JAVA THREADS

In “Coping with Java Threads” (Bo Sandén, Apr. 2004, pp. 20-27), I am concerned that a recurring comment is wrong. As a result, the author recommends actions that reduce the robustness of Java applications. Specifically, he recommends against exclusion synchronization for problem-domain resources, instead simulating a semaphore via condition synchronization. This increases the chance of errors that never release the resource, leaving the resource unavailable to the rest of the application.

The author states that exclusion synchronization forces waiting threads to spin. This is incorrect. Exclusion synchronization puts requesting threads on the requested object’s wait set. Three sources describe this behavior: the JVM specification, a JVM implementation, and a sample application.

The JVM spec (java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html) details the monitorenter and monitorexit opcodes, which implement synchronization.

The Java class below also demonstrates this behavior. It was executed on Windows and Linux by Sun’s JVM. In both cases, the process was virtually always in the “sleep” state even though the “waiting” thread was trying to access the Synchronized method “neverReturn.”

```
public class ThreadTest {
  public static class Synchronized {
    public synchronized void neverReturn(String name) {
      while (true) {
        try {
          Thread.sleep(10000);
          System.out.println(name);
        }
        catch (InterruptedException ex){
        }
      }}}
  public static class Sleepy implements Runnable {
    private Synchronized synched;
    private String name;
    public Sleepy(Synchronized synched, String name) {
      this.synched = synched;
      this.name = name;
    }
    public void run() {
      synched.neverReturn(name);
    }}
  public static void main(String[] args) {
    Synchronized synched = new Synchronized();
    Thread sleepy = new Thread(new Sleepy(synched, “sleepy”));
    sleepy.start();
    Thread waiting = new Thread(new Sleepy(synched, “WAITING”));
    waiting.start();
  }
}
```

Sample Java class implementing synchronization.

I respect the accomplishments of the Ada language and community. However, in my opinion, Ada has been overcome by events. Java has generated much research and commercial interest, and, as a result, it continues to improve and mature as a technology.

Jeff D. Sparkman
Huntsville, Ala.
jeff.beth.sparkman@ieee.org

The author responds:

Jeff Sparkman correctly observes that exclusion synchronization does not require threads to test a lock variable repeatedly, which I refer to as “spinning.” Spinning is one implementation technique, but others, involving thread suspension/queuing, are also possible. So threads may indeed be waiting on an object lock as Mr. Sparkman suggests. I oversimplified this and should have been more precise.

The assertion that “exclusion synchronization puts requesting threads on the requested object’s wait set” is incorrect, however. A thread cannot enter the wait set simply by calling a synchronized operation on a locked object or attempting to enter a synchronized block. The mechanism for exclusion synchronization is separate from the wait set mechanism. Thus, the wait set can be optimized for longer waits and multiple waiting threads, and the exclusion synchronization mechanism can be optimized for short waits and high performance.

I disagree with the statement that Ada has been “overtaken by events.” I certainly don’t expect it to be widely used for general-purpose programming. But it is still a living and evolving language, especially for safety-critical systems. A new version of the Ada standard is due in 2005, and “safe subsets” such as the Ravenscar tasking profile place Ada at the leading edge of high-integrity technology.

Even with real-time enhancements, Java is still a largely untested and thus risky technology for real-time and safety-critical applications. As my article attempts to show, the subtleties of Java's thread model are easily misunderstood. Unfortunately, many misuse-prone constructs are inherent in the model and cannot readily be amended.

IT SYSTEMS MANAGEMENT

The authors of "Managing Systems Development" (G. Richardson and B. Ives, IT Systems Perspectives, Mar. 2004, pp. 93-94) made a good point when they stated that management should view IT project development as a business rather than as a technical activity.

More often than not, organizations incorrectly characterize IT services as a department that merely spends money. In some cases, management does not understand that in addition to increasing efficiency, technology improves the entire working environment.

Because their department is mission critical, IT professionals bear tremendous responsibilities, some of which can be quite nerve-racking, especially if they are systems administrators. One way to help the IT department get the resources it needs is to regard it as a business entity based on the concept that maintaining each computer incurs a service fee. Such an approach would help develop the "understanding of an optimal process" that the authors refer to.

IT professionals have an obligation to educate management in how to exploit technology to benefit the company. The IT staff should carefully plan its projects and look at the big picture to find the most cost-effective solution for users.

After all, we work in the IT field not because it is easy, but because it is hard.

Hong-Lok Li
Vancouver, B.C.
lihl@ams.ubc.ca

SECURITY AT WHAT COST?

As an information security practitioner, author, and educator, I simply could not ignore the following questionable statement in Roy Want's otherwise excellent article on RFIDs ("Enabling Ubiquitous Sensing with RFID," Invisible Computing, Apr. 2004, pp. 84-86): "Despite the potential for misuse of invisible tracking, RFID's advantages far outweigh its disadvatanges."

Several questions pop up instantly: What advantages? For whom? At what cost? And whose liberties and privacy is the author ready to instantly sacrifice in search of increased streamlining, efficiency, and insecurity? Yes, insecurity, because it is largely thanks to engineers like Roy Want that we have so much cool technology with so much functionality—and so little assurance.

Why didn't the author cover the plethora of security and privacy issues associated with RFID technology? Is it because he has no answers or because that's not in the manufacturers' best interests? Instead, he makes passing mention of privacy advocates, which gives the impression that they are simply a nuisance.

When will we learn to take responsibility for opening our own Pandora's boxes? When will engineers consider more carefully the consequences of the cool technology they unleash upon us?

In the meantime, poor security practitioners and auditors are left to sort out the mess and take the blame for technology's insecurity.

Edgar Danielyan
edd@danielyan.com

SPAM AND THE LAW

Nowadays, everyone who actively uses the Internet for e-mail will inevitably have bad experiences with spam. Recent news stories describe legal action being undertaken in the US against spammers by ISPs and the FTC under the CAN-SPAM Act of 2003 (www.spamlaws.com/federal/108s877enrolled.pdf). In Europe, legislative measures against spam are also being developed under the EU Directive on Privacy and Electronic Communications. Yet, in many places such as Hong Kong, where I live, there is no antispam legislation.

It's true that the effectiveness of laws restricting spammers remains to be seen. It's also true that existing antispam laws are controversial. But this doesn't mean that we don't need antispam laws.

Many people argue that antispam laws are not going to be effective because the majority of spam doesn't originate in the countries where the laws have been enacted. That's true, but it's not a valid reason for not legislating against spamming. The Internet's global nature requires global cooperation for these legislative efforts to be fully effective. Otherwise spammers can just "escape" to another jurisdiction.

We cannot simply replace the current SMTP architecture with a tightly controlled—or even tolled—mail architecture. Instead, we must establish a norm that requires and empowers ISPs to disconnect spammers, or even spamming countries, from the Internet. A legal framework is an effective means for achieving this.

Moreover, we must not forget that spamming is not a new phenomenon restricted only to the Internet—we have junk faxes as well. Junk faxing is highly localized—with a humble computer and a modem, senders of junk faxes can create a nightmare in countries with toll-free local calls.

Does anyone really believe that antispam—or "antiunsolicited communications," to be exact—laws are not necessary?

Davy Cheung
Hong Kong
davyc@ieee.org

We welcome your letters. Send them to computer@computer.org. Letters are subject to editing for style, clarity, and length.