Esercizio 1. Concorrenza

Si consideri il codice dato, che gestisce un rudimentale servizio di posta. Nel sistema dato i client agiscono in modo concorrente, come thread di un unico programma, di cui fa parte anche la casella postale.

Si chiede di modificare il codice dato come segue:

1) evitare corse critiche e altre possibili problematiche (deadlock, starvation, ecc.) tipiche della programmazione concorrente;
2) modificare l'operazione di lettura della posta, in modo che un client si sospenda in attesa della posta per un tempo massimo T. La lettura termina quando arriva un messaggio per il client o quando il tempo T è trascorso senza che sia arrivato alcun messaggio. L'operazione deve terminare restituendo un valore nullo quando non è stato possibile leggere alcun messaggio per il client entro il termine dato.

Si ricorda che bisogna caricare il file .java (NON i .class) in un unico file zip.

Esercizio 2. Programmazione distribuita con socket

Si modifichi il codice realizzato come soluzione dell'esercizio di programmazione concorrente, in modo da realizzare un sistema distribuito avente le seguenti caratteristiche:

1) Un server gestisce le caselle di posta.
2) I client si comportano come nell'esercizio di programmazione concorrente.

Si realizzi il sistema usando i socket.

Come sempre, occorre evitare i problemi tipici della programmazione concorrente e distribuita (corse, critiche, deadlock, starvation, ecc.).

Si ricorda che bisogna caricare il file .java (NON i .class) in un unico file zip.

Esercizio 3. Programmazione distribuita con RMI

Si modifichi il codice realizzato come soluzione dell'esercizio di programmazione concorrente, in modo da realizzare un sistema distribuito in cui un server gestisce le caselle di posta. Il sistema ha le seguenti caratteristiche:

1) i client possono spedire messaggi come nell'esercizio di programmazione concorrente.
2) i client non effettuano operazioni di lettura: ogni volta che il server riceve un messaggio destinato loro, ricevono una notifica contenente il messaggio.

Si realizzi il sistema usando RMI.

Come sempre, occorre evitare i problemi tipici della programmazione concorrente e distribuita (corse, critiche, deadlock, starvation, ecc.).

Si ricorda che bisogna caricare il file .java (NON i .class) in un unico file zip.