[Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-four)]

4.8: Any Last Requests?

We've heard briefly that Java includes a *garbage collector*. Garbage collection is a kind of automatic memory management: when a block of memory is no longer in use, the Java system releases it and adds it back onto the available freestore. When an object is no longer referred to by any reachable variable (for example, the Foo object after the following method returns),

```
void doSomething()
{
  Foo f = new Foo();
  System.out.println(f);
}
```

then the object is marked as eligible for garbage collection. If at some later time the system finds itself short on memory, it can destroy that object and reuse the memory for something else.

Java gives objects a last chance to clean up after themselves before it destroys them: the finalize( ) method. You can define a method with this signature:

```
public void finalize() ...
```

in any class, and the garbage collector promises to call that function immediately before reclaiming the object's memory. Note that *finalizers* (as they're called) aren't precisely like C++ destructors, because in C++, destructors are guaranteed to be called at specific times; in Java, finalizers may never be called (if the system never needs to reuse the memory!). They are *not* guaranteed to be called on exit, either. As a result, finalizers are rarely used in practice. Java classes generally provide explicit cleanup methods as required.

In the following example, an instance of the MergeSort class creates a unique temporary directory for its own use when it is constructed, and deletes the directory when it is done. The assumption is made that sort( ) deletes all temporary files created during a sort.

```
public class MergeSort
{
   private File tmpDirectory;

   public MergeSort()
   {
     tmpDirectory = makeTmpDirectory();
   }

   public File sort(File[] inputs, File output)
   {
     ...
   }
```

```
    public void finalize()
    {
      tmpDirectory.delete();
    }

    private File makeTmpDirectory()
    {
      ...
    }
}
```