The Heap

- The Stack
  - Overview:
    - The amount of memory we have for the stack is SMALL
    - Accessing memory is very FAST
    - Each function takes up a chunk of it (a "frame")
    - Each function can only see variables stored in its own stack frame!
  - How does allocation work?
    - When the function is ***called***, that chunk is "allocated" (i.e. set aside for it)
    - This is usually fine, because we can see how many variables exist/how big they are
      - Example
    - HOWEVER, at the time the function is called, we might ***not*** always know how big all the arrays are!!!!
      - How could this happen?
    - SO, we ***cannot*** store arrays in the stack.
    - What do we do?!?!?!?!!??!?!
- The Heap
  - Overview
    - TONS and TONS of space
    - Accessing memory is slower than the stack
    - Each thing in the heap has an "address"
    - Each thing is "allocated" some amount of memory that belongs to it (no one else can touch it!)
  - Arrays in the heap
    - THIS is where we put arrays, since they're arbitrary lengths...
    - Buuuuut, the ***function*** that is using it needs to know where in the heap it was stored!
    - So we need to store the address of the array in the function's stack frame
      - THIS is what's stored in the "array" variable in the stack...
      - Also called a "reference" or "pointer"
    - "new" keyword: allocates memory in the heap
      - every time we make a new array, need this
      - every time we say "new", we know exactly how much memory to allocate
      - How we've been doing it is a LIE: int[] a = {10}; // shorthand for int[] a = new int[]{10};
      - a = {10}; // not valid code!!
  - "Dereferencing" arrays:
    - When we do array[0] or array.length, what happens?
    - Follow the pointer, and look at the object that's there!
- Two categories of types: Primitives and reference types
  - Everything else we've seen is a primitive
    - int, double, boolean, char, long, float
    - String is a reference type, but is special...
  - Primitives are stored in the stack, reference types in the heap