Welcome to CSE 143!

Hello world!

My name is Adam.
1. Administrivia
2. Internal Correctness (aka Style)
3. Arrays and ArrayLists
CSE 142 vs. CSE 143: The Big Picture

In **CSE 142**, you learned how to use logic, control flow, and decomposition to write programs.

In **CSE 143**, you will learn to solve more complex and larger tasks efficiently.

Big Learning Goals

- Abstraction (implementation vs. client)
- Data Structures (organizing complex data)
- Algorithms (standard ways of completing common tasks)

We’re going to build some **really cool** programs. And have a lot of fun!
Support and Asking for Help

Resources

- **TWO** sections a week
- Tons of TAs!
- The IPL (and my office hours!)
- Practice-It

Asking for help is not a sign of weakness; it’s a sign of strength.
Course Website

http://cs.uw.edu/143

Section

We have two sections a week. Each section has a warm-up; these are completely optional.

Grading

- 50% programming projects, 20% midterm, 30% final
- Weekly programming projects assigned Fridays, due on Thursdays
- 5 “free late days”; -2 points for subsequent days late; up to 3 days late on each hw
How Does the Course Fit Together?

**Descriptive Knowledge** ("What is it?")

Knowing Facts.

**Procedural Knowledge** ("How do I do it?")

Doing Procedures.

**Conceptual Knowledge** ("Where does it come from?")

Understanding HOW things work and WHY they work that way.

Lecture provides descriptive knowledge which feeds into

Section Warm-Ups, which review and feed into

QuickChecks, which activate knowledge and prepare you for

The rest of section which turns desc. into proc. knowledge and then

Homework helps you turn desc. and proc. into conceptual

Exams activate all three types of knowledge
What does it mean for a program to be "correct"?

A program is only correct if it is **internally** correct and **externally** correct.

What does this code do?

```c
(_(__,___,____){___/__<=1?(_(__,___+1,___ _):!(___%__)?(_(__,___+1,0):___
  %__==___ / __&!____?(printf("%d\t",___/___),(___, __+1,0)):___%_
  >1&___%<___/___?(_ __,1+ ___,____+!(__/__%(___%__))):___<___*
  ?(_(__,___+1,____):0;}main(){__(100,0,0);}}
```
Correctness

What is External Correctness?
The code does the right thing on all inputs.

What is Internal Correctness?
The code is...
- easy to read
- well documented
- well formatted
- efficient
- ...

Internal correctness matters, because:

- Do you want a job at a software engineering company?
- Do you want to ever reuse your code later?
- Do you want to ever write a large program? (Like a game, maybe)
- Important people think it does:
  - Programs must be written for people to read, and only incidentally for machines to execute. (Abelson & Sussman)

Goals For Internal Correctness

- Make non-obvious code obvious via comments.
- Document all features, limitations, design decisions.
- Make your code easy for someone else to read
- Explain what your methods, classes, etc. are supposed to do

Grading will be on both external and internal correctness!
Write code to read a file and display its words in reverse order.

(Bad) Solution with Arrays

```java
String[] words = new String[1000];
int i = 0;

Scanner inp = new Scanner(new File("words.txt"));
while (inp.hasNext()) {
    String word = inp.next();
    words[i] = word;
    i++;
}

for (int j = i - 1; j >= 0; j--) {
    System.out.println(words[j]);
}
```
Arrays are one way to store many values of the same type (int, String, DrawingPanel, etc.).

```java
int[] arr = new int[8];
```

```
arr:  12  49  -2  26  2  6  26  11
```

“Element #3 is 26”  “arr has size 8”

Limitations of Arrays

- Fixed, upfront size (once you create the array, it will remain that size)
- Adding and removing can get complicated
- No methods (and weird “.length” syntax)
  Functionality for arrays is in the Arrays class:

  - Arrays.copyOf
  - Arrays.equals
  - Arrays.sort
  - Arrays.toString
Collections and Lists

Collections

Collections store many pieces of data of the same type.

In Java, collections are in the util package:

```java
import java.util.*;
```

Different collections have different properties:

- “Data ordered by indices”
- “Sorted data”
- “Data without duplicates”
- etc.

Lists

A list is a collection of elements ordered by a 0-based index.

- It supports add/remove from anywhere!
- The size isn’t fixed!
- There are multiple implementations; first, ArrayList
Suppose we have an ArrayList with values: [1, 2, -6]:

Step 0: 
```
  0 1 2 3 4 5 6 7 8 9 10
 1 2 -6 ...
```

Insert 5 at index 2:

Step 1: 
```
 0 1 2 3 4 5 6 7 8 9 10
 1 2 5 -6 ...
```

Add 0 at the beginning:

Step 2: 
```
 0 1 2 3 4 5 6 7 8 9 10
 0 1 2 5 -6 ...
```

Get index 3:
```
arrayList.get(3) \rightarrow 5
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(val)</td>
<td>Appends val to the end of the list</td>
</tr>
<tr>
<td>add(idx, val)</td>
<td>Puts val at index idx; all elements at indices idx and larger get shifted forward</td>
</tr>
<tr>
<td>get(idx)</td>
<td>Returns the value at index idx</td>
</tr>
<tr>
<td>set(idx, val)</td>
<td>Replaces the value at index idx with val</td>
</tr>
<tr>
<td>remove(idx)</td>
<td>Removes and returns the value at index idx; all elements at higher indices get shifted backward</td>
</tr>
<tr>
<td>clear()</td>
<td>Removes all elements from the list</td>
</tr>
<tr>
<td>size()</td>
<td>Returns the number of elements in the list</td>
</tr>
<tr>
<td>indexOf(val)</td>
<td>Returns the smallest index such that get(idx).equals(val), or -1 if there is no such index</td>
</tr>
<tr>
<td>toString()</td>
<td>Returns a string representation of the list such as [3, 42, -7, 15]</td>
</tr>
</tbody>
</table>
Recall that we can create arrays of different types:

\[
\begin{align*}
\{1, 2, 5, 2\} & \quad \{"hi", "banana"\} \\
(new \ int[4]) & \quad (new \ String[2])
\end{align*}
\]

Since the array initializations specify the \textbf{type} of the elements, the declaration for ArrayList's should too:

\[
\begin{align*}
[1, 2, 5, 2] & \quad ["hi", "banana"] \\
(new \ ArrayList<Integer>) & \quad (new \ ArrayList<String>)
\end{align*}
\]

ArrayList is a \textbf{generic} class which means that it can handle any type you want! Java knows the type by what you put in <>:

\[
\text{ArrayList<String> \ arrayList = new ArrayList<String>());}
\]
Note that these two pieces of code have **different** loop bounds:

```java
String[] arr = new String[5];
arr[0] = "hi";
arr[1] = "bye";
String s = arr[0];
for (int i=0; i < arr.length; i++) {
    if (names[i].contains("b")) {...}
}

ArrayList<String> list = new ArrayList<String>();
list.add("hi");
list.add("bye");
String s = list.get(0);
for (int i = 0; i < list.size(); i++) {
    if (list.get(i).contains("b")) {...}
}
```

arr.length == 5  list.size() == 2
ArrayList can be a Parameter or a Return Value

ArrayList is just another type (like DrawingPanel or String)!

```java
definition

public void methodName(..., ArrayList<Type> name, ...) { ... }
```

```java
public ArrayList<Type> methodName(...) { ... }
```

The following takes in an ArrayList and returns a new list containing only the words that start with x:

```java
definition

public ArrayList<String> startingWithX(ArrayList<String> list) {
    ArrayList<String> newList = new ArrayList<String>();
    for (int i=0; i < list.length; i++) {
        if (list.get(i).startsWith("x")) {
            newList.add(list.get(i));
        }
    }
    return newList;
}
```

Words Exercise... Now with more ArrayList!

Write code to read a file and display its words...

1. in reverse order (but using an ArrayList)
2. with all words ending in “s” capitalized
3. with all words ending in “s” removed

```java
/* Read in the words */
ArrayList<String> allWords = new ArrayList<String>();
Scanner input = new Scanner(new File("words.txt"));
while (input.hasNext()) {
    String word = input.next();
    allWords.add(word);
}

/* Display in Reverse Order */
for (int i = allWords.size() - 1; i >= 0; i--) {
    System.out.println(allWords.get(i));
}
```
/* Remove All Words Ending in ‘s’ */
for (int i = 0; i < allWords.size(); i++) {
    String word = allWords.get(i);
    if (word.endsWith("s")) {
        allWords.remove(i);
        i--;
    }
}
Today’s Takeaways!

- Understand the course policies

- Learn why internal correctness is important (Are you convinced?)

- Recall arrays and how they work from CSE 142

- Begin being a client of the ArrayList class