Thread Pools

CSCI 201L

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

HTTP://WWW-SCF.USC.EDU/~CSCI201
Outline

- Thread Pools
Thread Pool Overview

- If you need to create a lot of threads, it may not be the most efficient to create them by instantiating them all as threads and starting them
  - There may be performance issues, such as limiting throughput, if there are too many threads

- A thread pool helps to manage the number of threads executing concurrently
  - If a thread in a thread pool completes execution, it can be reused instead of needing to create another thread
  - If a thread terminates due to failing, another thread will be created to replace it

- The `execute` method will execute the thread at some time in the future, determined by the `Executor`
Thread Management

- Thread pools also give us more information about the state of threads
- We are able to find out if all of the threads that were invoked have completed
  - `ExecutorService.isTerminated()`
- We can also make sure no additional threads can be created by terminating the pool (though existing threads will still be able to complete)
  - `ExecutorService.shutdown()`
  - `ExecutorService.shutdownNow()` will kill all currently executing threads in the pool
Executors Class

- The Executors class provides factory and utility methods for executing threads

  - newCachedThreadPool()
    › Creates a thread pool that creates new threads as needed and reuses previously constructed threads when they are available

  - newFixedThreadPool(int numThreads)
    › Creates a thread pool that reuses a fixed number of threads. At any point, a maximum of numThreads will be executing

  - newScheduledThreadPool(int corePoolSize)
    › Creates a thread pool that can schedule commands to run after a given delay or to execute periodically

  - newSingleThreadExecutor()
    › Creates an Executor that uses a single worker thread
import java.util.concurrent.ExecutorService;
import java.util.concurrentExecutors;

public class Test {
  public static void main(String[] args) {
    System.out.println("First line");
    ExecutorService executor = Executors.newFixedThreadPool(3);
    executor.execute(new TestThread('a'));
    executor.execute(new TestThread('b'));
    executor.execute(new TestThread('c'));
    executor.shutdown();// threads will still complete, executor.shutdownNow() otherwise
    System.out.println("Last line");
  }
}

class TestThread extends Thread {
  private char c;
  public TestThread(char c) {
    this.c = c;
  }
  public void run() {
    for (int i=0; i < 20; i++) {
      System.out.print(i + "" + c + "");
    }
    System.out.println("";
  }
}

First line
0b 0c 0a 1c 1b 2c 1a 3c 2b 4c 2a Last line
3a 5c 3b 6c 4a 7c 4b 8c 5a 9c 5b 10c 6a 11c 6b 12c 7a 13c 7b 14c 8a 15c 8b 16c 9a 17c 9b 18c 10a 19c 10b
11a 12a 13a 14a 15a 16a 11b 12b 13b 14b 17a 15b 18a 16b 19a
17b 18b 19b