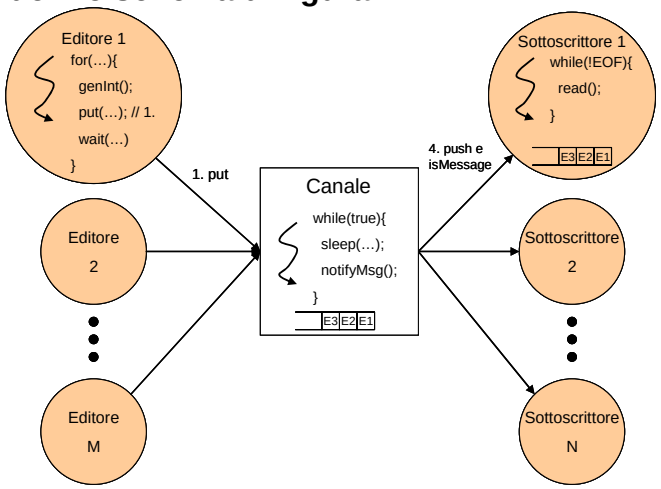
# 1° Esercitazione (proposta)
## Multithreading in Java
## Si consideri lo schema di figura…




1. **Modificare l'Editore per consentire editor multipli**: si consiglia di aggiungere un campo intero da utilizzare come identificativo, come accade già per i Sottoscrittori.
   Eliminare inoltre la generazione con input da Console sostituendola con una generazione automatica di stringhe. Si eseguano cioè N_PUBBLICAZIONI, ad esempio 7: ciascuna pubblicazione (stringa pubblicata) contiene l'identificatore dell'editore e il numero dell'iterazione corrente, ad esempio: "E1, pubblicazione 1".
   Il tempo tra una generazione e l'altra viene fissato in modo casuale fra 2 e 7 secondi. Modificare poi il programma di test vedendo cosa accade nel caso di più Editori che utilizzino tutti lo stesso Canale. Quali problemi possono verificarsi in questo caso? Il funzionamento è ancora corretto o sono necessari ulteriori accorgimenti per evitare deadlock e starvation?



2. **Modificare l'architettura del Canale** in modo da disaccoppiare il momento di generazione dell'evento dalla notifica.
   Il funzionamento finale che si vuole ottenere è il seguente: gli Editori pubblicano gli eventi generati in modo asincrono presso il Canale ed il Canale attivo, ogni 2 secondi, notifica a tutti i Sottoscrittori registrati gli eventi ricevuti dagli Editori dall'ultimo istante di notifica (cioè 2 sec. prima).

*Si noti che per ottenere il comportamento desiderato bisognerà modificare il Canale in modo da renderlo un processo e aggiungere una coda degli eventi.*
In particolare:

- Il **Canale** è un thread che deve realizzare il comportamento descritto al punto 2. Si realizzi la coda degli eventi come una struttura dati di tipo *Array* di MAX_EVENTS elementi. Contestualmente agli aggiornamenti di tale struttura dati sarà necessario aggiornare anche un *contatore* che ne indichi il grado di riempimento.
  **NB:** *la struttura dati contenenti gli eventi ha dimensione finita*. Si vuole perciò realizzare il metodo put in modo che sia sospensivo in caso di coda degli eventi del Canale piena. In tal caso, perciò i thread degli Editori verranno sospesi fino a quando c'è spazio nella coda.
- L'**Editore** è lo stesso descritto al punto 1. Si noti che rispetto all'Editore progettato nell'esercitazione svolta questo Editore *non* pubblica l'EOF, essendo l'interazione da console sostituita con una generazione automatica.
- Il **Sottoscrittore** rimane invariato, stampa cioè semplicemente a video un certo numero di eventi, ad esempio MAX_EVENTS, poi termina.

## Proposta di estensione

Si estenda il programma sviluppato in modo che gestisca la generazione e la distruzione di tutti i processi messi in esecuzione (Canale compreso); il risultato che si vuole ottenere è il seguente.

**Attivazione**: gli Editori non devono effettuare pubblicazioni prima che il Canale sia stato attivato; in altre parole, gli Editori vengono sospesi fino a quando il Canale è stato attivato. In modo simile, il Canale non effettua notifiche prima che almeno un Sottoscrittore si sia registrato.

**Terminazione**: quando tutti i Sottoscrittori terminano l'esecuzione e rimangono attivi solo degli Editori, tali Editori devono terminare senza effettuare ulteriori pubblicazioni e, una volta terminati tutti gli Editori, anche il Canale termina; in modo duale si vuole gestire la terminazione nel caso in cui gli Editori terminino prima dei Sottoscrittori.

## Suggerimento

Si suggerisce di istanziare, all'interno del main del programma, una struttura dati che mantenga il numero di Editori e Sottoscrittori attivi. Tale struttura dati, nota agli Editori, ai Sottoscrittori e al Canale, verrà utilizzata dagli stessi per gestire in modo sincronizzato la terminazione di tutti i processi.

## Consegna

*Chi vuole può inviare via email lo svolgimento dell'estensione ai docenti, in modo da discutere la soluzione ed eventualmente pubblicarla sul sito del corso*