4/6/2021

Bipin Kumar

shael.dhn88@gmail.com

**MultiThreading**

MultiThreading notes

**Process and Thread:**

Operating System:

* Interface between computer user and computer hardware.
* User don’t have to deal directly with hardware.
* Responsibilities: File management, memory management, process management etc.
* E.g. Mac OS, Window, Linux etc.

Process is an instance of a program in execution. A process consists of File, data and heap. It is slower and required more computing resource.

A thread is a light weighted process, one process can have multiple thread. It consists of a program counter, a stack, and a set of registers.

Thread scheduling algorithm:

1. First come first served
2. Shortest job first
3. Priority scheduling

**Context switch:** Context switching involves removing a running process or thread from the CPU, saving its state, and loading up another process or thread.

It is the price we pay for concurrency, too many threads cause thrashing. Time sink, should be minimized to gain efficiency and get more productive work done. Context switching between 2 threads of same process is cheaper than context switching between processes.

* Implementing runnable is better
* Composition is preferred over inheritance
* Java supports single inheritance so we can only extend one class
* Instantiating interface gives a cleaner separation between code and the implementation.
* In case of Thread it creates unique object for each thread where as in case of runnable it shares same object.

Ways of creating a thread:

1. By Extending Thread class and overriding run method.
2. By Implementing Runnable interface and overriding run method.
3. By using anonymous inner class

Why do we need ExecutorService?

* As a Programmers we had to worry about
* Starting threads
* Life cycle of threads
* Cannot restart threads
* Programmer focus should be on WHAT needs to be done.
* HOW it needs to be done can be outsourced to a service
* This service also maintains a thread pool: REUSE threads and reduce thread spawning cost.

How do we use ExecutorService?

* Create executor service (Thread pool)
* Submit tasks to the executor service
* ExecutorService:
  + Allocates threads to tasks: keeps track
  + Reuses threads
  + Repeat

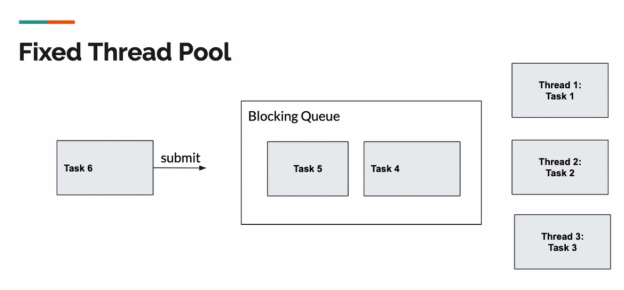
e.g.

ExecutorService executor = Executors.*newFixedThreadPool*(3);  
ThreadNamePrinter printer = new ThreadNamePrinter();  
for(int i = 0; i <3;i++){  
 executor.submit(printer);  
}  
  
executor.shutdown();

Types of thread pool:

1. Fixed thread pool:

* Fixed number of threads
* Tasks submitted to a blocking queue
* Tasks picked up from blocking queue by idle threads.



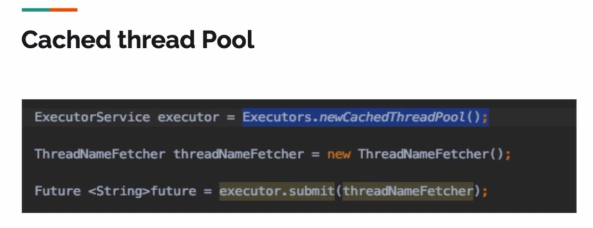


1. Single threaded executor

* Just one thread
* Used so that tasks can be carried out sequentially.

1. Cached Thread pool

* Threads created if no idle threads available
* Tasks submitted to a synchronous queue (Capacity of 1 item)
* Lots of thread can be created
* Idle threads killed in 60 seconds



1. Scheduled executor service:

* Used for repetitive tasks to be scheduled rather than carried out immediately
* Uses runnable
* Fixed pool size
* Fixed rate
* Fixed delay

