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

Course Website: [https://canvas.wisc.edu/](https://canvas.wisc.edu/) (Log in using NetID)
If Canvas is down, we may post updates on: [http://pages.cs.wisc.edu/~cs367-1/](http://pages.cs.wisc.edu/~cs367-1/)

See modules links for online readings and lecture outlines (no textbook)

Waitlisted or Canvas problems? Sign name, NetID, and UW ID# on sheet at front of class.

WACM Basic Linux Workshop Wed 5:30-7:00pm, Thurs 5:30-7:00pm

Homework h0 and h1, due 10 pm Friday, Sept 16th
Program p1 assigned. (See Assignments or Syllabus page for links)

Assignment questions? Post it on Piazza or Consult with a TA during scheduled hours.

Last Time
Lists
- using lists via the ListADT
- implementing the ListADT using an array (SimpleArrayList)
Java API Lists
Iterators Demo

Today
Iterators
- concept – finish last page of Tuesday outline
- using iterators
- options for implementing iterators
- making a class iterable

Next Time
Read: Exceptions
Program Handin Info
Exceptions Review
- throwing
- handling
- execution
- practice with exception handling
- throws and checked vs. unchecked
- defining
Interfaces - Iterators in Java API

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

specifies the operation to get an iterator to step 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`.

→ Next write a code fragment that uses `itr` to print the length of each item in `words`.
Use - Iterators

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 - Options for Iterator Classes

Indirect Access

Direct Access
Implementation - Indirect Access SimpleArrayListIterator Class

```java
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

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() { ... }

}
```

How would we do this using Java’s approach?