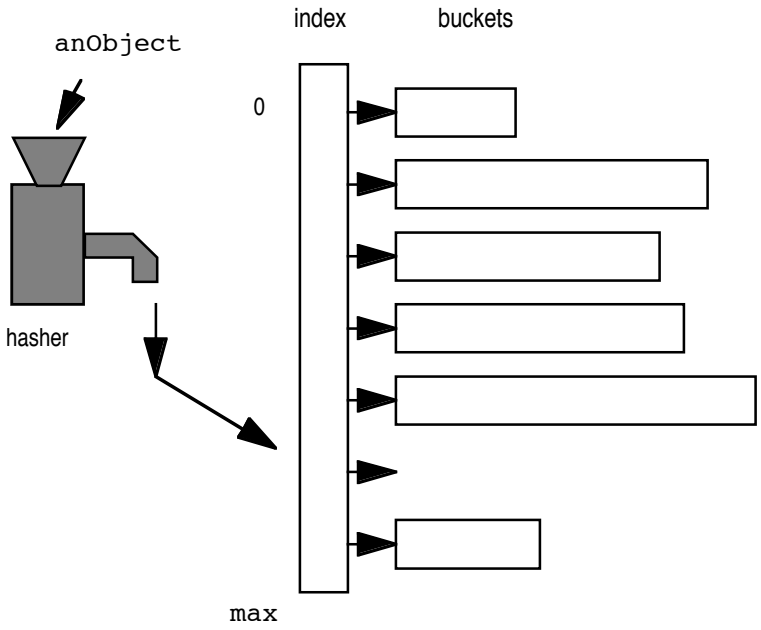
GCL SymbolTable

A Chain of Hash Tables

based on java.util.Hashtable

Hash Tables

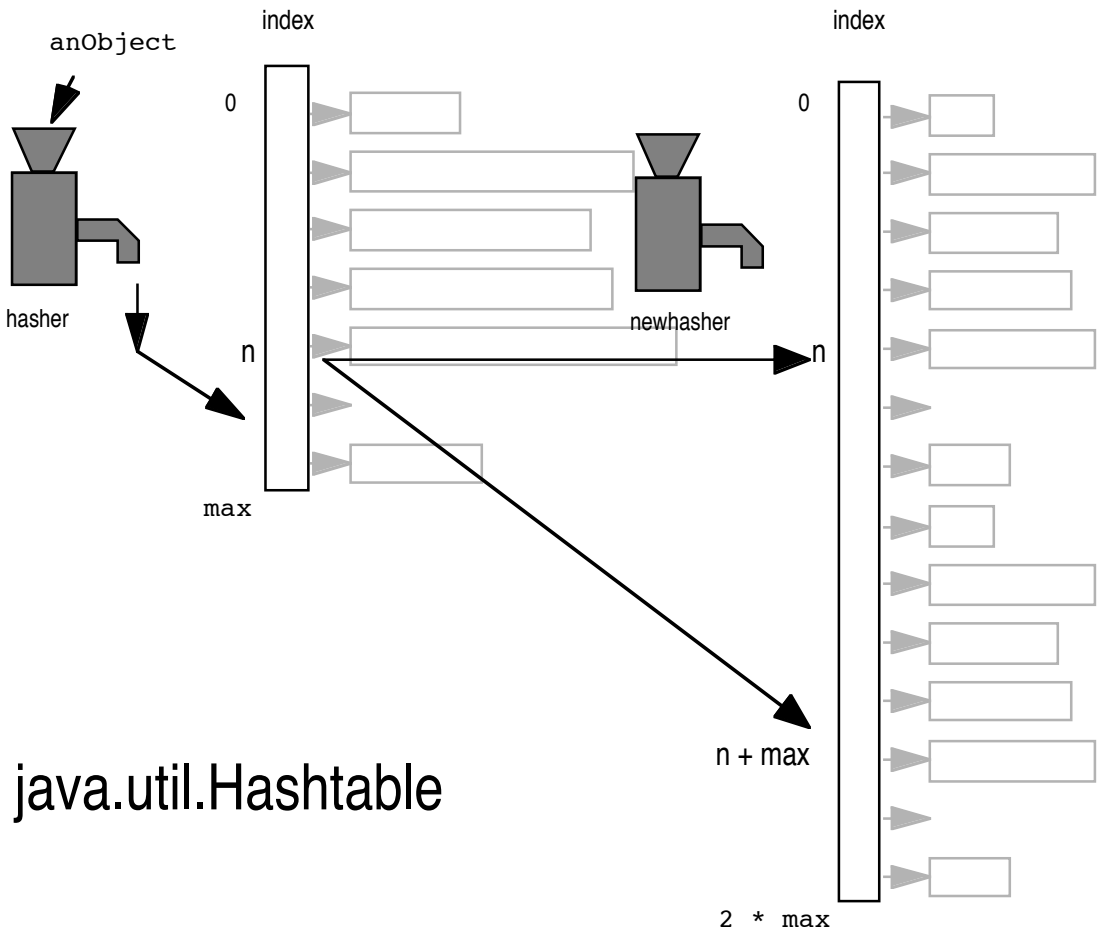
Keys are used to determine the location in which to store a Value.



Self Organizing Hash Table

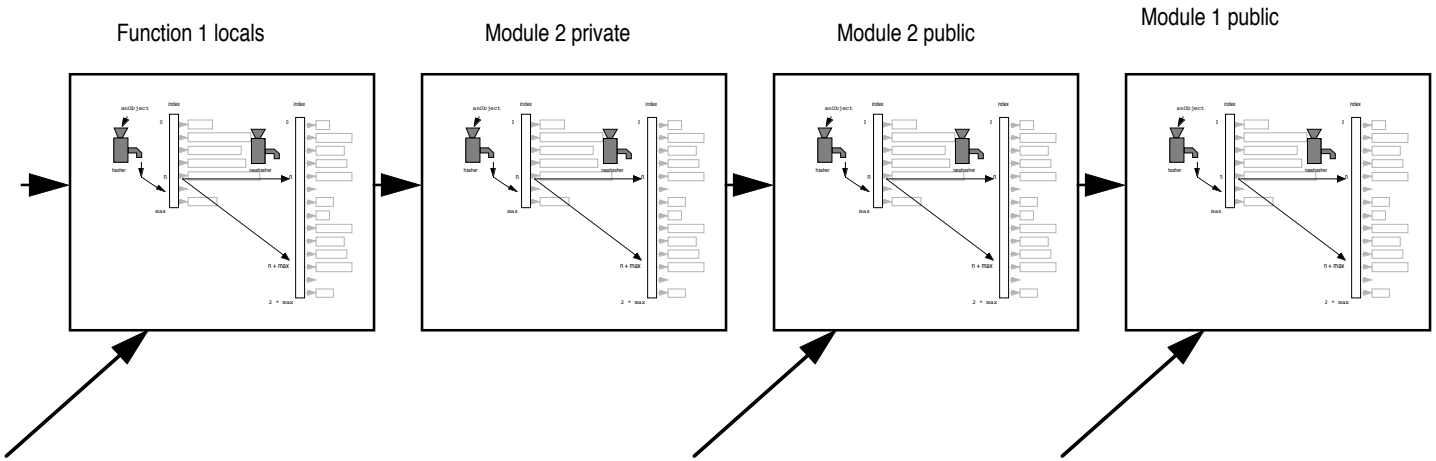
- Can achieve constant average time lookup if buckets have bounded average length.
- Can guarantee this if we periodically double number of hash buckets and re-hash all elements.
  - » Can be done so as to minimize movement of items.

Self Organizing Hash Table



Provided by java.util.Hashtable

Hashtable Chain



Conceptual view of GCL.SymbolTable

A scope (reference) is passed to each statement for lookups and to each declaration for insertion. This is always the topmost scope.

Using the SymbolTable

- When enter a new scope execute scoper.enter (push on a new “scope”)
- When exit that scope execute scoper.exit (pop that “scope”)
- New names are entered into the topmost scope
- Searching starts at any desired scope, as long as you have a reference to that scope. Search continues through following scopes.

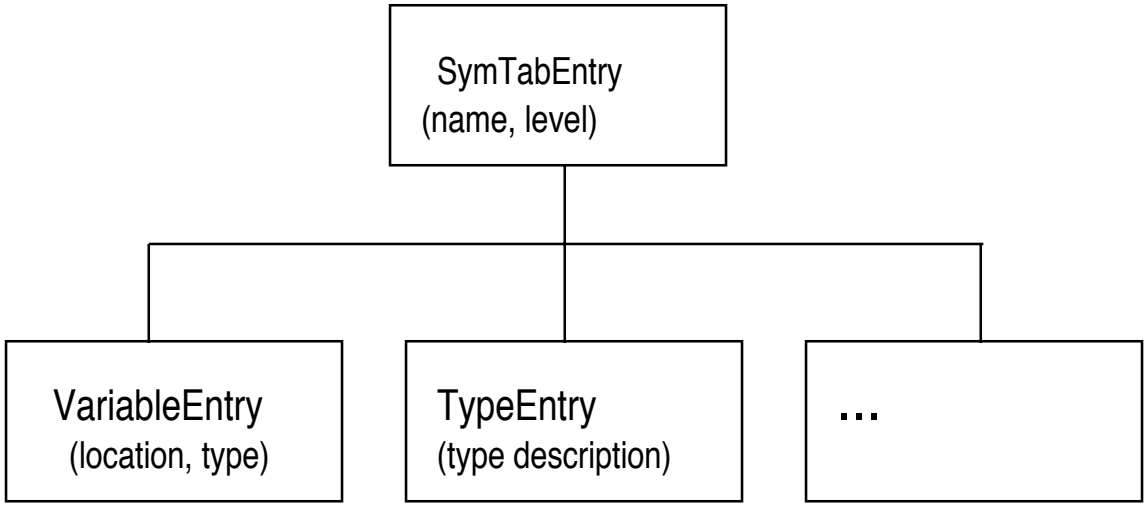
Entering into the SymbolTable

- A given identifier can be entered only once into each scope.
- The chain is to permit the same name to be redefined in a new context in the program.
- A chain is used so that the Symboltable is stack like. Lookups find the most recent definition of a name.

SymbolTable Entries

- Keys in the SymbolTable are Strings, representing the identifiers in the program.
- Values asociated with the identifiers depend on what that identifier represents (variable, type, function...)

SymbolTable Entries



SymbolTable.lookupId returns one of these objects