Programação_Concorrente-Studies/ Aula02: Aprofundando sobre Estados de uma Thread, Prioridades de Thread e Agendamento de Threads

Voltando a primeira aula:

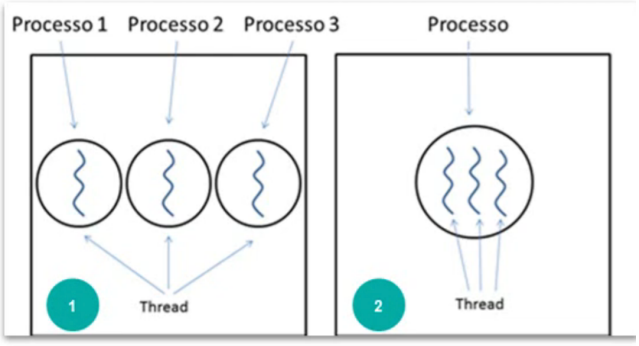
Programa: conjunto de instruções em uma linguagem de alto nível ou de máquina.

Processo: resultado da execução do programa.

01 | Sequencial: uma Thread (Single-Thread)

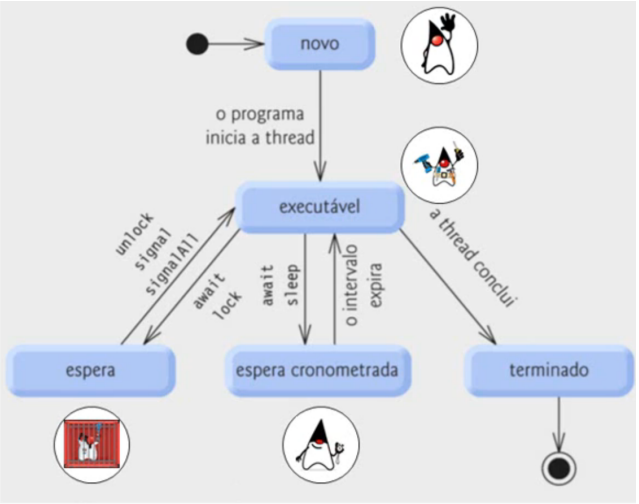
02 | Concorrente: várias Threads (Multi-Thread)

Threads: linhas de execução separadas



Estados de thread

- uma nova thread inicia seu ciclo de vida no estado **novo**
- permanece nesse estado até o programa iniciar a thread, colocando-a no estado **executável**
- entra no estado de **espera** a fim de esperar que uma outra thread realize uma tarefa.
- entra em **espera cronometrada** para esperar uma outra thread ou para transcorrer um determinado período de tempo;
- uma thread executável transita para o estado **bloqueado** quando tenta realizar uma tarefa que não pode ser completada imediatamente e deve esperar temporariamente até que essa tarefa seja concluída;
- quando uma thread no estado executável completa sua tarefa ela entra no estado **terminado**.



Ciclo de vida de thread: visão do programador

Em programação concorrente, uma thread pode estar em vários estados, dependendo de sua execução e interação com outras threads e o sistema operacional. No entanto, os estados principais de uma thread em muitas linguagens de programação, incluindo Java, são os seguintes:

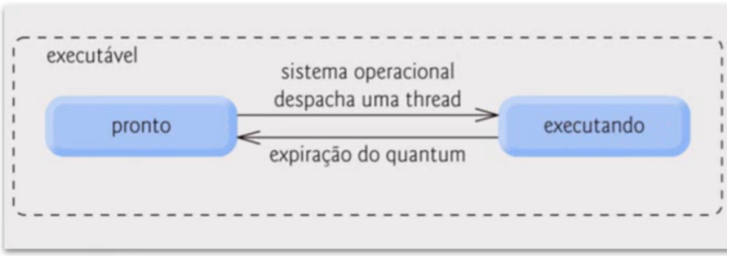
- **New (Novo)**: Neste estado, a thread foi criada, mas ainda não foi iniciada. Isso geralmente ocorre depois de uma chamada ao construtor da classe Thread, mas antes de chamar o método start().
- **Runnable (Executável)**: Quando a thread está pronta para ser executada, mas o sistema operacional ainda não a selecionou para execução. Isso inclui o tempo em que a thread está sendo executada pela CPU e também quando está pronta para ser executada, mas está aguardando sua vez.
- **Blocked (Bloqueado)**: Uma thread pode ser temporariamente impedida de executar enquanto espera por um recurso externo, como entrada/saída de dados, ou porque está esperando que outra thread libere um recurso que está segurando. Uma thread bloqueada não pode prosseguir até que a condição que a bloqueou seja resolvida.
- **Waiting (Esperando)**: Uma thread pode entrar neste estado quando está esperando uma determinada condição para ser satisfeita, como o término de outra thread ou uma notificação específica. A thread permanecerá nesse estado até que seja notificada ou interrompida.
- **Timed Waiting (Esperando por tempo limitado)**: Similar ao estado de espera, mas com uma diferença: a thread está esperando por um determinado período de tempo antes de continuar sua execução. Por exemplo, isso pode ocorrer quando uma thread chama o método Thread.sleep() ou espera por uma condição com um limite de tempo definido.
- **Terminated (Terminado)**: Neste estado, a thread concluiu sua execução e não está mais ativa. Isso pode ocorrer quando a execução da thread chega ao fim naturalmente ou quando é interrompida ou encerrada por algum motivo.

Estes são os principais estados de uma thread em muitos sistemas operacionais e linguagens de programação. No entanto, as implementações específicas podem ter variações ou estados adicionais, dependendo do sistema e do ambiente de execução.

Porém para o seu Sistema Operacional, diferente da CPU ele entende de uma forma muito mais simplificada os Estados de uma Thread:

Estados de thread

- **pronto:** uma thread nesse estado não está esperando uma outra thread, mas está esperando que o sistema operacional atribua a thread a um processador;
- **em execução:** uma thread nesse estado tem atualmente um processador e está executando. Uma thread no estado em execução frequentemente utiliza uma pequena quantidade de tempo de processador chamada fração de tempo, ou *quantum*, antes de migrar de volta para o estado pronto.



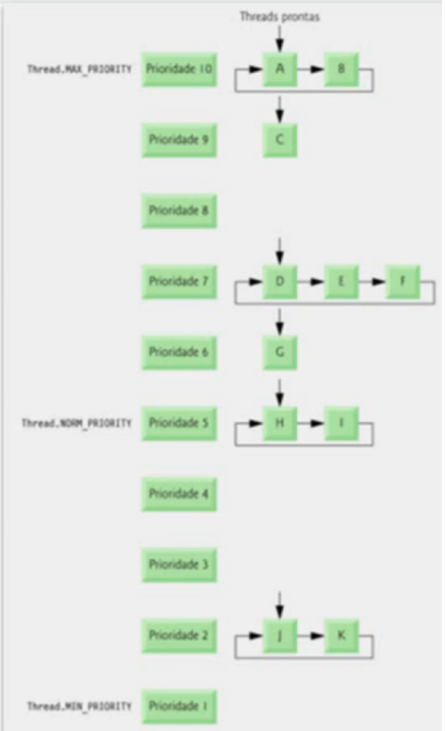
Ciclo de vida de thread: visão do sistema operacional

- A thread executa durante um quantum de tempo.
- Quando esse quantum acaba ela volta ao estado de aguardo para despacho, pois tem outra Thread sendo utilizada por aquele processador.

Prioridades e Agendamentos de Threads:

Prioridades e Agendamentos

- Cada thread Java tem uma prioridade;
- As prioridades do Java estão no intervalo entre:
  - MIN_PRIORITY (uma constante de 1) e
  - MAX_PRIORITY (uma constante de 10);
- As threads com prioridade mais alta são mais importantes e terão um processador alocado antes das threads com prioridades mais baixas;
- A prioridade padrão é NORM_PRIORITY (constante de 5).
- Em Java: `public void setPriority(xxx)`



Quando criamos uma thread ela entra com a NORM_PRIORITY(5), porém esses recursos de prioridade são só uma opção dada ao programador para ele informar para o SistemaOperacional, qual thread deve ter prioridade sobre a outra, porém isso não significa que o sistema operacional não vai alocar tempo para as outras threads, logo você não controla se a thread com maior prioridade vai executar completamente antes de uma thread com menor prioridade. Logo, se você tiver um programa concorrente a "ordem de execução" em si do seu programa não pode ser muito necessária, portanto se você necessitar de uma ordem de execução de instruções extritamente específica , é melhor que esse programa seja sequencial, caso não seja assim ele pode ser concorrente com o paralelismo.

Agendamento de Threads:

Prioridades e Agendamentos

Thread Scheduler: Agendador de Thread em um SO

- determina qual thread é executada em seguida;
- uma implementação simples executa threads com a mesma prioridade no estilo rodízio;
- threads de prioridade mais alta podem fazer **preempção** da thread atualmente **em execução**.
- em alguns casos, as threads de prioridade alta podem adiar indefinidamente threads de prioridade mais baixa; o que também é conhecido como **inanição**.

- Esse Thread Scheduler tem seu algoritmo próprio, que é controlado pelo Sistema Operacional.
- Threads de prioridade mais alta podem fazer preempção de uma thread que está executando, mas oq seria essa preempção?,a preempção seria o agendador de Thread parar a thread em execução e destinar quantum de tempo para a Thread de prioridade mais alta que estava aguardando.

- Porém essa preempção pode gerar em uma Thread de menor prioridade a Inanição, que no caso seria o ato de essa thread sempre ficar aguardando um quantum de tempo do SO, enquanto as outras Threads de prioridades mais altas revezam tempo de execução no processador , e essa thread de menor prioridade nunca recebe um quantum de tempo.

#Concorrent