

# Exclusão Mútua

Grupo de Sistemas Distribuídos  
Universidade do Minho

## Objectivos

Granularidade de exclusão mútua. Observação de *deadlocks* e soluções para evitar a sua ocorrência.

## Mecanismos

- Classe `ReentrantLock` de `java.util.concurrent.locks`
  - Métodos `void lock()` e `void unlock()`.

O lock é reentrante, permitindo que uma thread adquira com sucesso um lock já por ela detido.

## Exercícios propostos

1 Observe o código em `Bank.java` que representa um banco com várias contas independentes, suportando as operações:

```
int balance(int id);  
boolean deposit(int id, int value);  
boolean withdraw (int id, int value);
```

Complemente o código por forma a aplicar a exclusão mútua necessária.

2 Acrescente o método `transfer` à classe `Bank` como simples composição dos métodos de levantamento e depósito da classe `Bank`, e o método `totalBalance` que soma todos os saldos. Passando a suportar as operações:

```
int balance(int id);  
boolean deposit(int id, int value);  
boolean withdraw (int id, int value);  
boolean transfer (int from, int to, int value);  
int totalBalance();
```

Teste a sua implementação correndo o código em `BankTest.java` e `BankTest2.java`. Observe e procure explicar as diferenças entre os resultados dos dois testes, e modifique a implementação de modo a ambos os testes passarem com sucesso.

**3** Tendo em conta que a solução anterior é pouco eficiente, devido ao uso de exclusão mútua global a todo o banco, ajuste a sua implementação utilizando exclusão mútua ao nível das contas individuais.

Teste de novo a sua implementação correndo os dois programas de teste. Verifique possíveis bloqueios e diferenças no tempo de execução.