Matéria:

* *Thread* em Java

1. **– *Threads***

Recorde que uma *thread* pode ser vista como uma linha de execução de um processo. Um processo é algo que dispõe de um segmento de memória próprio e que executa de forma independente dos restantes processos. Um processo contém pelo menos uma linha de execução (ou *thread)*. *Thread* são frequentemente consideradas como *lightweight processes* já que tanto os processos como as *thread* possibilitam a execução de código. Contudo, uma *thread* existe dentro do contexto de um processo e partilha o segmento de memória, ficheiros abertos e demais recursos, com as restantes *thread* (caso existam) do mesmo processo.

* 1. **– Estados de uma *thread***

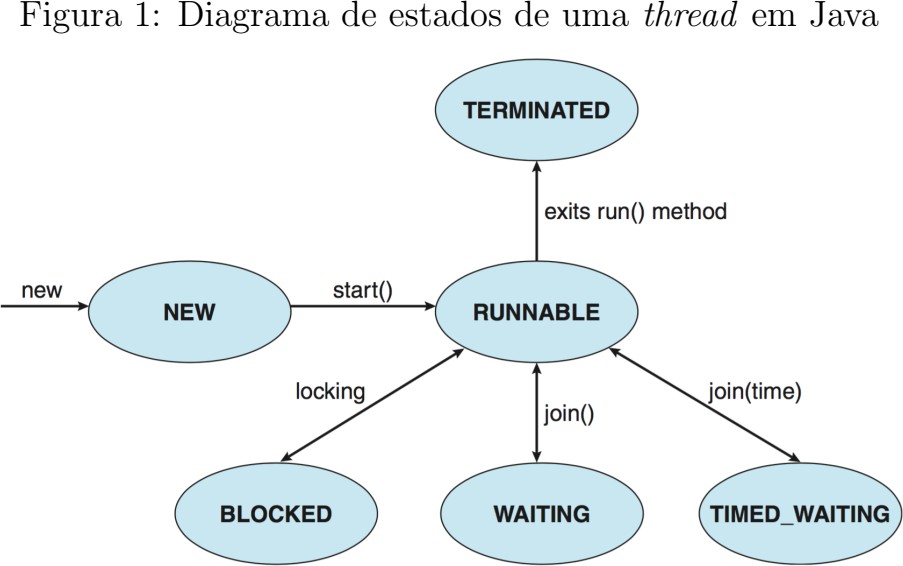
Uma *thread* em Java pode estar num de seis estados:

* + - **Nova:** *Thread* foi criada, mas ainda não foi iniciada.
    - **Pronta**: *Thread* já foi iniciada, através da invocação do método *start()*. A JVM não distingue *threads* em execução das prontas a executar.
    - **Bloqueada**: *Thread* está bloqueada, à espera de obter um *lock* (Ferramenta de sincronização de execução de *threads*);
    - **Aguarda**: *Thread* aguarda por uma ação de outra *thread*. Por exemplo, sempre que uma

*thread* invoca o método *join()* entra neste estado.

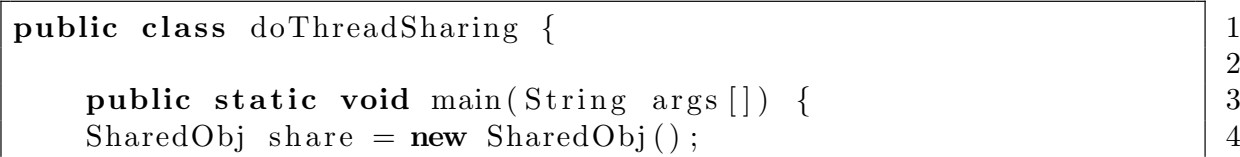
* + - **Aguarda temporariamente:** Semelhante ao anterior, contudo aguarda-se durante até um período máximo. Por exemplo, o método *join()* aceita como argumento o tempo máximo enquanto vai esperar;
    - **Concluída:** *Thread* concluiu o seu método *run()*.

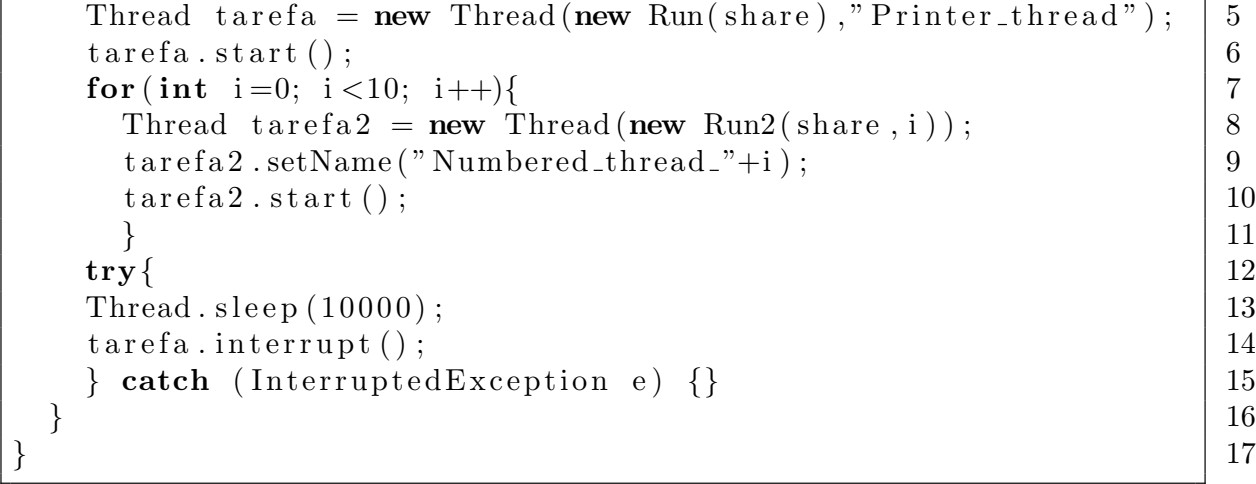
A classe *Thread* possibilita vários métodos para se obter o estado de uma *thread*. O método *isAlive()* permite saber se uma *thread* já iniciou o seu método *run()*. Alternativamente, poderá ser utilizado o método *getState()* que devolve o estado da *thread.*



* 1. **– *Threads* com nome**

A classe *Thread* permite que se atribuam nomes às *thread* no momento da sua criação. Também é possível fazê-lo mais tarde, recorrendo ao método *setName().* A atribuição de nomes às *thread* é útil para permitir a sua distinção em tempo de execução. O método *getName()* devolve o nome que foi atribuído à da *thread.* A classe *doThreadSharing* apresentada na Listagem 2 demonstra a utilização destas duas formas de nomear uma *thread.* Na linha 5 é atribuído o nome “Printer thread” recorrendo ao construtor da classe *Thread.* Já na linha 9 pode ver-se a atribuição de nome recorrendo ao método *setName().*

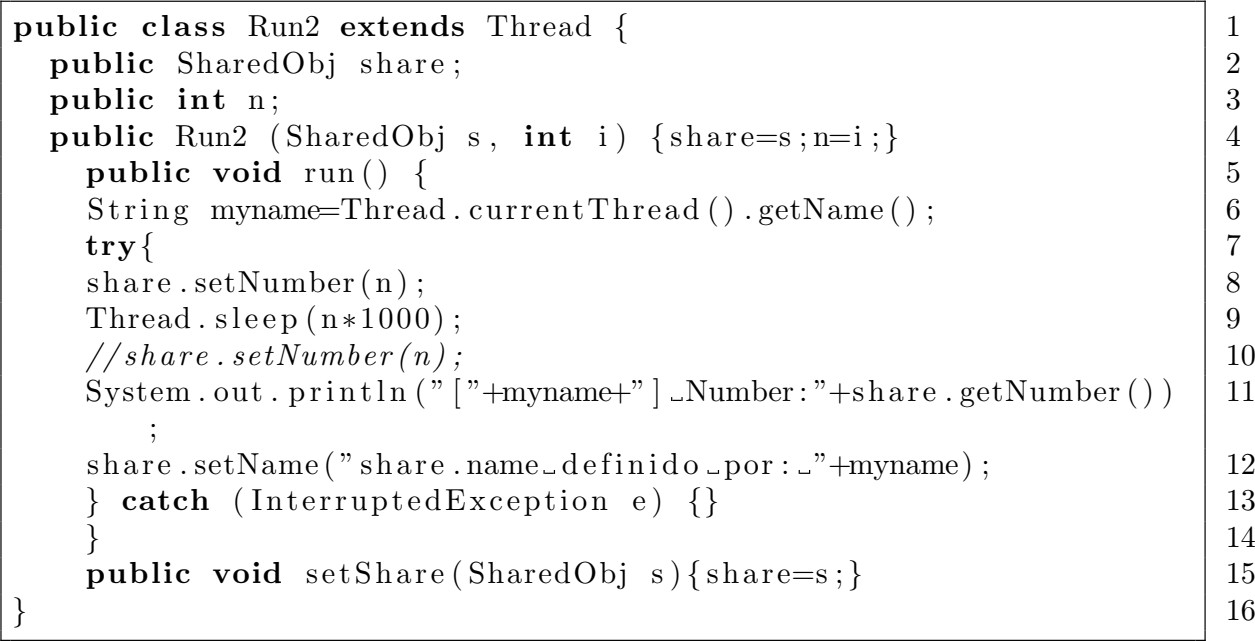
**



Listagem 2: Classe *doThreadSharing*

A classe *Run2*, apresentada na listagem 3, faz uso da capacidade de atribuição de nomes às

*thread* para identificar as linhas que imprimem para o ecrã, como pode ver na sua linha 11.

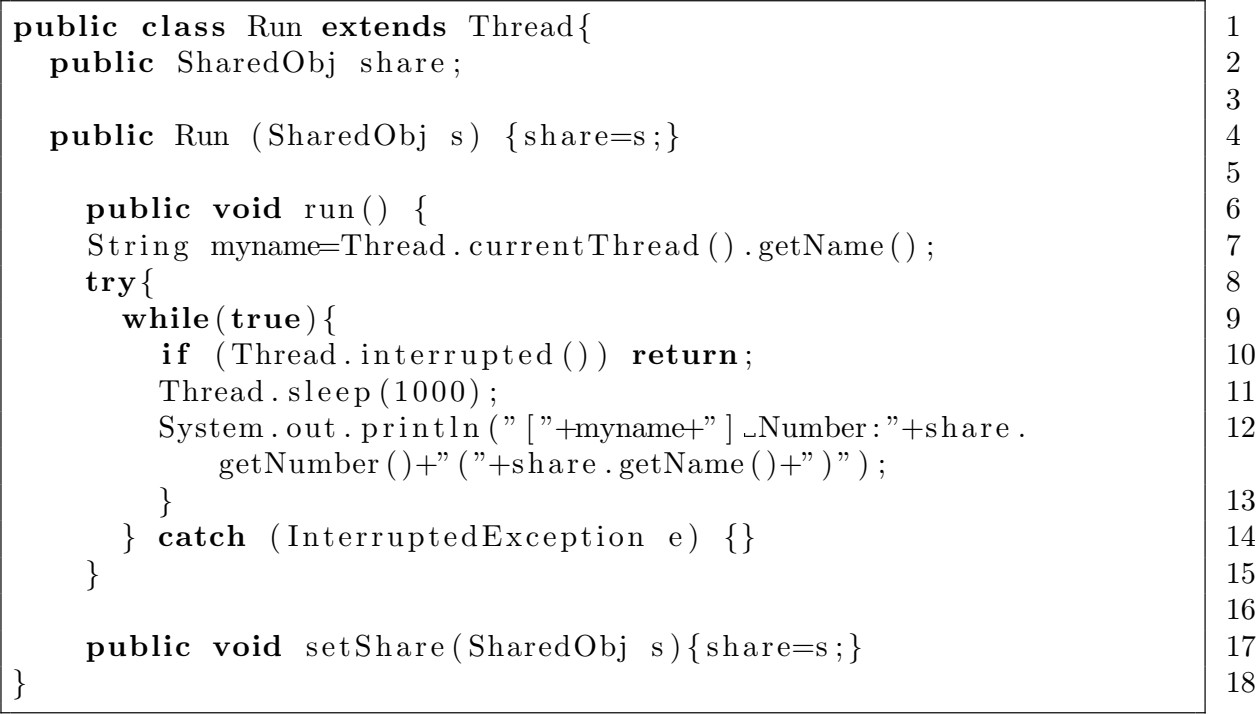


Listagem 3: Classe *Run2*

# – Interrupção da execução de *threads*

A execução de uma *thread* pode ser interrompida por recurso ao método *stop()* contudo este método encontra-se em desuso (*deprecated*) e será descontinuado no futuro. A forma preferível de se interromper uma *thread* é por recurso ao seu método *interrupt().* Este método não interrompe a *thread* de imediato, serve antes como uma forma de sinalizar que se pretende que a mesma seja interrompida. Cada *thread* deve, recorrendo ao método

*Thread.interrupted(),* verificar se deve terminar e, se for o caso, terminar ordeiramente. Este comportamento é adotado pela classe *Run* como se pode ver pela linha 8 da Listagem 4.



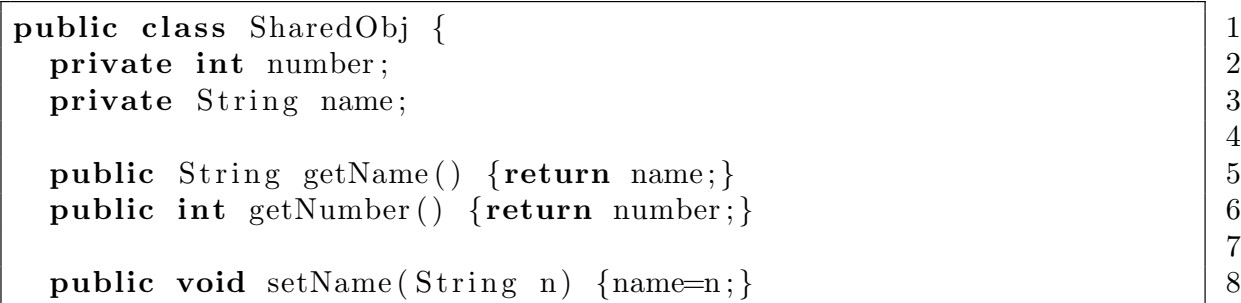
Listagem 4: Classe *Run*

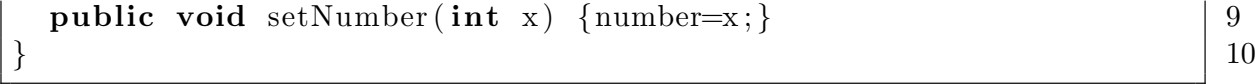
# – Suspensão temporária da execução de *threads*

A classe *Thread* possibilita a suspensão temporária, para tal basta que se invoque o seu método *sleep()* que recebe um número como argumento. O argumento representa o período de suspensão, em milissegundos. Ambas as classes *doThreadSharing* e *Run* fazem uso desta possibilidade.

# – Partilha de dados entre *thread*

A partilha de dados entre *thread* é algo que é natural já que todas as *thread* criadas no contexto de um mesmo processo, partilham o mesmo segmento de memória. Ou seja, desde que uma *thread* tenha acesso à referência de um objeto, pode utilizá-lo. A classe *doThreadSharing* cria um objeto do tipo *SharedObj* (linha 4 da Listagem 2) e passa-o para todas as *thread* que cria (linhas 5 e 8 da Listagem 2), usando o respetivo construtor. A classe *SharedObj* é apresentada na Listagem 5.





# – Esperar pelo fim de uma *thread*

A classe *Thread* contém o método *join()* que, após invocado, vai bloquear o objeto que invocou o *join()* até que a respetiva *thread* termine. Em suma, o método *join()* da classe *Thread* tem um comportamento similar ao do método *waitFor()* da classe *Process* (visto em aulas anteriores).

# 2– Exercício

1. Compile as classes *doThreadSharing, SharedObj, Run* e *Run2*. Execute-as e observe atentamente o seu *output*. As linhas impressas pelas *thread* vem corretamente numeradas (0,1,2,3,4,5,6,7,8 e 9) ou surgem sobretudo linhas numeradas com 9? Porquê?