Criação de Threads. Partilha de estado. Observação de corridas. Exclusão mútua.

Grupo de Sistemas Distribuídos
Universidade do Minho

Objectivos

Utilização de mecanismos para criação de threads em Java. Passagem de objectos para serem partilhados entre threads. Observação de corridas resultantes da manipulação concorrente de estado partilhado. Uso das facilidades de biblioteca em Java relativas a locks para obtenção de exclusão mútua.

Mecanismos

- Interface `java.lang.Runnable`
  - Interface a implementar por classes com instâncias executáveis por threads
  - Implementações devem disponibilizar um método `public void run()`.
- Classe `java.lang.Thread`
  - Implementa `java.lang.Runnable`.
  - Classes que extendam `Thread` devem reimplementar o método `public void run()`.
  - Métodos `void start()` e `void join()`.
- Classe `ReentrantLock` de `java.util.concurrent.locks`
  - Métodos `void lock()` e `void unlock()`.

Exercícios propostos

1 Consulte a documentação oficial do Java sobre `Thread` e escreva um programa que crie `N` threads (para `N=10`) em que cada uma escreve os números de 1 a `I` (para `I=100`). O *thread principal* do programa deve aguardar que todas as threads criadas terminem e escrever `"fim"`. Podem dar continuidade ao fragmento de código `inc-and-print.java` fornecido com o guião.

2 Ao longo dos próximos guiões iremos implementar várias versões de um banco. Considere uma primeira versão mais simples de um banco com uma única conta. Observe o código em `banco-uma-conta.java` que fornece as operações:

```
int balance();
boolean deposit(int value);
```

Crie a partir dele um programa em que `N=10` threads concorrentes fazem `I=1000` depósitos de um mesmo valor `V=100` nessa conta. Quando todos os threads terminarem, observe se o valor final na conta é o valor esperado de 1,000,000.

3 Partindo do exercício anterior e usando um `ReentrantLock`, crie uma exclusão mútua que garanta a atomicidade das operações de consulta e depósito.