CSC207.01 2013F, Class 08: Some Basic Types: Numbers and Strings in Java

Overview

- Admin.
- Design of ADTs: Minimalist vs. Maximalist
- Arrays, Continued.
- Modeling Strings.
- And Numbers.

Admin

- Warning! Friday the 13th falls on a Friday this month.
- Reminder: Mentor session TONIGHT at 8:30 pm
- Although there are links (broken?) to labs for yesterday and today, we won't be holding those labs. (Parts of the labs are in the homework assignment.) I do hope to hold lab on Friday.
- Nationwide will be on campus next Tuesday to talk about (and recruit for) jobs and internships. There's a 4:15 talk, but there's also a noon lunch for Math/Stats and CS majors. Signs should be around.
- Reading for Friday (expect it Thursday; I'll send email)
  - Input and Output
- It's been noted that the kinds of questions I'm asking on your homework assignment are a lot like coding questions that get asked in interviews. The overlap is *not* intentional. (I don't object to it, but I just choose questions that I think will challenge you appropriately.)
- If you are not on the CS mailing list and want to be on it, let me know and I'll get you added.
- I see only one (worried) prologue for HW3.
  - Here's a hint on the last problem: Recursion is likely to be your friend, at least at first.
  - Experiments should suffice FOR THIS PROBLEM ONLY.
- EC opportunities:
  - Humanities Center Speaker Sarah Hendron, TONIGHT, 7:30 p.m., JRC101 Waking the Machines: Art, Design, and Adaptive Technology
  - Learning from Alumni 2:15-4:05 Thursday: Ian Lunderskov '08
  - We think the Matt Atherton Thursday extra is NOT happening. Sorry.
  - CS Table, Friday: Trusting Trust.
  - More?
- No EC, but a chance to talk about Feminist Action Coalition at 8pm tonight in Younker

Minimalist vs. Maximalist ADT Design

Why Minimalist?

- Laziness.
- Less for your audience to read.
- All your clients should need are the building blocks; you can’t predict every possibility, so why try?
  - And your clients will benefit from having to implement them, particularly if they are novices
  - Helps the client learning curve
- Anti-max: The more you have to implement, the more restricted the implementation
  - And you’ll make tradeoffs
  - Example: Java has an ArrayList type: It’s indexed like arrays, but it allows you to insert in the middle, like lists
- Anti-max: The more you have to implement, the less likely you are to try something new

Why Maximalist?

- Experienced programmers will make fun of you if you don’t include
  the procedures they expect
- As a client programmer, you can know that you will have most, if not all of the utility procedures you will need.
  - Client programmers are more efficient
- Saves total programmer time

Sam likes minimalist approaches, at least for novice implementers

Arrays, Continued

- Start with a few basic procedures:
  - Create/initialize/declare
  - Get ith element
  - Set ith element
  - Get length/size
- Maximalists might add
  - um - fill the array with u’s and m’s
  - convert to and from lists
  - copy values from an array to another array (yeah, a for loop would do the job, but ...)
    - A clever implementation might do this as a bit copy or in parallel
  - apply proc to each element, mutating each element
    - Or returning a new array
  - shuffle - rearrange randomly
  - member; indexOf
  - sort
  - print

    - Grab a subsection of an array
- A hidden issue
    - collections of values
    - homogeneous (simple; necessary for the "jump to start + width*index)
    - indexed by integers (typically, starting with 0)
    - We’ve implicitly added "Fixed size"

Vectors

Java provides Vectors, which are dynamic arrays.

Declare

```
Vector<TYPE> name;
```

Initialize

```
 new Vector<TYPE>(SIZE)
```

Set

```
 name.set(*index*, *val*)
```

Get

Length? (# elements vs capacity)

- Read the documentation for more information.

Strings

What are strings? (Purpose/Philosophy)

- Human-readable ordered collections of characters (maybe ASCII, maybe EBCDIC, maybe Unicode)
- Practica: Interact with human beings

- Procedures (minimalist)

    - read them (input)
    - print them (output)
    - create new ones (with input of ...?)
    - compare alphabtically (given your local version of alphabetical)
    - length
    - check if empty (or does length == 0 suffice?)
    - concatenate
    - get individual elements
    - substring

    - mutate a letter

- In Java, strings are IMMUTABLE. (So they can share memory.)

  ```
  eyes.println("Hi, " + studentOne);
  eyes.println("Hi, " + studentTwo);
  ```

- When you want string-like things that are mutable, use StringBuffers

  ```
  String val = "Hello";
  String newval = val + val;
  eyes.println(val.charAt(i));
  eyes.println("There are " + val.length() + " characters in val");
  if (val.compareTo(other) < 0) {
    eyes.println (val + " comes before " + other);
  }
  ```

Numbers

Not covered