Università di Padova – Corso di Laurea triennale in Ingegneria Informatica
Sistemi Operativi – A.A. 2004/2005

Tema tipo per la prova scritta

Si voglia gestire il traffico aereo in un aeroporto in cui si presentano ripetutamente le due situazioni tipiche: aerei che devono decollare ed aerei che devono atterrare. La pista (rappresentata in figura) è suddivisa in due zone, zona A, impegnata per iniziare un decollo e come punto iniziale di atterraggio, e zona B, usata per il rullaggio dell'aereo (sia in decollo che atterraggio).

Le azioni dei piloti sono sincronizzate dalla torre di controllo. In particolare sono state formalizzate due sequenze distinte di azioni per le due situazioni.

- DECOLLO

I piloti si accodano sul canale d'ingresso della pista. La zona A può essere impegnata al massimo da due aerei contemporaneamente. Per impegnare la zona A il pilota invoca il metodo `richAccessoPista` della torre di controllo. Successivamente, quando ha impegnato la zona A, segnala alla torre di controllo l'intenzione di iniziare il decollo mediante il metodo `richAutorizDecollo` impegnando se possibile, la zona B e liberando la zona A o restando in attesa altrimenti. Inizia quindi il rullaggio per il decollo sulla zona B, che può essere impegnata al massimo da due aerei per decolli contemporanei. Quando il pilota è in volo annuncia di aver liberato la zona B mediante una chiamata al metodo `inVolo`.
Se si considera `tc` un'istanza della classe `TorreDiControllo<`*`Metodo`*`>`, dove *`Metodo`* identifica la tecnica utilizzata (`Mon` = Monitor di Hoare, `MJ` = Monitor di Java, `SemP` = semafori privati, `Reg` = regione critica), e `io` un indice che identifica l'aereo, il codice per il decollo è il seguente:

```
//il pilota è pronto per entrare in pista
tc.richAccessoPista(io);
//il pilota entra nella zona A
tc.richAutorizDecollo(io);
//il pilota impegna la zona B e libera la A
tc.inVolo(io);
//il pilota è in volo e la zona B è libera
```

- ATTERRAGGIO

I piloti sono in volo e si avvicinano alla pista che, per motivi di sicurezza, deve essere **completamente** libera per consentire un atterraggio. Per assicurarsi di questo i piloti chiedono l'autorizzazione all'atterraggio (e quindi la disponibilità della pista) invocando il metodo `richAutorizAtterraggio` della torre di controllo. Un aereo che sta per atterrare viene messo in attesa nel caso la pista o parte di essa risultasse impegnata. Una volta ottenuta l'autorizzazione, l'aereo si allinea con la pista e atterra occupando la zona A. Quando passa nella zona B, liberando la A, comincia a frenare e avvisa la torre di controllo mediante la chiamata al metodo `freniAttivati`. La procedura di atterraggio termina con il disimpegno della pista che viene attuato mediante la chiamata al metodo `inParcheggio` della torre di controllo.
Nelle condizioni viste per il decollo, il codice per l'atterraggio è il seguente::

```
//il pilota è in volo e deve atterrare
tc.richAutorizAtterraggio(io);
//il pilota atterra occupando la zona A
tc.freniAttivati(io);
//il pilota impegna la zona B e libera la A
tc.inParcheggio(io);
//il pilota esce dalla pista e libera B
```

Si gestisca il fatto che gli atterraggi hanno priorità sui decolli. In particolare, quando viene invocato il metodo della torre di controllo `richAccessoPista`, la torre verificherà se vi sia già una richiesta di atterraggio pendente: in caso affermativo il processo in fase di decollo verrà sospeso. Tale fase sarà eventualmente ripresa ad atterraggio concluso (sempre che non ci sono altri aerei in attesa di atterrare).

Sulla base di quando descritto è richiesto di realizzare:

1. la rete di Petri che descrive il funzionamento dell'aereoporto (dovranno essere evidenziati i processi che rappresentano gli aerei che decollano, quelli degli aerei che atterrano e la sincronizzazione imposta dalle regole della torre di controllo);
2. la classe astratta `TorreDiControllo`, che dichiara tutte le variabili necessarie a rappresentare lo stato dell'aeroporto, visualizzato dal metodo `stampaSituazioneAeroporto`, egualmente in `TorreDiControllo`;

3. la classe `AereoCheDecolla` che, ricevendo in fase di costruzione il riferimento all'istanza di tipo `TorreDiControllo`, che implementa i metodi di sincronizzazione, e un indice che identifica i singoli aerei, rappresenti un *thread* che esegue la descritta procedura prevista per un aereo che decolla;
4. la classe `AereoCheAtterra` che, similmente al punto precedente, rappresenti un *thread* che esegue la descritta procedura prevista per un aereo che atterra;
5. una classe `TorreDiControllo<`*`Metodo`*`>` per ciascuno degli approcci menzionati (`Mon` = Monitor di Hoare, `MJ` = Monitor di Java, `SemP` = semafori privati, `Reg` = regione critica) che implementi tutti i metodi di controllo;
6. una classe test `Aeroporto<`*`Metodo`*`>` che 'crea' e attiva un certo numero di aerei di ciascuno dei due tipi e utilizza la torre di controllo corrispondente a *`Metodo`*;

Se possibile, si inseriscano nel codice, in opportuni punti, attese a tempo per rendere più efficace la simulazione.

7. Successivamente si proponga una soluzione in ADA che segua le medesime specifiche.