
    public E get (int pos) { ... }

    public boolean contains (E item) { ... }

    public int size() { ... }
    public boolean isEmpty() { ... }

  //*** additional optional array list methods ***

}
```

The ways of implementing them are discussed in reading

Implementing `contains`

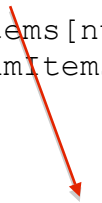
→ **Complete the method below** so that it returns true iff the given `item` is in the list.

```
public boolean contains(E item) {

    // if item is null, JAVA throws NullPointerException
    if ( item == null ) throws new IllegalArgumentException ( ) ;

    // otherwise, traverse
    for ( int i = 0 ;  i < numItems ;  i ++ )
        if ( item.equals( items [ i ] ) )
            return true;
    return false;
```

Implementing `add` at end

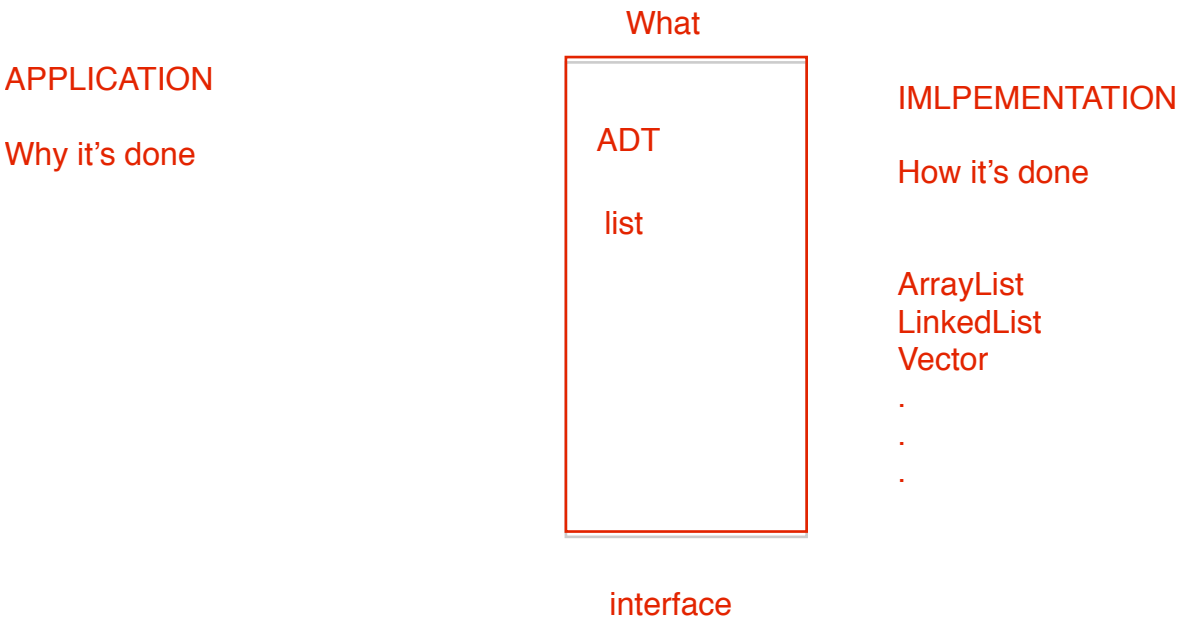
→ What problem might occur with the following implementation:

```
public void add(E item) {
    ……
   items[numItems] = item;
   numItems++;

}
```



```
// we need to add somethings to this code

// if the item is null
if ( item == null )
        throws new IllegalArgumentException (  );

if ( items.length == numItems)
        expandArray( );        // READING!
```

Java API Lists




```
List < String > words = new ArrayList < String > ( );
```

Design - Iterators

What are they?

Consider some lists of words …
e.g. a, b, c, d, …
The idea of “traversing through" make sense for any collection…

ITERATOR
- An Object used to step/iterate through a collection
- An abstraction of a pointer
  - stores its position for a particular collection
  - position can change
  - item at the position can be accessed
- is external / separate from its container
  - has its own interface & implementing class
  - we can instantiate many iterators pointing to the same or different items

Concept

step1. we need to get an iterator, how do we make them?
- go to a particular container and ask for an iterator

step2. use iterator
- to access items in the container

Operations

1. Container class modified to provide an iterator
2. We need an iterator class that…
   - advance to the next position
   - access the item at position x
   - determine if it is at the end of collection

Interfaces - Iterators in Java API

`Iterable<T>` **interface in** **`java.lang`**

specifies the operation to get an iterator for stepping through a collection:

- Iterator<T> iterator()

`Iterator<E>` **interface in** **`java.util`**

specifies the operations that iterators can do:

- boolean hasNext()
- E next()
- void remove() //"optional"

Use - Iterators

Suppose `words` is a `SimpleArrayList<String>` that implements the `Iterable` Interface.
→ **Write a code fragment** that gets an iterator, named `itr`, from `words`.

Suppose `words` is a `SimpleArrayList<String>` and `itr` is an iterator for `words`.
→ **Write a code fragment** that uses `itr` to print each item in `words`.

Use - Iterators

Assume `SimpleArrayList<String>` implements the `Iterable` Interface.
→ **Complete the method** using iterators to determine `list` contains duplicates.

```
public boolean hasDups(SimpleArrayList<String> list) {
```