We assume that you are proficient at object-oriented programming in Java.

Instructors
- Lec 2 & 1: Deb Deppeler, 5376 CS, deppeler@cs.wisc.edu
- Lec 3: Alexi Brooks, 4369 CS, albrooks@cs.wisc.edu

Course Website: https://canvas.wisc.edu/
If Canvas is down, check this website: http://pages.cs.wisc.edu/~cs367-1/
See Modules for online readings and lecture outlines (there is no textbook)
zyBooks? Online interactive text recommended for students seeking more practice
Homework h0 due 10pm Friday, January 27th
Homework h1
Program p0
Program p1
Assignment questions?
  Post it on Piazza. Invitations to join were sent yesterday to your UW email.
  Consult with a TA in 1366 CS. See course page for link to scheduled hours.

Last Week
  Collections, interface BagADT, autoboxing, ArrayBag implements BagADT, casting when
  using Object, using Java generics for generality, List ADT, designing the ListADT, coding the
  ListADT as a Java interface
  Read: Introduction to ADTs and Lists

This Week
  Read: finish reading Lists
  Lists
    - using lists via the ListADT
    - implementing the ListADT using an array (SimpleArrayList)
  Java API: interface List: superset of our ListADT with a few differences
  Iterators
    - concept
    - iterators and the Java API
    - using iterators
    - options for implementing iterators
    - making a class iterable

Next Week
  Read: Exceptions
  Handin Info
  Exceptions Review: throwing, handling, execution, practice with exception handling, throws
  and checked vs. unchecked, defining
Recall the List ADT

Concept
A List is a general, position-oriented container that stores a contiguous collection of items where duplicates are allowed. It maintains relative ordering and uses zero-based indexing.

Operations

```java
void add(E item); // add at end of list
void add(int pos, E item); // insert at pos and shift rest of list
E get(int pos); // get without remove item at pos
E remove(int pos); // remove and return item at pos
boolean contains(E item); // true if item is in the list
int size(); // return the number of items in list
boolean isEmpty(); // return true if list has 0 items
```

Issues

- add and contains:
  - null item – detect then signal with IllegalArgumentException
- add, get, and remove:
  - invalid (bad) position – detect then signal with IndexOutOfBoundsException
  - empty list – handle as a bad position

Compare ListADT with Java's List

Read javadocs

Similarities                      Differences:
Use – ListADT example: remove

→ Assume myList is a ListADT.
What does the following code fragment do in general?

```java
//
for ( int i = 0; i < myList.size() ; i++ )
{
  myList.remove( i );
}
```

Tip: trace code and generalize about the result of this code on a list.

Use – ListADT example: FLIP

→ Assume myList is a ListADT. Write a code fragment to reverse the contents of myList without using any additional ListADTs or other data structures (e.g., array).
Implementation - ListADT using a Generic Array

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() {

    }

    //*** 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 ***
    }
}
Implementing contains

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

    public boolean contains( E item ) {
Implementing add at end

→ What problem might occur with the following implementation:

```java
public void add( E item ) {
    items[numItems] = item;
    numItems++;
}
```
Design - Iterators

What are they?

CODING:

Concept

Container's operation(s)

Iterator's operation(s)
Iterators in Java API

- Separate the idea of tracking current position in a container from the implementation of the container.

HOW DO YOU GET AN ITERATOR?

`Iterable<T> is an interface in java.lang`  
Specifies the operation to get an iterator:

- `Iterator<T> iterator()`

WHAT CAN YOU DO WITH AN ITERATOR ONCE YOU HAVE IT?

`Iterator<E> interface in java.util`  
Specifies the operations that an iterator can do once you have the iterator:

- `boolean hasNext()`
- `E next()`
- `void remove() //"optional"`
Use – Iterators examples: get iterator, print all

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.

⇒ Next write a code fragment that uses itr to print the length of each item in words.
Use – Iterators example: use to determine hasDups?

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

    public boolean hasDups(SimpleArrayList<String> list) {

Implementation – Indirect access vs Direct access

Indirect Access

Direct Access
import java.util.*;
public class SimpleArrayListIterator<E> implements Iterator<E> {

    public SimpleArrayListIterator(                     ) {

    }

    public boolean hasNext() {

    }

    public E next() {

    }

    public void remove() {

    }
}
Implementation - Direct Access ArrayBagIterator Class

```java
import java.util.*;
public class ArrayBagIterator<E> implements Iterator<E> {

    public ArrayBagIterator() {

    }

    public boolean hasNext() {

    }

    public E next() {

    }

    public void remove() {
        throw new UnsupportedOperationException();
    }

    // Could we code this as an indirect access iterator instead?
```
Making Array Bags Iterable

Approach in Readings - Modify the BagADT Interface

```java
import java.util.*;
public interface BagADT<E> {
    void add(E item);
    E remove() throws NoSuchElementException;
    boolean isEmpty();
}
```

Also Modify the ArrayBag Class

```java
import java.util.*;
public class ArrayBag<E> implements BagADT<E> {

    // *** Data members (fields) ***
    private E[] items;
    private int numItems;
    private static final int INIT_SIZE = 100;

    //*** required BagADT methods ***
    public void add(E item) { ... }
    public E remove() throws NoSuchElementException { ... }
    public boolean isEmpty() { ... }
}
```

Java's approach is to use indirect iterators.
Can we make an indirect iterator for the ArrayBag class?