Sistemi Operativi T
Prova di laboratorio
15 Luglio 2024

Esercizio di Programmazione Concorrente in Java

In una sagra di paese è presente un **sistema per l'erogazione di birra "alla spina"**.
L'erogazione avviene mediante **1 erogatore**.
L'erogatore ha **un proprio serbatoio**, costituito da un **fusto** pre-confezionato di birra pronta per l'erogazione. Ogni fusto contiene inizialmente **Max** litri di prodotto.
Il sistema è a disposizione dei **clienti**, che, secondo la formula *"all-you-can-drink"*, possono autonomamente prelevare quantità prefissate di birra, corrispondenti ai soli due formati di bicchiere previsti:

- **Formato piccolo** da 0,33 litri
- **Formato grande**, da 0,5 litri.

L' erogatore può essere utilizzato da un cliente alla volta per erogare la quantità di birra richiesta (formato grande o piccolo). Si assuma che ogni erogazione richieda un tempo non trascurabile.
Per ogni cliente l'erogazione può avvenire soltanto se la quantità di birra richiesta (0,33 l o 0,5 l) all'interno del fusto è disponibile; altrimenti il cliente attende la sostituzione del **fusto esaurito** con un fusto pieno da parte di un **addetto**.

L'**addetto** interviene ogni volta che è necessario sostituire **il fusto esaurito**.
Durante ogni intervento il sistema è **inutilizzabile** da parte dei clienti.
Pertanto, prima di iniziare ogni intervento, l'addetto attende che l'erogatore sia libero (cioè, non utilizzato da clienti); analogamente, ogni cliente potrà utilizzare l'erogatore quando non c'è un intervento in corso.
Si assuma che la fase di sostituzione del fusto richieda un tempo non trascurabile.

Comportamento del cliente: ogni cliente ciclicamente ripete per un numero arbitrario di volte:

1. <decide il formato del bicchiere F>
2. **Inizia spillatura** di birra per riempire un bicchiere di formato F
3. <riempimento bicchiere>
4. **Termina spillatura**.
5. <beve>

Sistemi Operativi T
Prova di laboratorio
15 Luglio 2024

Comportamento dell'addetto: l'addetto ciclicamente ripete:

1. **Inizia intervento di sostituzione** del fusto;
2. <sostituzione >
3. **Termina intervento**.

Si realizzi un'applicazione concorrente in **Java** che rappresenti **l'addetto** e i **clienti** tramite thread concorrenti e che, utilizzando monitor e variabili condizione, controlli gli accessi dei clienti e dell'addetto al sistema in modo da soddisfare i vincoli dati, ed inoltre il seguente **vincolo di priorità**:

- i clienti che vogliono riempire il bicchiere piccolo abbiano la precedenza sui clienti che vogliono riempire il bicchiere grande**.**

Sistemi Operativi T
Prova di laboratorio 26 giugno 2024
TEMA A

L'amministrazione comunale di una città ha organizzato una mostra dedicata ai grandi fotografi che hanno raccontato la città con le proprie foto.
L'evento si svolge in una sala della sede del Comune ed è aperto al pubblico.

Tuttavia, la sala utilizzata è anche utilizzata per le **riunioni** dei dipendenti del Comune. In particolare, quando si svolge una riunione non è ammessa la presenza di visitatori della mostra; inoltre, una riunione non può iniziare fino a che vi sono visitatori della mostra dentro la sala.
Per ogni riunione è previsto che vi sia un particolare dipendente, **l'usciere**, che apre e chiude ogni riunione.

Pertanto, per ogni riunione, il **comportamento dell'usciere** sarà il seguente.

- **Entra nella sala vuota** per iniziare la riunione

<vi rimane per il tempo necessario allo svolgimento della riunione>

- **Esce dalla sala vuota** per terminare la riunione;

La presenza in sala dell'usciere significa quindi che c'è una riunione in atto; quindi ogni dipendente potrà partecipare alla prossima riunione, solo se l'usciere è già presente in sala.

Pertanto, per ogni riunione, **il dipendente avrà il seguente comportamento**:

- **Entra nella sala** quando l'usciere è presente;

<partecipa alla riunione>

- **Esce dalla sala** quando l'usciere è ancora presente;

Quando non ci sono riunioni in atto, la sala è utilizzata come mostra. In questo caso è accessibile solo da **visitatori esterni** (ovvero, né i dipendenti né l'usciere possono entrare, se c'è almeno un visitatore all'interno).
Pertanto, il **comportamento di ogni visitatore** sarà il seguente:

- **Entra nella sala** quando l'usciere non è presente;

<visita la mostra>

- **Esce dalla sala**;

La sala ha una **capacità limitata a MAX persone** (di qualunque tipo: visitatori, dipendenti e usciere) oltre le quali nessuna persona può entrare.

Realizzare un'applicazione concorrente basata sul monitor in linguaggio **Java**, nella quale usciere, dipendenti e visitatori siano rappresentati da thread concorrenti.
La sincronizzazione tra i thread dovrà tenere conto di tutti i vincoli dati, ed inoltre delle seguenti regole di priorità nell'accesso alla sala:

- l'usciere ha la priorità sui dipendenti
- i dipendenti hanno la priorità sui visitatori

Sistemi Operativi T
Prova di laboratorio 26 giugno 2024
TEMA A

L'amministrazione comunale di una città ha organizzato una mostra dedicata ai grandi fotografi che hanno raccontato la città con le proprie foto.
L'evento si svolge in una sala della sede del Comune ed è aperto al pubblico.

Tuttavia, la sala utilizzata è anche utilizzata per le **riunioni** dei dipendenti del Comune. In particolare, quando si svolge una riunione non è ammessa la presenza di visitatori della mostra; inoltre, una riunione non può iniziare fino a che vi sono visitatori della mostra dentro la sala.
Per ogni riunione è previsto che vi sia un particolare dipendente, **l'usciere**, che apre e chiude ogni riunione.

Pertanto, per ogni riunione, il **comportamento dell'usciere** sarà il seguente.

- **Entra nella sala vuota** per iniziare la riunione

<vi rimane per il tempo necessario allo svolgimento della riunione>

- **Esce dalla sala vuota** per terminare la riunione;

La presenza in sala dell'usciere significa quindi che c'è una riunione in atto; quindi ogni dipendente potrà partecipare alla prossima riunione, solo se l'usciere è già presente in sala.

Pertanto, per ogni riunione, **il dipendente avrà il seguente comportamento**:

- **Entra nella sala** quando l'usciere è presente;

<partecipa alla riunione>

- **Esce dalla sala** quando l'usciere è ancora presente;

Quando non ci sono riunioni in atto, la sala è utilizzata come mostra. In questo caso è accessibile solo da **visitatori esterni** (ovvero, né i dipendenti né l'usciere possono entrare, se c'è almeno un visitatore all'interno).
Pertanto, il **comportamento di ogni visitatore** sarà il seguente:

- **Entra nella sala** quando l'usciere non è presente;

<visita la mostra>

- **Esce dalla sala**;

La sala ha una **capacità limitata a MAX persone** (di qualunque tipo: visitatori, dipendenti e usciere) oltre le quali nessuna persona può entrare.

Realizzare un'applicazione concorrente basata sul monitor in linguaggio **Java**, nella quale usciere, dipendenti e visitatori siano rappresentati da thread concorrenti.
La sincronizzazione tra i thread dovrà tenere conto di tutti i vincoli dati, ed inoltre delle seguenti regole di priorità nell'accesso alla sala:

- l'usciere ha la priorità sui dipendenti
- i dipendenti hanno la priorità sui visitatori

Sistemi Operativi T
Prova di laboratorio
14 giugno 2023
Tema A

Esercizio di Programmazione Concorrente in Java (punti 10)

In seguito ad un evento alluvionale, il Palazzetto dello sport di una cittadina viene utilizzato per la raccolta e la distribuzione di beni di prima necessità alle famiglie di cittadini colpite.
I beni distribuiti sono donati da aziende e privati ad associazioni benefiche (ONLUS); tali associazioni confezionano i beni raccolti in pacchi pronti per la distribuzione e li depositano nel palazzetto per la distribuzione. In particolare, i pacchi possono essere di 2 formati:

- **Singolo**, pacco contenente il necessario per una persona;
- **Famiglia,** pacco contenente i beni necessari per una famiglia.

Ai fini della distribuzione, tutti i pacchi di uno stesso formato (singolo o famiglia) sono considerati equivalenti.
Al Palazzetto possono accedere 2 tipi di utenti:

- **Cittadini**: ogni cittadino accede al palazzetto per **prelevare 1 pacco di un dato formato (singolo o famiglia)**.
- **Fornitori,** ovvero gli addetti di associazioni ONLUS che ciclicamente accedono al palazzetto **per depositare un pacco di un dato formato** (**singolo o famiglia**) alla volta.

Nella struttura vi è, pertanto, un deposito nel quale vengono man mano depositati i pacchi dai fornitori e dal quale vengono prelevati per la consegna ai cittadini.
Per motivi di spazio è fissato un limite **Pmax** al numero di pacchi "**singoli**" che possono essere accumulati all'interno del deposito; a questo proposito si assuma che un **pacco famiglia occupi lo spazio di 3 pacchi singoli**.
La struttura è presidiata da un **addetto** che, in caso di **allerta meteo**, procede immediatamente alla chiusura del palazzetto interrompendo i servizi di raccolta e distribuzione; quando la situazione di allerta sarà finita, l'addetto riaprirà il palazzetto per consentire la ripresa delle attività di raccolta e distribuzione. Per semplicità si assuma che gli istanti di chiusura e di riapertura vengano definiti in modo casuale.

Comportamento del Cittadino:

- <arriva al palazzetto>
- **Preleva** un pacco di formato arbitrario (singolo o famiglia)
- <se ne va>

Comportamento del Fornitore:

in modo ciclico:

- <arriva al palazzetto>
- **Deposita** un pacco di formato arbitrario (singolo o famiglia)
- <esce>

Comportamento dell'Addetto:

in modo ciclico:

- <monitora la situazione>
- **Chiude** il palazzetto in caso di allerta meteo
- < interruzione dei servizi dovuta all'allerta>
- **Apre** il palazzetto

Realizzare un'applicazione concorrente in Java basata sul monitor nella quale **Cittadini**, **Fornitori** e **Addetto** siano rappresentati da **thread distinti**.
La politica di sincronizzazione dovrà tenere in considerazione tutti i vincoli dati, ed inoltre:

- nel **deposito di pacchi**: i **fornitori** di pacchi **singoli devono avere la precedenza sui** fornitori di pacchi **famiglia**;
- nel **prelievo di pacchi**: i cittadini che chiedono **pacchi formato famiglia devono avere la precedenza sui** cittadini che chiedono **pacchi singoli**.

Sistemi Operativi T
Prova di laboratorio
11 luglio 2022

1. Esercizio di Programmazione Concorrente in Java

In una località turistica di mare è presente un porto turistico presso il quale possono attraccare imbarcazioni da diporto e motovedette della capitaneria di porto.
In particolare ogni imbarcazione appartiene a una delle categorie seguenti:

- Motovedette della Capitaneria di porto,
- Imbarcazioni da diporto piccole (es. motoscafi, gommoni, ecc.),
- Imbarcazioni da diporto grandi (es. motoryacht, barche a vela cabinate, cc.).

Alle imbarcazioni da diporto, il porto offre:

- N posti barca "standard" per l'attracco di imbarcazioni piccole;
- M posti barca "maxi" per l'attracco di imbarcazioni grandi.

Ogni imbarcazione piccola può occupare sia un posto barca standard, che un posto barca maxi.
Ogni imbarcazione grande può occupare solo in un posto barca maxi.
Ogni motovedetta ha invece il suo posto riservato, pertanto non può accadere che all'arrivo in porto di una motovedetta non ci sia posto per attraccare.

Il porto è accessibile dal mare tramite un unico **canale** di larghezza limitata, utilizzato da tutte le imbarcazioni per entrare e uscire dal porto.
Le **regole di navigazione** del canale sono le seguenti:

- La presenza di una motovedetta M in direzione D nel canale impedisce ad ogni altra imbarcazione di percorrere il canale in qualunque direzione (in questo caso, il canale sarà quindi percorso da una sola motovedetta alla volta.
- La presenza di una imbarcazione grande nel canale in direzione D impedisce a qualunque altra imbarcazione in direzione opposta a D di percorrere il canale (in questo caso, quindi, il canale è utilizzato a senso unico);
- Infine, se nel canale vi sono soltanto imbarcazioni piccole, esso potrà essere utilizzato in entrambi i versi di navigazione.

Comportamento Motovedette:
Ogni motovedetta ciclicamente salpa dal porto per una ricognizione in mare; pertanto il suo comportamento sarà il seguente:

ciclicamente ripete:

1. **accede al canale in direzione OUT**
2. percorre il canale in direzione OUT impiegando un tempo arbitrario
3. **esce dal canale in direzione OUT**
4. effettua una ricognizione in mare impiegando un tempo arbitrario
5. **accede al canale in direzione IN**
6. percorre il canale in direzione IN impiegando un tempo arbitrario
7. **esce dal canale in direzione IN**
8. rimane nel porto per un tempo arbitrario

Comportamento Imbarcazioni da Diporto:
Ogni imbarcazione (piccola o grande) arriva alla località per attraccare nel porto. Pertanto:

1. **accede al canale in direzione IN**
2. percorre il canale in direzione IN impiegando un tempo arbitrario
3. **esce dal canale in direzione IN occupando un posto barca**
4. rimane nel porto per un tempo arbitrario
5. **accede al canale in direzione OUT liberando il posto barca**
6. percorre il canale in direzione OUT impiegando un tempo arbitrario
7. **esce dal canale in direzione OUT**
8. naviga in mare aperto

Realizzare un'applicazione concorrente in Java basata sul monitor nella quale Imbarcazioni da Diporto (piccole e grandi) e le Motovedette siano rappresentate da thread concorrenti.
La sincronizzazione tra i thread dovrà tenere conto di tutti i vincoli dati ed inoltre dei seguenti criteri di **priorità nell'accesso al canale**:

- le **motovedette abbiano la precedenza sulle imbarcazioni** da diporto; in particolare, le **motovedette in uscita abbiano la priorità sulle motovedette in entrata**.
- Tra le imbarcazioni da diporto: le **imbarcazioni in uscita abbiano la precedenza sulle imbarcazioni in entrata**; nell'ambito della stessa direzione, le **imbarcazioni grandi abbiano la precedenza su quelle piccole**.

Sistemi Operativi T
Prova di laboratorio
29 giugno 2022

Esercizio di Programmazione Concorrente in Java [punti 10]

Si consideri un sito speleologico, costituito da una grande grotta aperta alle visite del pubblico.
La grotta può essere visitata da due tipi di utenti:
• **adulti**;
• **bambini**.
Ogni visitatore (adulto o bambino) si comporta come segue:

1. **Entra** nella grotta
2. Visita la grotta impiegando un tempo arbitrario
3. **Esce** dalla grotta

Il regolamento del sito speleologico prevede che sia necessaria la presenza di **guide speleologiche** all'interno della grotta per supervisionare le visite ed eventualmente intervenire in caso di problemi. In particolare, detto **NG** il numero di guide nella grotta e **NV** il numero di visitatori nella grotta, deve essere sempre rispettata la relazione:

NG >= NV/5 (*)

Ogni guida può entrare e uscire dalla grotta, purchè la relazione (*) sia sempre rispettata.
Ogni guida avrà quindi un comportamento ripetitivo; ad ogni ciclo attraverserà le seguenti fasi:

1. **Entra** nella grotta;
2. Permane nella grotta a supervisionare i visitatori per un tempo arbitrario;
3. **Esce** dalla grotta;
4. Sta all'esterno della grotta per un tempo arbitrario.

La grotta ha una **capacità limitata** MAX che esprime il numero massimo di **visitatori** che contemporaneamente possono stare all'interno della grotta (le guide non vengono conteggiate).

Realizzare un'applicazione concorrente in Java basata sul monitor nella quale Visitatori (adulti e bambini) e Guide siano rappresentati da thread concorrenti.
La sincronizzazione tra i thread dovrà tenere conto di tutti i vincoli dati ed inoltre dei seguenti criteri di priorità:
nell'accesso alla grotta:

- i visitatori adulti abbiano la precedenza sui bambini.

nell'uscita dalla grotta:

- i visitatori abbiano la precedenza sulle guide;
- i visitatori bambini abbiano la precedenza sugli adulti.

Sistemi Operativi T
Prova di laboratorio
25 Giugno 2021

1. Esercizio di Programmazione Concorrente in Java [punti 10]

Si consideri il servizio di **pronto soccorso odontoiatrico** (PSO), a disposizione dei turisti in una località di vacanza.

Gli utenti del servizio possono essere di 2 tipi:

- **Utente adulto**: ha almeno 18 anni e accede **singolarmente** al pronto soccorso;
- **Utente minorenne**: ha meno di 18 anni e accede al pronto soccorso **con 1 familiare, che lo accompagnerà durante l'intera permanenza nel PSO**.

Il servizio è collocato in una sede composta da:

- **NA ambulatori**, ognuno **assegnato permanentemente a un dentista**; ogni utente del servizio verrà visitato ed eventualmente curato in un ambulatorio dal dentista assegnato a quell'ambulatorio. Si considerino tutti gli ambulatori (e tutti i dentisti) equivalenti tra loro.
- **Una sala d'aspetto**, nella quale ogni utente (con l'accompagnatore, se minorenne) attenderà di essere visitato. Per motivi di sicurezza sanitaria la sala d'aspetto ha una capacità massima fissata a MAXS persone (utenti ed eventuali accompagnatori). L'ingresso della sala d'aspetto è presidiato da un operatore sanitario che, **contestualmente all'ingresso di ogni utente** (con l'accompagnatore, se minorenne), effettua un rapido triage, assegnando all'utente entrante un codice di gravità (rosso, giallo o verde).
- **Un'uscita indipendente**, tramite quale gli utenti (con gli accompagnatori, se minorenni), dopo essere stati visitati/curati, lasciano il pronto soccorso.

Ogni utente si comporta come segue:

1. **Entra nella sala d'aspetto** (insieme all'accompagnatore, se minorenne) ricevendo contestualmente il suo codice di gravità.
2. **Richiede l'accesso ad un ambulatorio** (con l'accompagnatore, se minorenne) per essere visitato ed eventualmente curato.
3. <permane nell'ambulatorio per tutto il tempo necessario alla visita e all'eventuale terapia>
4. **Esce dal pronto soccorso** (con l'accompagnatore, se minorenne).

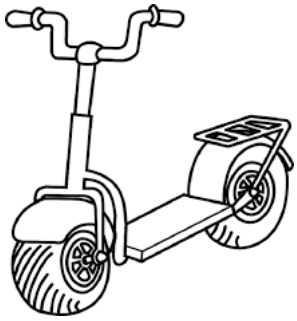
Realizzare **un'applicazione concorrente in Java basata sul monitor** nella quale **ogni utente** sia rappresentato da un **thread distinto** ed il **pronto soccorso** sia una **risorsa** condivisa dagli utenti.
La politica di gestione del pronto soccorso dovrà tenere in considerazione tutti i vincoli dati ed inoltre:

- **Nell'ingresso alla sala d'aspetto** gli utenti adulti avranno la precedenza sui minorenni.
- **Nell'accesso agli ambulatori:** la priorità è ai codici **rossi**, poi ai codici **gialli**, ed infine ai codici **verdi** (indipendentemente dal tipo di utente).

Sistemi Operativi T
Prova di laboratorio
12 Luglio 2019

Tema A

1. Esercizio di Programmazione Concorrente in Java [punti 10]



Si consideri un grande complesso fieristico situato in un'importante città. Il complesso fieristico ospita contemporaneamente 2 manifestazioni:

1. **MotorExpo (MEX)**: un'esposizione di prodotti e tecnologie nel settore *automotive*;
2. **FoodExpo (FEX)**: un'esposizione di prodotti e tecnologie nel settore alimentare.

Entrambe le manifestazioni sono aperte al pubblico e sono collocate in **aree distinte**, tra loro **non comunicanti**. Ogni area è caratterizzata da una **capacità massima** (rispettivamente **MAX_M** e **MAX_F**) che esprime il massimo numero di visitatori contemporaneamente ammissibili all'interno dell'area.
La visita di ogni esposizione è possibile previo acquisto del biglietto di ingresso. L'acquisto dei biglietti avviene unicamente online, e prevede che ogni visitatore acquisti un biglietto "singolo" per visitare la singola esposizione di suo interesse (MEX o FEX); il biglietto vale per un qualunque giorno a un qualunque orario di ingresso per l'esposizione scelta.
All'ingresso del complesso fieristico ogni visitatore presenta il proprio biglietto e (eventualmente dopo una fase di attesa) ottiene l'autorizzazione ad entrare nell'esposizione richiesta.
Data l'estensione delle aree da visitare, in fase di ingresso i visitatori vengono dotati ciascuno di **un monopattino elettrico**. A questo proposito si assuma che il numero dei monopattini complessivamente disponibili sia NMP e che valga la relazione
NMP < MAX_F+MAX_M.
Una volta conclusa la visita, ogni visitatore restituirà il monopattino in fase di uscita.
Nell'accesso all'area si dia la **precedenza ai visitatori dell'area con più posti liberi**.

Realizzare un'applicazione concorrente in Java basata sul monitor **per la gestione dell'area fieristica** che tenga conto di tutti i vincoli dati, nella quale **i visitatori** siano rappresentati da thread concorrenti.

Sistemi Operativi T
Prova di laboratorio
12 Giugno 2020

1. Esercizio di Programmazione Concorrente in Java [punti 10]

Si consideri il Pronto Soccorso di un ospedale.
Il Pronto Soccorso è accessibile agli automezzi attraverso una **rampa di accesso** che consente **sia l'ingresso che l'uscita** dei veicoli per il trasporto dei pazienti.
La rampa conduce a un'area coperta (detta "**camera calda**") dove gli automezzi possono sostare per scaricare i pazienti.
Gli automezzi autorizzati all'utilizzo della rampa si suddividono in 2 categorie:

- **Ambulanze**;
- **Automobili private**, ovvero ogni auto utilizzata per accompagnare un paziente al PS.

La camera calda dispone di **capacità limitata pari a N posti**, dei quali **almeno N/2 devono essere sempre disponibili per le ambulanze**. Pertanto, il numero di auto private che possono sostare nella camera calda non potrà mai eccedere il valore N/2; diversamente, le ambulanze in sosta potranno superare il valore N/2.

Ogni automezzo si comporta come segue:

1. **imbocca** la rampa per **entrare** nella Camera Calda del pronto soccorso;
2. percorre la rampa in direzione di entrata;
3. **esce** dalla rampa per occupare un posto nella Camera Calda;
4. sosta nella camera calda per un tempo arbitrario, necessario allo scarico e all'accompagnamento del paziente nel Pronto Soccorso;
5. **imbocca** la rampa per **uscire** dalla Camera Calda;
6. percorre la rampa in direzione di uscita;
7. **esce** dalla rampa e si allontana dal pronto soccorso.

Ogni ambulanza che vuole accedere al PS può trovarsi in uno dei due stati seguenti:

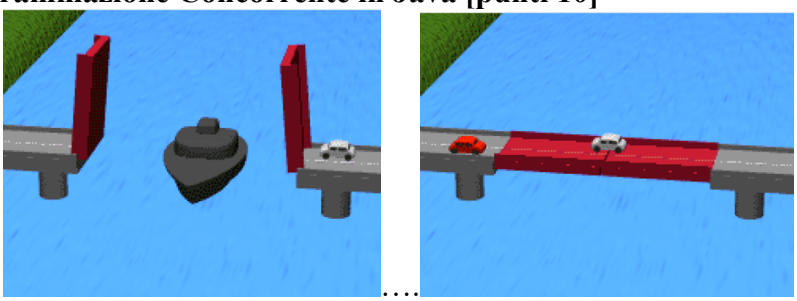
- **Sirena ON**: l'ambulanza sta trasportando un paziente di gravità medio-alta;
- **Sirena OFF**: l'ambulanza sta trasportando un paziente di gravità bassa.

La rampa di ingresso ha una larghezza ridotta, tale per cui non sia permesso il transito contemporaneo nella rampa di ambulanze in direzioni opposte (ingresso e uscita); il transito contemporaneo di auto e ambulanze in direzioni opposte sia invece consentito.
Realizzare un'applicazione concorrente in Java basata sul monitor nella quale **ogni veicolo** sia rappresentato da un **thread distinto**.
La politica di sincronizzazione dovrà tenere in considerazione tutti i vincoli dati ed inoltre:

- dovrà **privilegiare veicoli in uscita dal Pronto Soccorso** rispetto a quelli in entrata.
- **Nell'accesso al Pronto Soccorso**: le ambulanze abbiano la precedenza sulle auto private e, tra le ambulanze, siano favorite le ambulanze con Sirena ON.
- **Nell'uscita dal Pronto Soccorso**: le ambulanze abbiano la priorità sulle auto private.

Sistemi Operativi T
Prova di laboratorio
30 Giugno 2020

1. Esercizio di Programmazione Concorrente in Java [punti 10]



TB **TA**

Si consideri un tratto di un fiume navigabile nel quale le due rive sono collegate mediante un ponte **mobile.**
Il fiume viene percorso da barche di grandi dimensioni; per consentire il passaggio di tali imbarcazioni, il ponte, una volta svuotato dagli autoveicoli, può essere sollevato. In ogni istante il ponte può quindi trovarsi in uno dei due stati seguenti:

- **TA**(transito auto). Il ponte **non è sollevato**: pertanto è **aperto** al transito degli autoveicoli in entrambe le direzioni, e non consente ad alcuna imbarcazione il passaggio nel tratto di fiume considerato;
- **TB** (transito barche). Il ponte è **sollevato**: pertanto è **chiuso** al transito di autoveicoli e consente il passaggio contemporaneo di una o più imbarcazioni nel tratto di fiume considerato.

L'apertura (transizione da TB a TA) e la chiusura (transizione da TA a TB) del ponte possono avvenire solo **in condizioni di assenza di autoveicoli sul ponte e di assenza di barche in transito nel tratto di fiume interessato.**
Il ponte ha una **capacita` limitata** pari a MAX, che esprime il numero massimo di autoveicoli che possono transitare contemporaneamente su di esso.
Gli autoveicoli si suddividono in 2 categorie: **mezzi pubblici** (es. taxi, bus, ecc.) e **mezzi privati**.
In particolare, le **imbarcazioni** devono avere la **precedenza su tutti gli autoveicoli** nell'attraversamento del tratto di fiume.
Quando una barca giunge in prossimità del ponte:

- se il ponte è nello stato TA, la barca attende; non appena vi saranno le condizioni per la transizione di stato (TA->TB) il ponte passerà automaticamente allo stato TB per consentire il transito delle barche.
- pe il ponte è nello stato TB, la barca percorre il tratto di fiume.

Quando un autoveicolo vuole attraversare il ponte:

- se il ponte è nello stato TB, l'autoveicolo attende; non appena vi saranno le condizioni per la transizione di stato (TB->TA) il ponte passerà automaticamente allo stato TA per consentire il transito degli autoveicoli.
- quando il ponte è nello stato TA, l'autoveicolo attraversa il ponte.

Nell'accesso al ponte da parte dei veicoli, **i mezzi pubblici** dovranno avere la **precedenza** su quelli **privati**.
Realizzare un'applicazione concorrente in Java basata sul monitor nella quale ogni autoveicolo ed ogni barca siano rappresentati da thread distinti.
La politica di sincronizzazione dovrà tenere in considerazione tutti i vincoli dati.

Sistemi Operativi T
Prova di laboratorio
15 Luglio 2020

Esercizio di Programmazione Concorrente in Java

Si consideri il centro commerciale ***GoodBuy*** durante l'emergenza COVID-19.
Il centro è composto da 2 negozi:

- il supermercato ***GoodCoop***;
- il negozio di dispositivi elettronici e di elettrodomestici ***MediaGoods***.

Durante l'emergenza sanitaria si è dovuto applicare un protocollo per la regolazione degli accessi dei clienti ai negozi del centro commerciale, in accordo con i seguenti vincoli:

- il supermercato non può accogliere più di **NS clienti** contemporaneamente;
- il negozio *MediaGoods* non può accogliere più di **NM clienti** contemporaneamente;
- il numero massimo di clienti complessivamente ammessi all'interno del centro commerciale sia N.

Si assuma che **NS+NM > N**.
Si assuma inoltre, per semplicità, che ogni cliente acceda al centro per visitare uno ed un solo negozio; l'indicazione del negozio desiderato viene fornita dal cliente al momento dell'accesso.
Il comportamento di ogni cliente di un negozio x (supermercato o MediaGoods) sarà quindi articolato in 3 fasi:

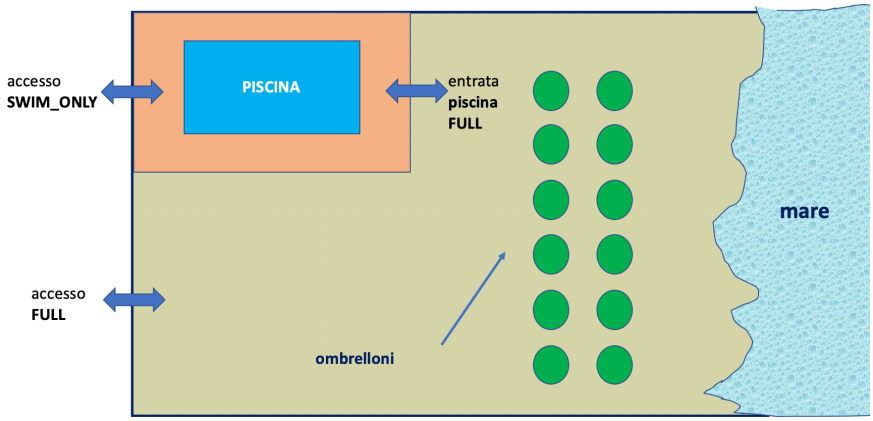
1. **Accesso** al centro per entrare nel negozio x; al termine di questa fase il cliente ha occupato un posto nel negozio x.
2. < visita del negozio x>
3. **Uscita** dal negozio x e dal centro commerciale.

I negozi vengono **periodicamente** sottoposti a **sanificazione**, tramite l'intervento di 1 **addetto** che si occupa di questo compito per entrambi i negozi. La fase di sanificazione di un negozio richiede che il negozio sia vuoto (ovvero non vi sia alcun cliente al suo interno).

La politica di controllo degli accessi ai negozi dovrà dare la **precedenza all'addetto**, rispetto ai clienti. Tra i clienti, si dia la **precedenza ai clienti del negozio che ha più posti liberi**.
Realizzare un'applicazione concorrente in Java basata sul **monitor** nella quale **Clienti** e **Addetto alla sanificazione** siano rappresentati da **thread concorrenti**.
La sincronizzazione tra i thread dovrà tenere conto di tutti i vincoli dati.

Sistemi Operativi T
Prova di laboratorio
13 Luglio 2021

Esercizio di Programmazione Concorrente in Java



Si consideri uno stabilimento balneare in una località marittima, suddiviso in **due aree distinte**:

- la **spiaggia** attrezzata con ombrelloni e sdraio;
- una **piscina**, situata in posizione arretrata rispetto alla spiaggia

I **clienti dello stabilimento balneare** possono essere di due tipi:

- clienti "SWIM_ONLY" che usano solo la PISCINA, accedendo da un'entrata dedicata SWIM_ONLY, esterna allo stabilimento (v. figura);
- clienti "FULL" che usufruiscono della SPIAGGIA che della PISCINA: ogni cliente di questa categoria **occupa un ombrellone** e, una volta entrato nello stabilimento tramite l'accesso FULL (v. figura), può spostarsi tra le 2 aree. Nel caso voglia andare in piscina, utilizzerà l'entrata piscina FULL (v. figura); quando il cliente FULL è in piscina, il suo ombrellone rimane occupato. I clienti FULL possono essere di 2 tipi:
    - abbonati: hanno acquistato un abbonamento stagionale
    - occasionali: accedono allo stabilimento occasionalmente

La spiaggia è attrezzata con N_OMB ombrelloni, pertanto un cliente FULL potrà entrare nello stabilimento solo se c'è un ombrellone libero.

L'area della PISCINA ha una capacità limitata fissata rispettivamente a Np (massimo numero di clienti ammessi in piscina): l'accesso alla piscina di ogni cliente (cliente SWIM_ONLY o cliente FULL) dovrà essere sempre subordinato al rispetto del vincolo di capacità.

Pertanto, **il comportamento tipico di un cliente SWIM_ONLY** è il seguente:

1. **Entra in piscina** tramite l'entrata esterna;

   <rimane in piscina per un tempo arbitrario>

2. **Esce dalla piscina** tramite l'entrata SWIM_ONLY

Sistemi Operativi T
Prova di laboratorio
13 Luglio 2021

Il comportamento tipico di un cliente FULL è, invece, il seguente:

1. **Entra nello stabilimento occupando un ombrellone** tramite l'entrata FULL;

   Ripete per un numero arbitrario di volte: {

   { <sta in spiaggia per un tempo arbitrario>

   Se decide di andare in piscina:

   { 1.a **Entra in piscina** dall'entrata piscina FULL (il suo ombrellone rimane occupato)

   <rimane in piscina per un tempo arbitrario>

   1.b **Esce dalla piscina** tramite l'entrata piscina FULL

   }

   }

2. **Esce dallo stabilimento** tramite l'accesso FULL

Realizzare un'applicazione in **Java** basata su monitor e variabili condizione, nella quale i **clienti,** siano rappresentati da **thread concorrenti**. La sincronizzazione tra i processi dovrà tenere conto dei vincoli dati ed, inoltre, dei seguenti vincoli:

- **nell'ingresso dei clienti FULL allo stabilimento**, i clienti abbonati siano privilegiati rispetto agli occasionali;
- **nell'ingresso alla piscina:** i clienti SWIM_ONLY abbiano la precedenza sui clienti FULL; tra i clienti FULL gli abbonati abbiano la priorità sugli occasionali.

Sistemi Operativi T
Prova di laboratorio - 10 Giugno 2022

GRUPPO A-K

2. Esercizio di Programmazione Concorrente in Java [punti 10]

Si consideri una sede locale dell'azienda di logistica ***ASped,*** che si occupa della consegna in zona di pacchi provenienti da tutto il territorio nazionale.
La sede è situata in una località del Centro Italia e svolge il ruolo di **centro di smistamento** delle merci provenienti da due direttrici:

- la direttrice Nord, attraverso la quale viaggiano i camion che trasportano i pacchi provenienti da località del nord Italia;
- la direttrice Sud, attraverso la quale viaggiano i camion che trasportano i pacchi provenienti da località del sud Italia;

Pertanto, ogni **pacco** è caratterizzato da una provenienza P, che può essere Nord o Sud Italia e da una destinazione (sempre appartenente alla zona in cui si trova la sede).
Per semplicità, si assuma che tutti i pacchi abbiano la stessa dimensione.

I pacchi **arrivano al centro di smistamento** tramite **Camion** ognuno dei quali depositerà nel centro un insieme di pacchi con la stessa provenienza (Nord o Sud)
Pertanto, ogni camion ha una provenienza predefinita (Nord o Sud), ed inoltre un contenuto costituito da un numero arbitrario **nc** di pacchi (nc non è una costante, ma può variare da camion a camion).
All'arrivo al Centro di smistamento ogni camion **scarica tutti gli nc pacchi** nel **deposito** del centro.
Il deposito ha una capacità limitata a **MaxD** pacchi.
Pertanto, lo scarico di un camion C può avvenire solo se c'è posto nel deposito per tutti gli nc pacchi contenuti in C.

I pacchi immagazzinati nel Centro di Smistamento vengono prelevati da **Furgoni** ognuno dei quali si occuperà, della consegna verso la destinazione locale.
Ogni furgone trasporta un numero costante NF di pacchi.
Prima di partire, ogni furgone carica dal deposito NF pacchi. Pertanto se gli NF pacchi da caricare non sono disponibili il Furgone attende.

I furgoni possono essere di due tipi:

- **aziendale**, ovvero di proprietà dell'azienda *ASped*;
- **esterno**, ovvero di proprietà di un'azienda convenzionata con *ASped.*

Realizzare un'applicazione concorrente in Java basata sul monitor nella quale **Furgoni e Camion** siano rappresentati da un **thread distinti**.
La politica di sincronizzazione dovrà tenere in considerazione tutti i vincoli dati, ed inoltre:

- nello **scarico di camion**: tenuto conto del numero totale di camion scaricati per ogni provenienza, si dia la precedenza ai camion che provengono dalla zona con il minor numero di camion scaricati**;**
- nel **carico di furgoni**: i furgoni aziendali dovranno avere la precedenza su quelli esterni**.**