

CS61B: Lecture 40
Wednesday, December 1, 2010

Today's reading: Goodrich & Tamassia, Sections 14.1.2-14.1.3.

GARBAGE COLLECTION

Objects take up space in memory. If your program creates lots of objects, throws them away, and creates more, you might eventually run out of memory. To reduce the chance that this will happen, Java has garbage collection.

While the Java Virtual Machine (JVM) runs your program, it also spends little bits of time searching for objects that you're no longer using, so it can reclaim their memory for use by other objects.

You don't have to know anything about garbage collection to be an effective Java programmer. But garbage collection is interesting because the JVM uses a lot of hidden data structures to manage memory. These data structures are hidden from your Java program--after all, the JVM, just like any other encapsulated module, should hide the details of how it is implemented. Here's a peek at what's going on under the hood.

Roots and Reachability

Garbage collection's prime directive is to never sweep up an object your program might possibly use or inspect again. These objects are said to be live. The opposite of "live" is garbage--objects that your program cannot reference again. Java's design makes it possible for the JVM to determine whether an object can ever be used again by your program or not.

Garbage collection begins at the roots. A root is any object reference your program can access directly, without going through another object. There are two kinds (that we know about). First, every local variable (including parameters) in every activation record on the program stack is a root if it is a reference. (Primitive types like ints are not roots; only references are.) Second, every class variable (aka "static" field) that is a reference is a root.

An object is live, and should not be garbage collected, if

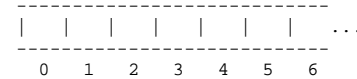
- it is referenced by a root (obviously), or
- it is referenced by a field in another live object.

Note that this definition is recursive. Another way to say it is that an object is live if it is reachable from the roots. If you run depth-first search starting at all the roots, you will visit all the live objects and none of the garbage. And that's exactly what garbage collectors do: run depth-first search from the roots.

Each object has a small tag that allows the garbage collector to mark whether the object has been visited or not. The tag is invisible to your Java program, but it takes a few bits of the object's memory. (This is not the only "hidden" memory Java associates with an object--for example, every object has a hidden label that tells Java what class it's in.)

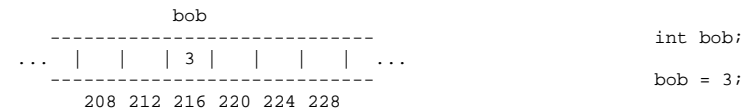
Memory Addresses

In any modern computer, memory is one huge array of bytes with addresses.

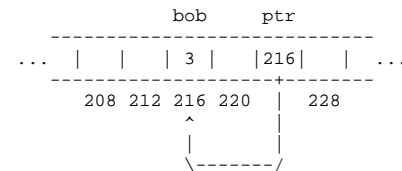


However, modern computers prefer to read or write four bytes at a time, and they do this much faster if the four bytes start at an address divisible by four, so that's how things like ints and floats are stored.

Every time you declare a local variable, you are naming a memory location. (You pick the name, Java picks the address.) An assignment statement writes something into a memory location.



Computers can store memory addresses in memory. To reduce the number of syllables, memory addresses are called pointers for short. Some languages like C allow you to declare variables that are pointers. A pointer field is a memory location that points to another memory location.



Java uses pointers too, but it considers them confidential information, and won't let you print them or look at the numbers directly. Java references are a little bit like pointers, but as we'll learn soon, they're actually more complicated than pointers.

The important point is that your computer's memory is just one giant array that has no structure except the structure you impose on it. Java saves you a huge amount of time and effort by structuring memory for you. Java does this by using hidden pointer-based data structures that you can't access from a Java program.

The advantage of copying garbage collection is that it's fast. The disadvantage is that you effectively cut in half the amount of heap memory available to your program.