Esercizi Z

università di ferrara

DA SEICENTO ANNI GUARDIAMO AVANTI.

Presentazione di specifiche Z

- Per favorire la leggibilità, le specifiche Z contengono una parte in prosa che introduce e spiega la parte formale (schemi).
- Generalmente, la struttura di una specifica Z è la seguente:

1. Breve introduzione che dà una descrizione generale del sistema

Presentazione di specifiche Z

2. Lista dei tipi definiti dall'utente e delle costanti
3. Schema che descrive lo stato del sistema
4. Stato iniziale
5. Specifica di tutte le operazioni

Esercizio 1

- Si specifichi in Z un sistema di controllo accessi ad un parcheggio avente N posti macchina. L'accesso al parcheggio è consentito unicamente a coloro che sono in possesso di un apposito tesserino e purché vi siano dei posti liberi nel parcheggio. Si modellino in particolare le seguenti operazioni:

1. Richiesta di una tessera: fornendo Nome e Cognome del nuovo utente il sistema provvede ad aggiornare l'elenco degli utenti e rilascia una tessera (modellabile attraverso un naturale) per l'accesso al parcheggio.

Esercizio 1

2. Ritiro di una tessera: permette di disabilitare una tessera, ossia cancellare il proprietario dall'elenco utenti.
3. Ingresso al parcheggio: permette ad un utente di entrare nel parcheggio facendo uso della tessera purché vi siano ancora posti liberi.
4. Uscita dal parcheggio: permette di uscire dal parcheggio Inoltre il sistema deve far si che NON sia possibile entrare due volte consecutive con la stessa tessera senza un'uscita intermedia.

Soluzione

- Tipo definito dall'utente: Persone = {(Nome, Cognome) | Nome Cognome è una persona}
- Costante N (numero di posti del parcheggio)

Stato, inizializzazione, successo

- Variabili di stato:
  - presenti: numeri di tessera degli utenti con l'auto nel parcheggio
  - utenti: funzione (parziale) che associa a ogni persona il suo numero di tessera
- Inizialmente il parcheggio e l'insieme degli utenti sono vuoti.

Parcheggio

$$\begin{array}{l} presenti: \mathbb{P}\ \mathbb{N} \\ utenti: Persone \pfun \mathbb{N} \\ \hline |\ presenti\ | \leqslant N \\ presenti \subseteq ran\ utenti \end{array}$$

InitParcheggio

$$\begin{array}{l} Parcheggio' \\ \hline presenti' = \varnothing \\ utenti' = \varnothing \end{array}$$

Successo

$$\begin{array}{l} rep!: Report \\ \hline rep! = \text{'Okay'} \end{array}$$

Richiesta tessera

Precondizione: l'utente da aggiungere non deve essere già presente

AggiungiUtente

ΔParcheggio
nuovo_utente?: Persone
nuova_tessera!: $\mathbb{N}$

---

nuovo_utente? $\notin$ dom utenti
nuova_tessera! $\notin$ ran utenti
utenti' = utenti $\cup$ {nuovo_utente? $\mapsto$ nuova_tessera!}
presenti' = presenti

UtenteGiàInserito

ΞParcheggio
nuovo_utente?: Persone
rep!: Report

---

nuovo_utente? $\in$ dom utenti
rep! = 'Utente già inserito'

RichiediTessera $\widehat{=}$ AggiungiUtente $\wedge$ Successo
$\vee$
UtenteGiàInserito

Ritiro tessera

Precondizione: la tessera è associata a un utente

TogliTessera

ΔParcheggio
tessera?: $\mathbb{N}$

---

tessera? $\in$ ran utenti
utenti' = utenti $\setminus$ {utenti^{-1}(tessera?) $\mapsto$ tessera?}
presenti' = presenti

TesseraNonRegistrata

ΞParcheggio
tessera?: $\mathbb{N}$
rep!: Report

---

tessera? $\notin$ ran utenti
rep! = 'Tessera non registrata'

RitiraTessera $\widehat{=}$ TogliTessera $\wedge$ Successo
$\vee$
TesseraNonRegistrata

Ingresso nel parcheggio

Precondizioni:

1. C'è un posto libero
2. L'utente con tessera? non è già presente
3. tessera? corrisponde a un utente registrato

EntraParcheggio

$\Delta Parcheggio$
$tessera? : \mathbb{N}$

$|\ presenti\ | < N$
$tessera? \notin presenti$
$tessera? \in ran\ utenti$
$utenti' = utenti$
$presenti' = presenti \cup \{tessera?\}$

ParcheggioPieno

$\Xi Parcheggio$
$rep! : Report$

$|\ presenti\ | = N$
$rep! = \text{'Parcheggio pieno'}$

Ingresso nel parcheggio

TesseraGiàPresente

ΞParcheggio
tessera?: $\mathbb{N}$
rep!: Report

---

tessera? $\in$ presenti
rep! = 'L'utente è già presente nel parcheggio'

IngressoParcheggio $\widehat{=}$ EntraParcheggio $\wedge$ Successo
$\vee$
ParcheggioPieno
$\vee$
TesseraNonRegistrata
$\vee$
TesseraGiàPresente

Uscita dal parcheggio

Precondição: l'utente con tessera? è presente

EsciParcheggio

ΔParcheggio
tessera?: $\mathbb{N}$

---

tessera? $\in$ presenti
utenti' = utenti
presenti' = presenti \ {tessera?}

TesseraNonPresente

ΞParcheggio
tessera?: $\mathbb{N}$
rep!: Report

---

tessera? $\notin$ presenti
rep! = 'L'utente non era presente nel parcheggio'

UscitaParcheggio $\widehat{=}$ EsciParcheggio $\wedge$ Successo

$\vee$

TesseraNonPresente

Esercizio 2

- Un cineclub ha tre sale di proiezione (identificate da 1 a 3). Per prenotare una proiezione in uno dei giorni della settimana corrente (da lun a ven) occorre specificare il giorno (intero da 1 a 5), il proprio codice fiscale (CF) e quello del film (che deve essere in catalogo). Se esiste una sala disponibile per quel giorno, essa viene assegnata al CF specificato (e viene restituito il numero della sala assegnata).
- Si specifichi in Z un tale sistema di gestione delle sale del cineclub e in particolare la prenotazione di una proiezione.

Soluzione

- Tipi definiti dall'utente
  - Giorno=1..5
  - Sala=1..3
  - CodiceFiscale: insieme dei codici fiscali delle persone
  - CodiceFilm: insieme dei codici dei film
- Variabili che descrivono lo stato del sistema:
  - catalogo: insieme dei film appartenenti al catalogo
  - prenotazioni: funzione parziale che associa alle sale e ai giorni la persona che le ha prenotate e il film
- Inizialmente, il catalogo è vuoto e non ci sono prenotazioni.

Stato, inizializzazione, successo

Cineclub

$catalogo: \mathbb{P}\ CodiceFilm$

$prenotazioni: Sala \times Giorno \pfun CodiceFiscale \times CodiceFilm$

$catalogo \supseteq \{codice\_film: CodiceFilm \mid (\exists\ cf: CodiceFiscale \bullet (cf, codice\_film) \in \ran prenotazioni)\}$

InitCineclub

$\Delta Cineclub$

$prenotazioni' = \emptyset$

$catalogo' = \emptyset$

Successo

$rep!: Report$

$rep! = \text{'Okay'}$

PrenotaSala

- Precondizioni:
  - C’è una sala libera
  - Il film è in catalogo

Prenota

ΔCineclub
cf?: CodiceFiscale
giorno?: Giorno
film?: CodiceFilm
sala!: Sala

---

film? $\in$ catalogo
(sala!,giorno?) $\notin$ dom prenotazioni
prenotazioni' = prenotazioni $\cup$ {(sala!,giorno?) $\mapsto$(cf?,film?)}

PrenotaSala

NessunaSalaLibera

ΞCineclub
giorno?: Giorno
rep!: Report

$\nexists$ sala: Sala • ((sala,giorno?) $\notin$ dom prenotazioni)
rep! = 'Nessuna sala libera per il giorno richiesto'

FilmNonInCatalogo

ΞCineclub
film?: CodiceFilm
rep!: Report

---

film? $\notin$ catalogo
rep! = 'Il film non appartiene al catalogo'

PrenotaSala $\widehat{=}$ Prenota $\wedge$ Successo
$\vee$
NessunaSalaLibera
$\vee$
FilmNonInCatalogo

Esercizio 3

- Si specifichi in Z la procedura di erogazione di un mutuo bancario da parte di una filiale bancaria.
- Un mutuo può essere erogato solo a chi è correntista della filiale.
- Per ottenere un mutuo occorre fare domanda alla filiale, specificando l'importo e il numero di conto corrente (numero naturale). Il mutuo viene erogato se si è correntisti della filiale e se l'importo in conto corrente è di almeno 1000 euro.

Esercizio 3

- Si modellino in particolare le seguenti operazioni:
  - Apertura di un conto corrente presso la filiale. Fornendo Nome, Cognome, Importo_iniziale è restituito un identificativo del conto corrente aperto presso la filiale, nel quale viene depositato l'importo iniziale specificato, che deve essere maggiore di 0.
  - Richiesta di un mutuo. Fornendo il numero di Conto e l'Importo_richiesto, se l'operazione va a buon fine si restituisce un messaggio di successo. Il mutuo viene erogato se si ha un conto presso la filiale e se l'importo in conto corrente è di almeno 1000 euro.

Soluzione

- Tipi definiti dall'utente
  - Nomi
  - Cognomi
- Variabili che descrivono lo stato del sistema:
  - conti_correnti: funzione parziale che associa alle persone il relativo conto corrente,
  - saldo: funzione parziale che associa a ciascun conto corrente il suo saldo
- Inizialmente non ci sono conti correnti

Stato, inizializzazione, successo

Filiale

$conti\_correnti: Nomi \times Cognomi \pfun \mathbb{N}$

$saldo: \mathbb{N} \pfun \mathbb{R}$

$dom\ saldo = ran\ conti\_correnti$

InitFiliale

$\Delta Filiale$

$conti\_correnti' = \emptyset$

$saldo' = \emptyset$

Successo

$rep!: Report$

$rep! = 'Okay'$

ApriConto

Precondizione: l'importo iniziale e' maggiore di 0

ApriConto

ΔFiliale
nome?: Nomi
cognome?: Cognomi
importo_inziale?: $\mathbb{R}$
conto!: $\mathbb{N}$

---

importo_iniziale? > 0
conto! $\notin$ ran conti_correnti
conti_correnti' = conti_correnti $\cup$ {(nome?,cognome?) $\mapsto$conto!}
saldo' = saldo $\cup$ {conto! $\mapsto$importo_iniziale}

ApriConto

SenzaSoldi

ΞFiliale
nome?: Nomi
cognome?: Cognomi
importo_inziale?: $\mathbb{R}$
rep!: Report

---

importo_iniziale? $\leqslant 0$
rep!='Senza soldi non si apre un conto'

$$\text{AperturaConto} \mathrel{\widehat=} \text{ApriConto} \land \text{Successo}$$
$$\lor$$
$$\text{SenzaSoldi}$$

ChiediMutuo

- Precondizioni:
  - il richiedente ha un conto presso la filiale
  - ci sono almeno 1000 euro sul conto

ChiediMutuo

ΔFiliale
conto?: $\mathbb{N}$
importo_richiesto?: $\mathbb{R}$

---

conto? $\in$ ran conti_correnti
saldo(conto?) $\geqslant$ 1000

ContoNonValido

ΞFiliale
conto?: $\mathbb{N}$
rep!: Report

---

conto? $\notin$ ran conti_correnti
rep!='Conto corrente non valido'

SaldoInsufficiente

ΞFiliale
conto?: $\mathbb{N}$
rep!: Report

---

saldo(conto?) < 1000
rep!='Saldo insufficiente'

RichiediMutuo $\widehat{=}$ ChiediMutuo $\wedge$ Successo
$\vee$
ContoNonValido
$\vee$
SaldoInsufficiente

Esercizio 4

- Si dia una specifica in Z di uno sportello Bancomat avente le seguenti caratteristiche. Per poter effettuare un'operazione allo sportello (prelievo o saldo) si fornisce un numero di conto corrente (intero) e un codice segreto (intero). Il sistema verifica che il codice fornito sia effettivamente quello associato al numero di conto corrente inserito.
- Se la verifica va a buon fine il sistema permette di effettuare:
  - un prelievo di contanti (in quantita' fissa, ad esempio 100 EUR)
  - una richiesta di saldo del conto
- In particolare devono essere specificate in Z le seguenti 4 operazioni:

Esercizio 4

- Accesso(Numero_conto, Numero_segreto). L'operazione di Accesso andata a buon fine ha come effetto quello di bloccare lo sportello assegnandolo al codice specificato e proibendo ulteriori accessi da parte di altri utenti con codice diverso.
- Preleva() che restituisce il valore della cifra prelevata (100 EUR) e modifica il valore del saldo. Tale operazione ha effetto solo se è stata preceduta da un'operazione di Accesso andata a buon fine (ovvero lo sportello e` assegnato a un utente).
- Saldo() che restituisce il valore del saldo del conto corrente. Tale operazione ha effetto solo se è stata preceduta da un'operazione di Accesso
- Fine() che sblocca il sistema precedentemente bloccato da un'operazione di accesso.

Soluzione

- Variabili che descrivono lo stato del sistema:
  - utenti, funzione parziale che associa ad ogni conto corrente (rappresentato da un intero positivo) un numero segreto (intero)
  - saldo, funzione parziale che associa ad ogni conto corrente il suo saldo
  - bloccato, rappresenta lo stato del Bancomat. Se vale 0 lo sportello è libero, se diverso da zero è occupato (dall'utente il cui numero di conto coincide con il valore della variabile bloccato).

Stato, inizializzazione, successo

Bancomat

$bloccato : \mathbb{N}$

$utenti : \mathbb{N}_1 \pfun \mathbb{N}$

$saldo : \mathbb{N}_1 \pfun \mathbb{R}$

$dom\ utenti = dom\ saldo$

Inizialmente lo sportello è libero e vengono definiti i codici dei conti correnti e il saldo

InitBancomat

$Bancomat'$

$bloccato' = 0$

$utenti' = \{ 698978 \mapsto 4324, 345678 \mapsto 4321 , \ldots\}$

$saldo' = \{ 698978 \mapsto 1200, 345678 \mapsto 3000, \ldots\}$

Successo

$rep! : Report$

$rep! = 'Okay'$

Accesso al bancomat

Precondizioni:
1)Deve esistere un conto corrente con il Numero_conto e Numero_segreto specificati
2)Lo sportello deve essere libero

AccessoAlBancomat

ΔBancomat
numero_conto?: $\mathbb{N}_1$
numero_segreto?: $\mathbb{N}$

---

numero_conto? $\mapsto$numero_segreto? $\in$ utenti
bloccato $=$ 0
bloccato' $=$ numero_conto?
utenti' = utenti
saldo' = saldo

NumeroCodiceNonCorrisp

ΞBancomat
numero_conto?: $\mathbb{N}_1$
numero_segreto?: $\mathbb{N}$
rep!: Report

---

numero_conto? $\mapsto$numero_segreto? $\notin$ utenti
rep! = 'Numero segreto e conto non corrispondenti'

BancomatBloccato

ΞBancomat
rep!: Report

---

bloccato $\neq$ 0
rep! = 'Bancomat bloccato'

Accesso $\widehat{=}$ AccessoAlBancomat $\wedge$ Successo $\vee$
NumeroCodiceNonCorrisp $\vee$
BancomatBloccato

Prelievo

Precondizione: il bancomat deve essere assegnato a un conto corrente (ciò significa che è stata eseguita prima un'operazione di accesso)

PrelievoDalConto

ΔBancomat
prelievo!: $\mathbb{N}$

---

bloccato $\neq 0$
prelievo! $= 100$
utenti' $=$ utenti
bloccato' $=$ bloccato
saldo' $=$ saldo $\oplus$
{bloccato $\mapsto$ saldo(bloccato)-100}

BancomatNonBloccato

ΞBancomat
rep!: Report

---

bloccato $= 0$
rep! = 'Sportello Bancomat non assegnato'

Prelievo $\widehat{=}$ PrelievoDalConto $\wedge$ Successo
$\vee$
BancomatNonBloccato

Saldo

Precondizione: il bancomat deve essere assegnato a un conto corrente (ciò significa che e` stata eseguita prima un'operazione di accesso)

SaldoDelConto

ΞBancomat

saldo_del_conto!: $\mathbb{R}$

---

bloccato $\neq 0$

saldo_del_conto! = saldo(bloccato)

Saldo $\widehat{=}$ SaldoDelConto $\wedge$ Successo

$\vee$

BancomatNonBloccato

Fine

Precondizione: il bancomat deve essere assegnato a un codice

FineBancomat

ΔBancomat

---

bloccato $\neq 0$
utenti' = utenti
bloccato' $= 0$
saldo' $=$ saldo

Fine $\widehat{=}$ FineBancomat $\wedge$ Successo
$\vee$
BancomatNonBloccato

Prelievo (versione 2)

Accesso ha l'effetto di bloccare lo sportello impedendo l'accesso ad altri utenti. Se nelle operazioni Preleva e Saldo si inserisce il conto, si può controllare se
è proprio l'utente che ha bloccato il Bancomat a effettuare la transazione.
Precondizioni:
1. Il bancomat deve essere assegnato a un conto corrente (ciò significa che è stata eseguita prima un'operazione di accesso)
2. L'utente che richiede l'operazione è quello che ha bloccato lo sportello (autenticazione tramite il numero di conto corrente)

Prelievo (versione 2)

PrelievoDalConto

ΔBancomat
prelievo!: $\mathbb{N}$
numero_conto?: $\mathbb{N}_1$

---

bloccato = numero_conto?
prelievo! = 100
utenti' = utenti
bloccato' = bloccato
saldo' = saldo $\oplus$ {bloccato $\mapsto$ saldo(bloccato)-100}

AccessoNonAutorizzato

ΞBancomat
rep!: Report
numero_conto?: $\mathbb{N}_1$

---

bloccato $\neq$ numero_conto?
bloccato $\neq$ 0
rep! = 'Sportello Bancomat assegnato ad un altro utente'

BancomatNonBloccato

ΞBancomat
rep!: Report

---

bloccato = 0
rep! = 'Sportello Bancomat non assegnato'

Prelievo $\widehat{=}$ PrelievoDalConto $\wedge$ Successo
$\vee$
AccessoNonAutorizzato
$\vee$
BancomatNonBloccato

Saldo (versione 2)

Precondizioni:
1. Il bancomat deve essere assegnato a un conto corrente (ciò significa che è stata eseguita prima un'operazione di accesso)
2. L'utente che richiede l'operazione è quello che ha bloccato lo sportello (autenticazione tramite il conto corrente)

SaldoDelConto

ΞBancomat
saldo_del_conto!: $\mathbb{Z}$
numero_conto?: $\mathbb{N}_1$

---

bloccato = numero_conto?
saldo_del_conto! = saldo(bloccato)

$$Saldo \mathrel{\widehat=} SaldoDelConto \land Successo \lor AccessoNonAutorizzato \lor BancomatNonBloccato$$

Fine (versione 2)

Precondizioni:

1. Il bancomat deve essere assegnato a un codice
2. L'utente che richiede l'operazione è quello che ha bloccato lo sportello (autenticazione tramite il conto corrente)

FineBancomat

ΔBancomat
numero_conto?: $\mathbb{N}_1$

---

bloccato = numero_conto?
utenti' = utenti
bloccato' = 0
saldo' = saldo

$$Fine \mathrel{\widehat=} FineBancomat \land Successo \lor AccessoNonAutorizzato \lor BancomatNonBloccato$$

Esercizio 5

- Si dia una specifica in Z di un sistema di prenotazione delle aule di una facoltà. Ciascuna aula della facoltà è identificata da un numero (da 1 a 20). Si suppone che le aule abbiano tutte la stessa capienza. Ogni aula può essere prenotata il mattino o il pomeriggio di un giorno dell'anno (365 giorni) da un membro della facoltà. I membri della facoltà sono identificati da un codice numerico personale. La stessa aula può essere prenotata ***al più*** da una persona per un dato giorno e parte della giornata. Non c'è limitazione al numero di aule che una persona può prenotare per un dato giorno e parte della giornata.
- Per fare una prenotazione, una persona indica il giorno, la parte della giornata e il proprio codice personale. La prenotazione va a buon fine se il codice esiste tra quelli assegnati ai membri della facoltà e se esiste un'aula libera per quel giorno e parte della giornata, aula che viene prenotata assegnandola alla persona.

Esercizio 5

- In particolare, devono essere specificate in Z le seguenti operazioni:

1. Registrazione di un nuovo membro della Facoltà, al quale viene assegnato un codice identificativo personale;
2. Prenotazione di un'aula: dati il giorno, l'orario e il codice identificativo personale, il sistema prenota un'aula libera per quell'orario e giorno se essa esiste e la restituisce;
3. Annullamento di una prenotazione di un'aula: dati il giorno, l'orario, il numero dell'aula e il codice identificativo personale di chi aveva fatto la prenotazione, il sistema libera l'aula precedentemente prenotata (se tale prenotazione esiste).

Tipi definiti dall'utente

- Persone = {(Nome,Cognome) | Nome Cognome indica una persona}
- Aule= 1..20
- Giorni = 1..365
- Orari = {1,2}

Stato, inizializzazione, successo

Variabili che descrivono lo stato del sistema:

1. membri, funzione che associa ad ogni persona che fa parte della facoltà un codice (naturale)
2. prenotazioni, funzione che associa ad ogni tripla <orario,giorno,aula> il codice della persona che ha fatto la prenotazione

Inizialmente non c'è nessun membro registrato e nessuna prenotazione.

Facoltà

$membri: Persone \pfun \mathbb{N}$

$prenotazioni: Orari \times Giorni \times Aule \pfun \mathbb{N}$

$ran\ prenotazioni \subseteq ran\ membri$

InitFacoltà

$Facoltà'$

$membri' = \{\}$

$prenotazioni' = \{\}$

Successo

$rep!: Report$

$rep! = 'Okay'$

Registrazione

Precondizione: la persona da aggiungere non deve essere già presente

AggiungiMembro

ΔFacoltà
nuovo_membro?: Persone
nuovo_codice!: $\mathbb{N}$

---

nuovo_membro? $\notin$ dom membri
nuovo_codice! $\notin$ ran membri
membri' =membri $\cup$ {nuovo_membro? $\mapsto$nuovo_codice!}
prenotazioni' = prenotazioni

MembroGiàInserito

ΞFacoltà
nuovo_membro?: Persone
rep!: Report

---

nuovo_membro? $\in$ dom membri
rep! = 'Membro già inserito'

Registrazione $\widehat{=}$ AggiungiMembro $\wedge$ Successo
$\vee$
MembroGiàInserito

Prenotazione

Precondizioni:
1. il numero di codice deve appartenere a un membro della Facoltà
2. c'è un'aula libera nell'orario e giorno indicati non ancora prenotata

PrenotaAula

ΔFacoltà
codice?: $\mathbb{N}$
giorno?: Giorni
orario?: Orari
aula!: Aule

---

codice? $\in$ ran membri
$\exists$ aula: Aule •
((orario?,giorno?,aula) $\notin$ dom prenotazioni $\wedge$
aula! = aula)
prenotazioni' = prenotazioni $\cup$
{(ora?,giorno?,aula!) $\mapsto$(codice?)}
membri' = membri

Prenotazione: alternativa

PrenotaAula

ΔFacoltà
codice?: $\mathbb{N}$
giorno?: Giorni
orario?: Orari
aula!: Aules

---

codice? $\in$ ran membri
(orario?,giorno?,aula!) $\notin$ dom prenotazioni
prenotazioni' = prenotazioni $\cup$
{(ora?,giorno?,aula!) $\mapsto$(codice?)}
membri' = membri

Infatti una variabile di uscita nella signature di uno schema equivale a una variabile esistenziale nella parte contenente i predicati, come si vede anche dallo schema hiding.

Prenotazione

NessunaAulaLibera

ΞFacoltà
giorno?: Giorni
orario?: Orari
rep!: Report

$\nexists$ aula: Aule •
(orario?,giorno?,aula) $\notin$ dom prenotazioni
rep! = 'Nessuna aula libera'

NonMembro

ΞFacoltà
codice?: $\mathbb{N}$
rep!: Report

---

codice? $\notin$ ran membri
rep! = 'Chi non è membro non può prenotare'

PrenotazioneAula $\widehat{=}$
PrenotaAula $\wedge$ Successo
$\vee$
NessunaAulaLibera
$\vee$
NonMembro

Libera aula

Precondizioni:
1. il numero di tessera/codice deve appartenere a un membro della Facoltà
2. l'aula che si vuole liberare nell'orario e giorno indicati compare risulta nelle prenotazioni fatte con il numero di tessera/codice indicato

LiberaAula

ΔFacoltà
codice?: $\mathbb{N}$
giorno?: Giorni
orario?: Orari
aula?: Aules

---

codice? $\in$ ran membri
(orario?,giorno?,aula?) $\mapsto$(codice?) $\in$ prenotazioni
prenotazioni' = prenotazioni $\setminus$
{(orario?,giorno?,aula?) $\mapsto$(codice?)}
membri' = membri

Libera aula

AulaGiaLibera

ΞFacoltà
codice?: $\mathbb{N}$
giorno?: Giorni
ora?: Orari
aula?: Aule
rep!: Report

(ora?,giorno?,aula?) $\mapsto$(codice?) $\notin$ prenotazioni
rep! = 'Prenotazione inesistente'

LiberazioneAula $\widehat{=}$ LiberaAula $\wedge$ Successo

$\vee$

AulaGiaLibera

$\vee$

NonMembro

Esercizio 6 (10/1/2011)

- Si specifichi in Z la procedura di gestione del portafoglio azioni dei clienti di una filiale bancaria. Ogni cliente della filiale ha un portafoglio azioni che memorizza, per ogni azione posseduta, la quantità. Ogni cliente ha un inoltre un conto corrente presso la filiale. Il prezzo di ciascuna azione si suppone noto alla filiale e fisso. Si modellino con tipi definiti dall'utente i numeri di conto, i nomi delle aziende quotate in borsa e la funzione prezzo.

Esercizio 6

- Si modellino in particolare le seguenti operazioni:

1. Acquisto di azioni: fornendo il numero di conto del cliente, nome dell'azienda e quantità, viene acquistato il numero di azioni della società richiesto. Le azioni vengono aggiunte al portafoglio del cliente e l'importo dedotto dal saldo del suo conto corrente. L'operazione fallisce se il cliente non ha sufficiente credito nel conto corrente.

Esercizio 6

2. Vendita di azioni: fornendo il numero di conto del cliente e il nome dell'azienda, la filiale vende tutte le azioni dell'azienda nel portafoglio del cliente. Il valore delle azioni viene aggiunto al saldo del conto corrente del cliente. L'operazione fallisce se l'azienda non è nel portafoglio del cliente.

Tipi definiti dall'utente

- [Conti, Aziende, Prezzi]
- Conti={insieme dei numeri di conto di tutti i clienti}
- Aziende={insieme delle aziende quotate in borsa}
- Prezzo=funzione che fornisce il prezzo di un'azione, Prezzi: Aziende $\longrightarrow \mathbb{R}$

Variabili di stato

- saldo: è una funzione totale che associa ad conto corrente un saldo in euro;
- portafoglio: è una funzione parziale che associa ad un conto corrente e ad una azienda il numero di azioni possedute

Stato e inizializzazione

Banca

$saldo: Conti \rightarrow \mathbb{R}$

$portafoglio: Conti \times Aziende \nrightarrow \mathbb{N}$

$dom\ saldo \supseteq \{c: Conti \mid \exists\ a \bullet (c,a) \in dom\ portafoglio\}$

InitBanca

$\Delta Banca$

$saldo' = \{\}$

$portafoglio' = \{\}$

Successo

$rep!: Report$

$rep! = 'Okay'$

Acquisto di azioni:

- Precondizione: saldo sufficiente

AcquistoOK

ΔBanca
conto?: Conti
azienda?: Aziende
quantità?: $\mathbb{N}$

$saldo(conto?) \geqslant Prezzo(azienda?)*quantità$
$portafoglio' = portafoglio \oplus \{(conto?, azienda?) \mapsto$
$\quad portafoglio((conto?, azienda?)) + quantità?\}$
$saldo' = saldo \oplus \{conto? \mapsto saldo(conto?) - Prezzo(azienda?)*quantità?\}$

Acquisto di azioni:

SaldoInsufficiente

ΞBanca
conto?: Conti
azienda?: Aziende
quantità?: $\mathbb{N}$
rep!: Report

Prezzo(azienda?)*quantità?> saldo(conto?)
rep! = 'Saldo insufficiente'

$$\text{Acquisto} \mathrel{\widehat=} \text{AcquistoOK} \land \text{Successo}$$
$$\lor$$
$$\text{SaldoInsufficiente}$$

Vendita di azioni:

- Precondizione: azioni possedute

VenditaOK

ΔBanca
conto?: Conti
azienda?: Aziende

(conto?,azienda?) $\in$ dom portafoglio
portafoglio'=portafoglio $\setminus$
{(conto?,azienda?) $\mapsto$ portafoglio((conto?,azienda?))}
conto'=conto $\oplus$ {conto? $\mapsto$ saldo(conto?)+
Prezzo(azienda?)*portafoglio((conto?,azienda?))}

Vendita di azioni:

AziendaNonPresente

ΞBanca
conto?: Conti
azienda?: Aziende

(conto?,azienda?) $\notin$ dom portafoglio
rep! = 'Azione non in portafoglio'

Vendita $\widehat{=}$ VenditaOK $\wedge$ Successo
$\vee$
AziendaNonPresente

Esercizio 7 (12/9/2011)

- Si dia una specifica in Z di un sistema di scheduling dei processi di un sistema operativo batch. Si supponga che il sistema disponga di quattro code a diversa priorità, con la coda 1 a priorità più alta e la coda 4 a priorità più bassa. Si supponga inoltre che le code possano contenere al massimo 10 processi e che i processi siano identificati da un numero intero.

Esercizio 7

- Si modellino in Z le seguenti operazioni:

1. Accodamento di un processo: dato l'identificativo del processo e la sua priorità (un numero da 1 a 4), inserire il processo nella coda relativa alla priorità. L'operazione fallisce se la coda è piena
2. Esecuzione di un processo: si rimuove il processo avente l'identificavo più basso dalla coda avente priorità più alta. L'operazione fallisce se tutte le code sono vuote.

Tipi e variabili

- Tipi definiti dall'utente:

[Code]

Code={1,2,3,4}

- Variabili che descrivono lo stato del sistema:

coda: è una funzione totale da Code agli insiemi di interi

Stato

SO

$$coda: Code \rightarrow \mathbb{P}\,\mathbb{N}$$

$$\forall\, c: Code \bullet |coda(c)| <= 10$$

Inizializzazione

InitSO

ΔSO

---

coda(1)'={}
coda(2)'={}
coda(3)'={}
coda(4)'={}

Successo

rep!: Report

---

rep! = 'Okay'

Accodamento

- Precondizione: spazio nella coda

AccodamentoOK

ΔSO

processo?: $\mathbb{N}$

priorità?: Code

$|code(priorità?)|<10$

$code' = code \oplus \{priorità? \mapsto code(priorità?) \cup \{processo?\}\}$

Accodamento

CodaPiena

ΞSO

pirorità?: Code

rep!: Report

|code(priorità?)|=10

rep! = 'Coda piena'

$$Accodamento \mathrel{\widehat{=}} AccodamentoOK \wedge Successo$$

$$\vee$$

$$CodaPiena$$

Esecuzione

- Precondizione: almeno una coda non vuota

EsecuzioneOK

$$\Delta SO$$
$$processo! : \mathbb{N}$$

$$\exists\ c: Code \bullet \forall\ c': Code \mid c'<c \bullet code(c')= \{\} \land \exists\ p: \mathbb{N} \mid p \in code(c) \bullet \forall\ p': \mathbb{N} \mid p' \in code(c) \bullet p<p' \land processo!=p \land code'=code \oplus \{c \mapsto code(c) \setminus \{processo!\}\}$$

Esecuzione

CodeVuote

ΞSO

$\forall$ c: Code • |coda(c)|=0

rep! = 'Code vuote'

$$Esecuzione \mathrel{\widehat{=}} EsecuzioneOK \wedge Successo \vee CodeVuote$$