Pràctica 2. `Introducció als Threads`: Sender, Receiver i Channel
Laboratori d'Aplicacions i Serveis Telemàtics

Josep Cotrina, Marcel Fernàndez, Jordi Forga, Juan Luis Gorricho, Francesc Oller

Introducció

Espera activa amb `sleep`

Modifiqueu el codi de la pràctica anterior creant dos threads: `Sender` i `Receiver`. El `Sender` llegeix de teclat fins EOF, emet pel `Channel` i tanca l'emissió. El `Receiver` llegeix del `Channel` fins EOF i escriu per pantalla.

En aquesta pràctica hem de suposar que tan sols sabem programar threads però no podem aplicar cap mecanisme de sincronització propi de la concurrència. Si un thread s'ha d'esperar ho haurà de fer amb espera activa.

L'estil de programació dels threads de protocols serà extendre de `Runnable` més que no pas de `Thread`—és més flexible—.

S'ha de programar una classe `BusyWaitChannel` que implementi `Channel`. Al ser la cua de capacitat limitada el mètode `send` sols ha de fer `put` quan la cua no sigui plena i el mètode `receive` sols ha de fer `get` quan la cua no sigui buida.

Podeu reaprofitar el codi que calgui de la pràctica anterior.

A l'hora de provar-ho convé que l'input associat al teclat sigui suficientment gran. Una idea seria redirigir l'entrada standard des de un arxiu i la sortida standard a un arxiu. Exemple—des de la consola—:

```
java -cp build/classes/:lib/ast-protocols-1.3.1.jar ast.practica1.Main <arxiu_input
    >arxiu_output
```

Una altre seria simplement preeditar un conjunt de segments a transmetre en el codi. Pregunteu al vostre professor de pràctiques quina opció realitzar.

Quin és l'atribut de la cua que es modifica concurrentment pels dos threads? Per tal que l'execució no es bloquegi, cóm s'ha de declarar aquest atribut?—pregunta al teu professor de pràctiques— Per tal de garantir que l'execució falla substituïu l'increment o decrement del comptador de segments—`num`— de la cua per:

```
    tmp = num; sleep; tmp = tmp ± 1; sleep; num = tmp
```

Funciona?

Espera activa amb `await`

Programeu ara una implementació de `Channel` anomenada `AwaitChannel` fent servir la construcció `await` vista a classe per implementar exclusió mútua i espera basada en condicions. Mantingueu els sleeps per assegurar que es produeixen canvis de contexte. Ara hauria de funcionar.

La programació d'una implementació de `Channel` en quant als aspectes de sincronització i concurrència sense espera activa serà l'objectiu de la següent pràctica. Programareu una implementació de `Channel` tant amb monitors natius de JAVA com generals.