Concurrent Systems and Applications

Example Questions, Section 3, Michaelmas 2001

tim.harris@cl.cam.ac.uk

1. Describe a situation in which the `volatile` keyword is useful in Java.

2. Is it ever *necessary* for a multi-threaded program to use a `volatile` field? If so then give an example where there is no alternative. If not then sketch a general way that some other mechanism can be used instead.

3. Consider the following code used to implement the *singleton* design pattern in a single-threaded application:

```
class SingleThreaded {
  private Helper helper = null;
  public Helper getHelper() {
      if (helper == null)
          helper = new Helper();
      return helper;
    }
  }
```

   Re-implement this class so that it is safe to use in a multi-threaded program – i.e. the `Helper` class should be instantiated exactly once and that instance should be returned by all calls to `getHelper()`.

   (`http://www.cs.umd.edu/ pugh/java/memoryModel/Double CheckedLocking.html` catalogues a large number of broken attempts at this problem in which apparently-efficient designs turn out to be incorrect)

4. Describe how *soft references* can be used to implement a hashtable that discards objects when memory is tight. How can the implementation be extended to approximate a LRU (least-recently-used) policy for discard?

5. Compare and contrast *object finalizers* and *phantom references* as mechanisms to clean-up after objects that have become otherwise-unreachable. For each approach indicate, along with any specific problems or benefits:

   - in which class the clean-up code is located,
   - in which thread or threads it may be executed,
   - what happens if the code blocks or takes a long time to execute.