quora.com

(87) Peter Schaeffer's answer to What is concurrency in programming?

3-4 minutes

The question was about concurrency in general, not just Java. My answer below is Java specific. However, the concepts apply to programming in general, not just in Java.

As always, check the other answers, in addition to my comments. They have a wealth of useful information. My concept of concurrency is that basically thread level concurrency. In other words, the Java language allows for the creation of multiple threads in a Java program and the ability of a computer to run more than one thread at (literally) the same time.

1. All computers that can run Java support multiple threads. Mechanisms for creating multiple threads and synchronizing multiple threads are built into the standard Java language. Note that this is true even on machines with just one physical core and no hyper threading.

2. Some machines that run Java, have more than one core

and/or hyper threading. On these machines more than one Java thread can actually be executing at the same time. On machines with just one core and no hyper threading, simultaneous execution is just simulated by running one thread at a time and (quickly) switching between threads.

Note that multi-core machines are now very, very common even on phones. For example, my machine (a laptop) has 4 physical cores and two logical cores per physical core. At this point you can assume that all machines have multiple cores (a few low-end machines don't).

3. A crucial point is that all Java threads in one program share the same memory space. In other words, they "see" and can update the same data variables. This isn't quite true, but is a very important idea.

4. As stated above, multiple Java threads share the same memory space and data values. However, this sharing isn't quite perfect and the limitations of memory sharing are incredibly important in writing and debugging Java programs that have more than one thread. To use a simple example, if thread A changes a data value and thread B reads the data value, thread B may or may not get the new updated value for the data value.

5. The rules for memory sharing (when updated values "appear" to other threads) are complex but very, very important. See the "Java memory model" for the details of how the JMM works. Note that Java supports atomic values with

special concurrency properties (and big performance issues).

6. Java has many other concurrency mechanism. These include the synchronized keyword, locks, and queues. Quite a few Java data structures are thread-safe. You must check each data structure you are using to determine if it is thread-safe or not. Note that some Java concurrency mechanisms look easy to use (and are), but have very bad performance implications.

7. Java has memory barriers and fences. This topic is well beyond an answer in Quora. However, you really need to understand these (hard) topics if you are going to build concurrent programs in Java.

If your Java program has just one thread, you can ignore all of this stuff. However, if you are designing/creating/building a Java program with multiple threads, then these issues are vital. They are not easy to deal with. Indeed, the sheer difficulty of building multi-threaded programs is the key reason that they are not very common.

2.7k views ·

View Upvoters

· Answer requested by Daniel Noor