Computer Science 134C
Introduction to Computer Science, in Python
Lecture #30 (Java III)
November 28

Keywords
arrays, vectors

Building an extensible container class in Java.

1. Questions?

2. Java supports many container classes. Today, we'll take a look at `String`s, arrays, and `Vector`s.

3. The `String` class in Java is very similar to the `str` class in Python:

   (a) They're immutable. To make changes, you *must* construct the result from scratch.

   (b) The length of a `String` is determined by the `.length()` method.

   (c) Concatenation is accomplished with the + operator, as in Python.

   (d) They're indexed, starting at 0. You can retrieve a `char` with the `.charAt(i)` method.

   (e) You can extract a portion of the string with the `.substring(a,b)`. The sub-string is found starting at index `a` and ending just before `b` (just like Python). You can't use negative indicies, and there is no method to "stride" through the string typical of Python slices.

   (f) You can locate a substring with `.indexOf(sub)`; it returns -1 if not found.

   (g) You can `split`, `join`, and `trim` (ie. `strip`), similar to Python. It includes `.startSwith(s)` and `.endsWith(s)` methods.

   (h) There are method `.toLower()`, `.toUpper()`, etc.

   (i) There is a `.format` mechanism, but the format string is very different.

4. Unlike Python, Java has an "array" type.

   (a) Arrays are fixed length, indexable containers.

   (b) They are declared in one of two ways. You can specify the length explicityly:

   ```
   int[] data = new int[10];          // ten zeroes.
   ```

   or you can provide initial values with

   ```
   int[] data = { 2, 5, 7, 9, 11 };  // array contains 5 primes
   ```

   (c) Elements are indexed (e.g. `data[0]`, `data[1]`, etc.). The number of values in the array is an instance constant, `data.length`. Note that this is not a method call.

   (d) Two typical ways of iterating across the elements:

   ```
   for (int i = 0; i < data.length; i++) {
       System.out.println(data[i]);
   }
   ```

   or you can use an iterator-based for loop (similar to Python's `for-in` loop):

```
for (int value : data) {
    System.out.println(value);
}
```

The latter approach is more modern, and increasingly preferred.

5. There is a `Vector` class that is most similar to Python's `list` class.
   (a) The `Vector` is an ordered structure, like an array, but it is *extensible*.
   (b) The current "length" of a `Vector` is determined by the `.size()` method call. This can be confusing, but realize that `.size()` is the norm. Only `String` has a `.length()`.
   (c) `Vectors` can be indexed, but not with square-brackets. You must use `.elementAt(i)` or `.get(i)`.
   (d) You can change values with `.setElementAt(v,i)` or `.set(v,i)` (`.set` also returns the previous value).
   (e) You can find values with `.index(v)` (which returns -1, if not found) and check for the existance of a value with `.contains(v)`.
   (f) You can append values with `.add(v)`, or "insert" them at specific locations with `.add(v,i)`.
   (g) Many other operations. Look for documentation in `java.util`'s `Vector` page.

6. Example: A simple (?) little machine that can be simply (?) programmed.
   (a) The machine has an infinite tape (all initially 0), read and written with a "tape head".
   (b) The machine has a current state (a small integer).
   (c) Instructions are 5-tuples that contain a `state` and `symbol`, a *new* state and symbol, and a direction to move the tape head.
   (d) An instruction executes when its `state` and `symbol` match the state of the machine and the symbol under the tape head.
   (e) As the instruction executes, it (1) writes the new symbol on the tape under the tape head, (2) shifts the tape in the machine, and (3) the machine enters the new state.
   (f) Execution halts when no instruction matches the current state of the machine.
   (g) This machine is called a *Turing Machine*, perhaps the most important model of computation.

⋆