Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores
Programação Concorrente, Inverno de 2017/2018

Série de Exercícios 1

1. Implemente em Java a classe **`ExpirableLazy<T>`** com a seguinte interface pública:

```
public class ExpirableLazy<T>   {
  public ExpirableLazy(Supplier<T> provider, long timeToLive);
  public T get()   throws ExecutionException, ThreadInterruptedException
}
```

Esta classe implementa um produtor *lazy* e *thread-safe* de valores a partir do `provider` passado na construção, com limitação no tempo de vida de cada valor produzido especificada através do parâmetro `timeToLive`.
O método `get()` deve ter o seguinte comportamento: (a) caso o valor já tenha sido calculado e o seu tempo de vida ainda não tenha expirado, retorna esse valor; (b) caso o valor ainda não tenha sido calculado ou o tempo de vida já tenha sido ultrapassado, inicia o cálculo chamando `provider` na própria *thread* invocante e retorna o valor resultante; (c) caso já exista outra *thread* a realizar esse cálculo, espera até que o valor esteja calculado; (d) lança `ThreadInterruptedException` se a espera da *thread* for interrompida.
Caso a chamada a `provider` resulte numa excepção: (a) a chamada a `get()` nessa *thread* deve resultar no lançamento da exceção `ExecutionException` que embrulhe a excepção ocorrida; (b) se existirem outras *threads* à espera do valor, deve ser seleccionada uma delas para a retentativa do cálculo através da função `provider`. Não existe limite no número de retentativas. O tempo de vida inicia-se quando um novo valor é produzido pela função `provider`.