Thread Pools

CSCI 201
Principles of Software Development

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

USC Viterbi
School of Engineering

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 alive
- `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

Thread Pool Example

```
1   import java.util.concurrent.ExecutorService;
2   import java.util.concurrent.Executors;
3
4   public class Test {
5     public static void main(String[] args) {
6       System.out.println("First line");
7       ExecutorService executor = Executors.newFixedThreadPool(3);
8       executor.execute(new TestThread('a'));
9       executor.execute(new TestThread('b'));
10      executor.execute(new TestThread('c'));
11      executor.shutdown(); // threads will still complete, executor.shutdownNow() otherwise
12      while (!executor.isTerminated()) {
13        Thread.yield();
14      }
15      System.out.println("Last line");
16    }
17  }
18  class TestThread extends Thread {
19    private char c;
20    public TestThread(char c) {
21      this.c = c;
22    }
23    public void run() {
24      for (int i=0; i < 20; i++) {
25        System.out.print(i + "" + c + " ");
26      }
27      System.out.println("");
28    }
29  }
```

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

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