**Symbol Table Contents**

Symbol table: a function from identifiers in the current scope to the symbol that defines relevant information about this identifier

All information is derived from syntax tree

* symbol table is interconnected with syntax tree
* in old one-pass compilers there was only symbol table, no syntax tree
* in modern compiler: we could always go through entire tree, but symbol table can give faster and easier access to the part of syntax tree, or some additional information

Goal: efficiently supporting phases of compiler

In the initial name analysis phase:

* finding which identifier refers to which definition
* we store *definitions*

What kinds of things can we define? What do we need to know for each definition?

* variables (globals, fields, parameters, locals)
  + need to know types
  + later: memory layout
    - e.g. 3rd field in an object should be stored at offset e.g. +6 from the address of the object
  + sometimes need to make more information explicit:
    - whether variable is declared local or global
* methods, functions, classes - recursively with their own tables