Welcome to CSE 143!

Outline

1. Administrivia
2. Code Reviews
3. Back to CSE 142
4. ArrayLists

Course Goals

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!

Boring Administrivia

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”; 10% off for subsequent days late; up to 2 days late on each HW

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.
Program Correctness

What does it mean for a program to be “correct”?

What does this code do?

```
(..., ___, ___) {..., ___, +1, ___}!!(..., ___+1, 1.8); ___

% ___ / ___ / ___ % % ___
>166 % % % / ___ / ___ / ___ / ___ 167 (... +1, 1) ___ +1 ___ ___

7.(___, ___, +1, 1); main(|/(100, 0, 0|));
```

Programs must be written for people to read, and only incidentally for machines to execute. (Abelson & Sussman)

Correctness

What “Program Correctness”?

- Your code does the right thing on all inputs
- The code is easy to read
- The code is well documented
- The code is well formatted
- The code is efficient

How Can We Determine If A Program is Correct?

- Write tests
- Think about “edge cases”
- Ask someone experienced to review your code!

This last one is actually really important!

Industry calls it a “code review”.

Code Reviews @ Google

Google has a “legalese” document describing “Google Style”:

https://google.github.io/styleguide/javaguide.html

If a Google programmer submits code that misses even one of these guidelines, it is

REJECTED!

Each programming language (C++, Java, Python, etc.) has different guidelines.

The actual guidelines themselves aren’t important; Facebook, for example, has different ones.

The relevant skill here is being able to follow style guidelines.

Code Reviews in CSE 143

We will grade your programming assignments by code review.

Graders will check that your code follows “CSE 143 Style”:

http://courses.cs.washington.edu/courses/cse143/16au/style

Beware! The style guide may not include everything!

Google’s doesn’t either.

Review: Arrays

Arrays are one way to store many values of the same type (int, String, DrawingPanel, etc.).

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

“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

ArrayList Mechanics

- Suppose we have an ArrayList with values: [1, 2, -6]:
  - Step 0: 1 2 -6
  - Insert 5 at index 2:
    - Step 1: 1 2 5 -6
  - Add 0 at the beginning:
    - Step 2: 0 1 2 5 -6
  - Get index 3:
    ```java
    ArrayList<String> list = new ArrayList<String>();
    list.add("hi"); list.add("banana");
    System.out.println(list.get(3)); // prints 5
    ```

ArrayList Reference

<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 indices higher 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>

ArrayList Demo

```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 (arr[i].startsWith("b")) { // Step 0: 1 2 -6
        int idx = indexOf("b"); // to find the index of "b"
        if (idx != -1) { // Steps 1 and 2: 1 2 5 -6
            arr[i] = "hi"; // Step 3: 1 2 5 -6
        }
    }
}
```

Generics

Recall that we can create arrays of different types:
```java
(new int[4]) (new String[2])
```

Since the array initializations specify the type of the elements, the declaration for ArrayList’s should too:
```java
(new ArrayList<String>) (new ArrayList<Integer>)
```

ArrayList is a generic class which means that it can handle any type you want! Java knows the type by what you put in <>:
```java
ArrayList<String> arrayList = new ArrayList<String>();
```

ArrayList can be a Parameter or a Return Value

ArrayList is just another type (like DrawingPanel or String)
```java
public void method(String x) { ... } // x is a String
public ArrayList<String> method() { ... } // returns a new ArrayList<String>
```

The following takes in an ArrayList and returns a new list containing only the words that start with x:
```java
public ArrayList<String> startingWithX(ArrayList<String> list) {
    ArrayList<String> newList = new ArrayList<String>();
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i).startsWith("x")) {
            newList.add(list.get(i));
        }
    }
    return newList;
}
```
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 code reviews are important (Are you convinced?)
- Recall arrays and how they work from CSE 142
- Begin being a client of the ArrayList class