Garbage Collection

- In Java, garbage collection provides some automated memory management.
- All objects in Java live on the *heap*.
- The *heap* is also known as the *garbage collectible heap*.
- The purpose of garbage collecting is to find and delete objects that can't be reached.
- Only the *JVM* decides exactly when to run the garbage collector.
- You (the programmer) can only *recommend* when to run the garbage collector.
- You can't know the G.C. algorithm; *maybe* it uses mark and sweep, *maybe* it's generational and/or iterative.
- Objects must be considered *eligible* before they can be garbage collected.
- An object is eligible when no live thread can *reach* it.
- To reach an object, a live thread must have a live, reachable reference variable to that object.
- Java applications can run out of memory.
- Islands of objects can be garbage collected, even though they have references.
- To reiterate: garbage collection can't be forced.
- Request garbage collection with System.gc(); (recommended).
- Class Object has a finalize() method.
- The finalize() method is guaranteed to run *once and only once* before the garbage collector deletes an object.
- Since the garbage collector makes no guarantees, finalize() *may never run*.
- You can uneligibilize an object from within finalize().