Matéria:

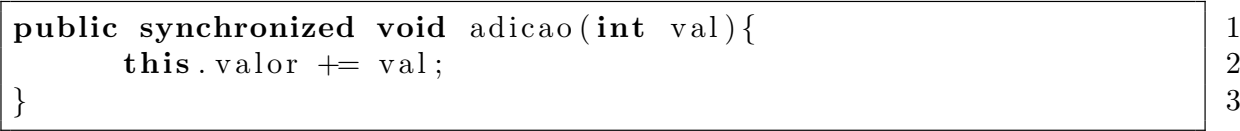
* Concorrência entre *Thread* em Java

1. **– *Race conditions* (competição)**

Uma situação de competição (ou *race condition*) é aquela em que o resultado, da execução de várias *thread* que operam sobre um mesmo recurso, é diferente dependendo da ordem de execução das várias *thread*. Note-se que a existência de múltiplas *thread* simultaneamente num mesmo processo não é sinónimo de situações de competição. O problema surge quando duas, ou mais, *thread* tentam alterar o estado de um recurso partilhado em simultâneo. O último exercício da ficha prática anterior (Ficha Prática n. º4) expõe este problema. A solução para evitar situações de competição recai na sincronização da execução das várias operações de alteração de estado de recursos partilhados. Por outras palavras, a solução passa por evitar que as alterações de estado de um recurso partilhado ocorram em simultâneo, forçando uma determinada ordem de execução das operações de alteração. A sincronização de *thread* em Java pode ser conseguida de várias formas. Os mecanismos *synchronized* e *Locks* são dois exemplos de formas de se conseguir resolver situações de competição.

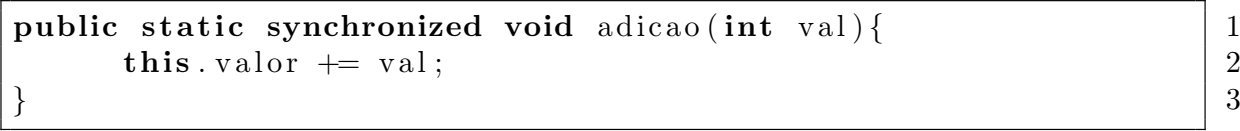
1. **– Palavra-chave: *synchronized***

Blocos de código em Java que se quer que executem livre de problemas relacionados com competição podem ser implementados com recurso à palavra-chave *synchronized*. Esta palavra- chave pode ser utilizada para identificar partes de código ou métodos completos, como blocos *synchronized*. A JVM assegurará que apenas existirá uma *thread* a executar esse bloco em cada instante de tempo, bloqueando as restantes *thread* para identificar as linhas que imprimem para o ecrã, como pode ver na sua linha 11.



Listagem 1: Exemplo aplicacional do método *synchronized*

A Listagem 1 exemplifica como se pode definir um método como *synchronized*, bastando par tal a adição da palavra *synchronized* à definição do método. Note-se que aqui que o sincronismo é garantido no contexto do objeto que tem o método *synchronized*. Ou seja, a JVM resolve situações de competição entre *thread* de um mesmo objeto, mas não entre objetos diferentes. A Listagem 2 apresenta um método estático também *synchronized*. Nestas situações, o sincronismo é garantido no contexto da classe e não do objeto.



Listagem 2: Exemplo aplicacional do método estático *synchronized*

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Listagem 3: Classes *Counter, CounterThread* e *Exer*

**3– Exercício**

* 1. As classes apresentadas na Listagem 3 (no final da ficha) geram uma situação de competição (ou *race condition*). Identifique qual é método onde tal competição acontece. Resolva a situação de competição recorrendo a uma, ou mais, invocações ao método *sleep()*;

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

* 1. Resolva novamente o último exercício da ficha anterior (*Threads* em Java, Ficha Prática n.º 4), removendo todas as invocações ao método *sleep()* e recorrendo à palavra chave *synchronized*;

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

* 1. Contabilize o tempo de execução de ambas as soluções das alíneas anteriores. Compare os valores obtidos;

Os valores dos dois programas são iguais, mas no primeiro, para evitar race condition, adicionamos o método sleep() que atrasou a execução da thread e demorou a ser apresentado o resultado na linha de comandos; já com o segundo programa, ao utilizar o método synchronized, o valor aparece de imediato na linha de comandos.

* 1. Escreva um programa que lance 10 *threads*. Cada *thread* deve imprimir para o ecrã 100 números consecutivos, começando num valor definido com recurso ao método construtor da *thread.* Garanta que os números são apresentados no ecrã de forma ordenada, entre 0 e 999. Garanta ainda que surge a mensagem “Impressão terminada" no final de tudo;

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

* 1. Escreva um programa que recorra a cinco *threads*, além da *thread* principal, para acelerar a execução de um algoritmo de pesquisa pelo maior valor. Crie um *array* com 1000 posições preenchida com valores inteiros. A cada *thread* compete a análise da sua parte do *array* (200 posições). À *thread* principal compete a análise das respostas às cinco *threads*. A *thread* principal só deve iniciar a sua análise após terem terminado todas as restantes *thread*.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente