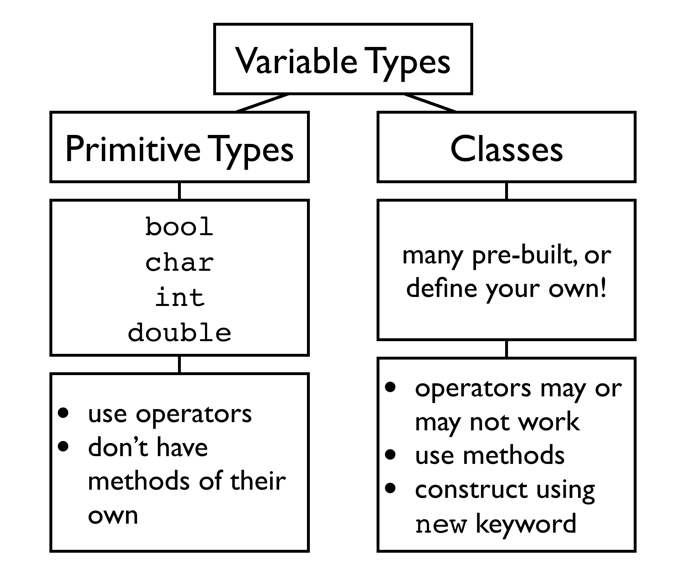
Julia Daniel Section #1

CS 106A April 18, 2018

Section #2: Lesson Plan

Written by Julia Daniel

This week, we’re diving into Java in section for the first time. The main goal of section is getting students to understand control flow (as a sequence of steps the computer takes, including repeats and conditions) and variables (as named and typed ‘boxes’ that can store values).



**Key Concepts**

* Console vs. graphics programs
* Variable types (see diagram at right, and maybe even copy it onto the board)
* Constants vs. local variables
* Variable scope
* Declaring, instantiating, accessing, and modifying variables
* For loops: nesting, using the counter, modifying step size, sentinel loops
* Operator precedence
* Parameter(s) 🡪 method 🡪 return value

**Lecture Recap**

Here’s quick summaries of the relevant lectures for this week’s section. Please check out the slides in full if you haven’t been in lecture.

1. **Intro to Karel**

Course logistics, 4 basic Karel commands, anatomy of a program: import statements + class definition + run method + helper methods (Chris referred to this as “adding vocabulary words” to Karel’s vocabulary)

1. **Control flow in Karel**

**for**, **while**, **if**-**else**, and running programs in Karel (direction testing, etc.)

1. **Decomposition**

Break down a problem into milestones, then use top-down decomposition to get to a particular milestone, test and fix, then move on to next milestone

1. Variables

Variables as named boxes that you can put values into and read values from

1. Control flow in Java

Booleans, operator precedence, using for loop variables

**Simple Java Problem(s)**

Students will be wrapping up their time with Karel after this week (except for the midterm!), but we’ve given two problems that should help solidify some of the important concepts they need for approaching future assignments, specifically **milestoning**, **problem decomposition**, and **testing**. One major point to make is that ***finding bugs is a sign of achievement, not failure*** – no one writes perfect code on the first try all the time, but *finding* your bugs quickly, and writing code that leads to *simpler, easy-to-fix*bugs, are skills to learn. Feel free to share your own past frustrations with buggy code with students, especially later on in the quarter with students who feel discouraged or “bad at coding”.

**United Nations Karel**

There are two main milestones to tackle, and the order doesn’t really matter:

* Building a single house
  + Assuming we know we are at a location where we want to build a house (abstracting away how we got there), how do we do it?
  + Talk about decomposing repeated tasks – in this case, making a column of three beepers – and adding “vocabulary items” (methods) to Karel.
* Crossing the entire world and putting a house wherever you find rubble
  + Talk about abstracting away ***how*** the house is built to just focus on telling Karel ***when*** to build a house; assume you have a working **buildHouse()** method.
  + Test that this can be done correctly, without worrying about whether the house itself is built correctly. This can just be a couple sentences – *“let’s run through this. If we have rubble here and here, will this [pseudo]code put houses where we want?”* – to build in habits in your sectionees of testing whenever they think they’ve finished coding a milestone.

**Karel Defends Democracy**

Here, too, think about how to incorporate lessons of milestoning, decomposition, and testing. If you don’t make it through this problem, no worries. Make sure to tell students that we give them more problems than they’ll get to in section on purpose – these are good practice problems for them to tackle on their own!