2006 Paper 6 Question 4

Concurrent Systems and Applications

(*a*) Java's *Reference Objects* provide a means to interact with the runtime garbage collector.

(*i*) Provide a Java class implementing a *Leaky Array*: a fixed-size, ordered sequence of `Objects`, indexed by integers from zero upwards. The items may be discarded (individually) by the garbage collector when the system is running low on memory. Provide concurrency-safe accessor methods to get and set the items stored in the array. Ensure that any internal data structures cannot be manipulated other than via your accessor methods. It should be possible for classes in other packages to construct new Leaky Array objects and be able to invoke the accessor methods. Derived classes should have concurrency-safe means of determining the size of the array (but be unable to change it) and counting the number of not-null items stored. [6 marks]

(*ii*) Describe carefully when a `Finalizer` method defined on a Java class will be executed. Might two finalizers be executed concurrently? What guarantees does Java make about the execution of finalizers? [5 marks]

(*iii*) How can a `ReferenceQueue` be used to provide more control than finalizer methods? [4 marks]

(*b*) Does Java's garbage collector clean up memory allocated by code in a *native method*? What facilities exist to specify whether or not Java objects created by native code are eligible for garbage collection? [5 marks]