1.1 Lecture Summary

1.1 Java Threads

Lecture Summary: In this lecture, we learned the concept of threads as lower-level building blocks for concurrent programs. A unique aspect of Java compared to prior mainstream programming languages is that Java included the notions of threads (as instances of the `java.lang.Thread` class) in its language definition right from the start.

When an instance of `Thread` is *created* (via a `new` operation), it does not start executing right away; instead, it can only start executing when its `start()` method is invoked. The statement or computation to be executed by the thread is specified as a parameter to the constructor.

The Thread class also includes a *wait* operation in the form of a `join()` method. If thread `t0` performs a `t1.join()` call, thread `t0` will be forced to wait until thread `t1` completes, after which point it can safely access any values computed by thread `t1`. Since there is no restriction on which thread can perform a `join` on which other thread, it is possible for a programmer to erroneously create a *deadlock cycle* with `join` operations. (A deadlock occurs when two threads wait for each other indefinitely, so that neither can make any progress.)

Further Reading:

1. Wikipedia article on Threads
2. Tutorial on Java threads
3. Documentation on Thread class in Java 8

Mark as completed