Ingegneria del Software

Esercitazione 5

Conto corrente

Tre persone hanno un fondo comune. Una persona, il produttore, ha il compito di depositare i soldi per tutti alla fine del mese ma non può spenderli mentre le altre due, i consumatori, posso prelevare. Il conto non può andare in rosso.

Si implementino in Java 3 processi che simulino il fondo comune e gli accessi. Il produttore deposita max 200€ ogni 5 secondi, mentre il primo consumatore può prelevare max 300€ ogni secondo, mentre l'altro max 200€ ogni tre secondi.

Pentola

- Dei campeggiatori mangiano servendosi da una pentola comune.
- La pentola può contenere P porzioni di cibo (con P non necessariamente maggiore del numero di campeggiatori). Ogni campeggiatore mangia una porzione per volta. Quando la pentola si svuota (e solo allora), il cuoco provvede a riempirla con nuove P porzioni.
- Implementare in Java per le sole parti relative alla sincronizzazione tra i processi, i programmi che realizzano i comportamenti dei campeggiatori e del cuoco e la gestione della pentola.

I/O Buffer

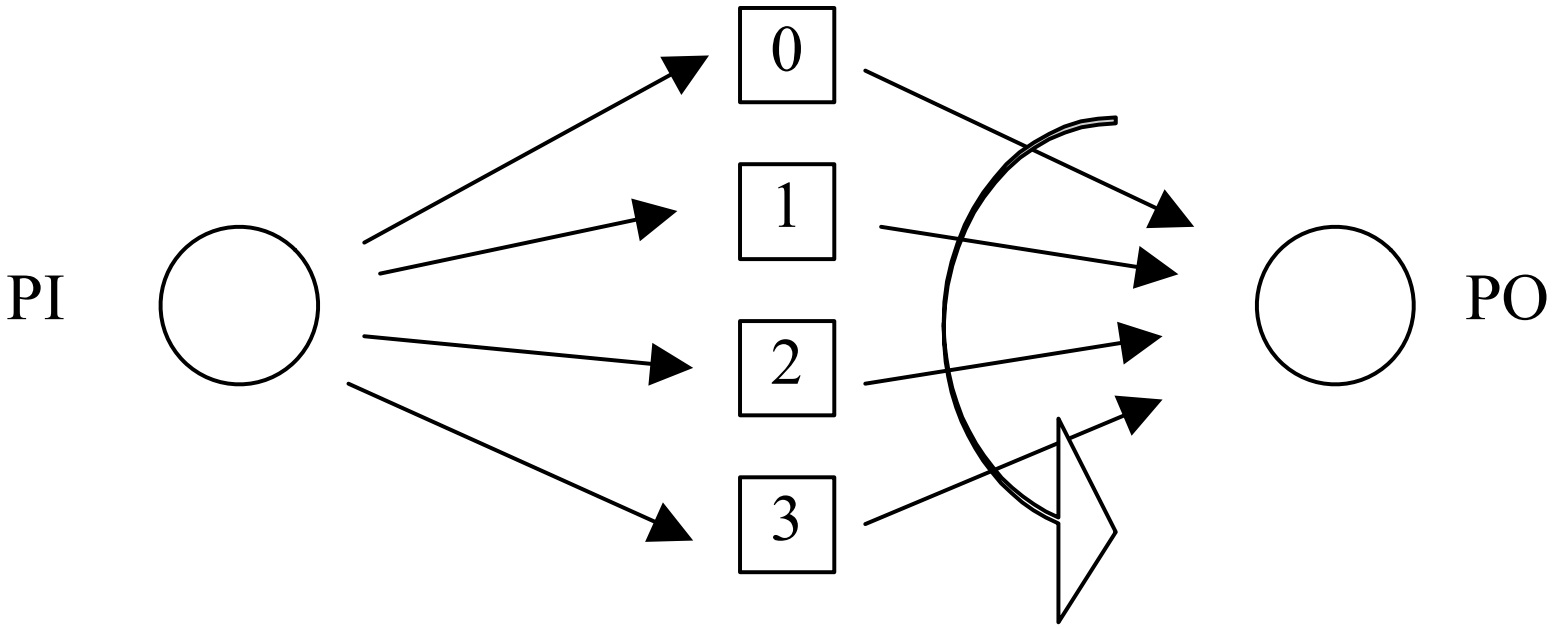
Dato un sistema con quattro buffer di caratteri.

Il modulo PI esegue ripetutamente le seguenti operazioni: legge da tastiera una coppia di valori <i, ch>, dove *i* è un numero tra 0 e 3, *ch* un carattere, e inserisce il carattere *ch* nel buffer *i* (ognuno dei quattro buffer contiene al più un carattere).

Il modulo PO considera a turno in modo circolare i quattro buffer e preleva il carattere in esso contenuto, scrivendo in uscita la coppia di valori <i, ch> se ha appena prelevato il carattere *ch* dal buffer *i*.

L'accesso a ognuno dei buffer è in mutua esclusione; PI rimane bloccato se il buffer a cui accede è pieno, PO se è vuoto.

I/O Buffer



Parte 1

- Data la seguente sequenza di valori letta da PI, scrivere la sequenza scritta in corrispondenza da PO.

  <1, c> <0, b> <2, m> <0, f> <1, h> <3, n>

Soluzione 1

<0, b> <1, c> <2, m> <3, n> <0, f> <1, h>

Parte 2

- Descrivere brevemente in quali casi si può verificare una situazione di *deadlock* tra PI e PO. Illustrare con un semplice esempio.

Soluzione 2

- Deadlock: <1,a> <1,b>

Parte 3

- Implementare il sistema in Java

Un impianto con valvola

Un impianto può portarsi dallo stato di funzionamento normale in uno stato di gestione di malfunzionamento (probabilità 20%). Entrato in tale stato, deve essere aperta una valvola di scarico (in un altro thread). La valvola si può aprire in 0 e 8 secondi. Se non si apre entro 5 secondi, l'impianto passa ad uno stato di fermo. Altrimenti, se la valvola viene aperta nel tempo stabilito, essa rimane in tale stato per un tempo non inferiore a 5s e non superiore a 10s e solo dopo aver atteso lo sfogo della valvola l’impianto ritorna nello stato normale.

Networking

SocSeq

Si realizzi in Java, utilizzando le librerie Thread e Socket un server di rete che gestisce un gioco elettronico distribuito, a cui possono giocare più utenti contemporaneamente.

Il gioco consiste nel leggere rapidamente una sequenza di numeri e nel riscriverla correttamente.

Una volta lanciato il client chiede all'utente di inserire username e password e le invia al server dopo aver inviato il messaggio "login". Se l'utente inserisce un username mai usato da nessuno nel sistema, il server lo registra associandolo con la password inserita. Se invece l'utente inserisce uno username già registrato il server verifica che la password ricevuta sia corretta. La procedura viene ripetuta fino a che il processo di login termina correttamente. Se il processo va a buon fine il server invia la stringa "true", altrimenti invia la stringa "false".

SocSeq (II)

Una volta loggato l'utente può decidere di iniziare una partita o di richiedere la lista dei record personali di tutti gli utenti.

Nel primo caso il client invia la stringa "partita". Il server invia al client la prima sequenza da ricordare. Il client la mostra per qualche istante all'utente, poi chiede all'utente di ridigitarla, e la invia al server. Il server controlla che essa sia uguale a quella inviata, se essa non è corretta invia la stringa "false" e termina la partita, altrimenti risponde con la stringa "true" e invia un'altra sequenza di numeri.

Se invece l'utente sceglie di vedere i record dei giocatori, il client invia al server la stringa "record" e questi risponde con una stringa rappresentante il numero dei giocatori seguita da una stringa per ognuno dei giocatori contenente i suoi record.

Quando il giocatore sceglie di uscire dal programma, il client invia al server la stringa "fine", e il server chiude la connessione col client.

Chat RMI

Implementare una Chat distribuita in RMI. Il client deve inserire un nome utente e l'indirizzo del server (nel nostro caso *localhost*). Restituisce quindi gli username degli utenti in chat e la conversazione può iniziare. Ogni messaggio è inviato in broadcast a tutti i partecipanti.