Ingegneria del Software

Esercitazione 5

Stack

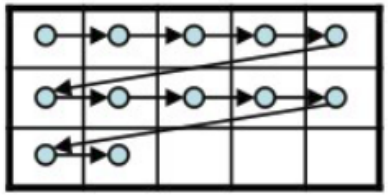
Implementare la classe Stack con i Generics

Stack (II)

Aggiungere un Iteratore alla classe Stack

Matrice

Scrivere un iteratore per gestire l'accesso sequenziale (Per riga) agli elementi di una matrice



```
public class Matrix{

    public int rows(){ /**/}
    public int columns(){ /**/}

    public float element(int row, int column){/**/}

}
```

Iteratore di Fibonacci

$$F_1 = 1,$$
$$F_2 = 1,$$
$$F_n = F_{n-1} + F_{n-2} \text{ (per ogni n>2)}$$

Polinomi

- Scrivere un iteratore che restituisca i coefficienti del polinomio, ordinati per grado
- **Esempio**:
- per 3 + 2x deve restituire la sequenza

  3

  2

  NoSuchElementException

Multi-threading

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

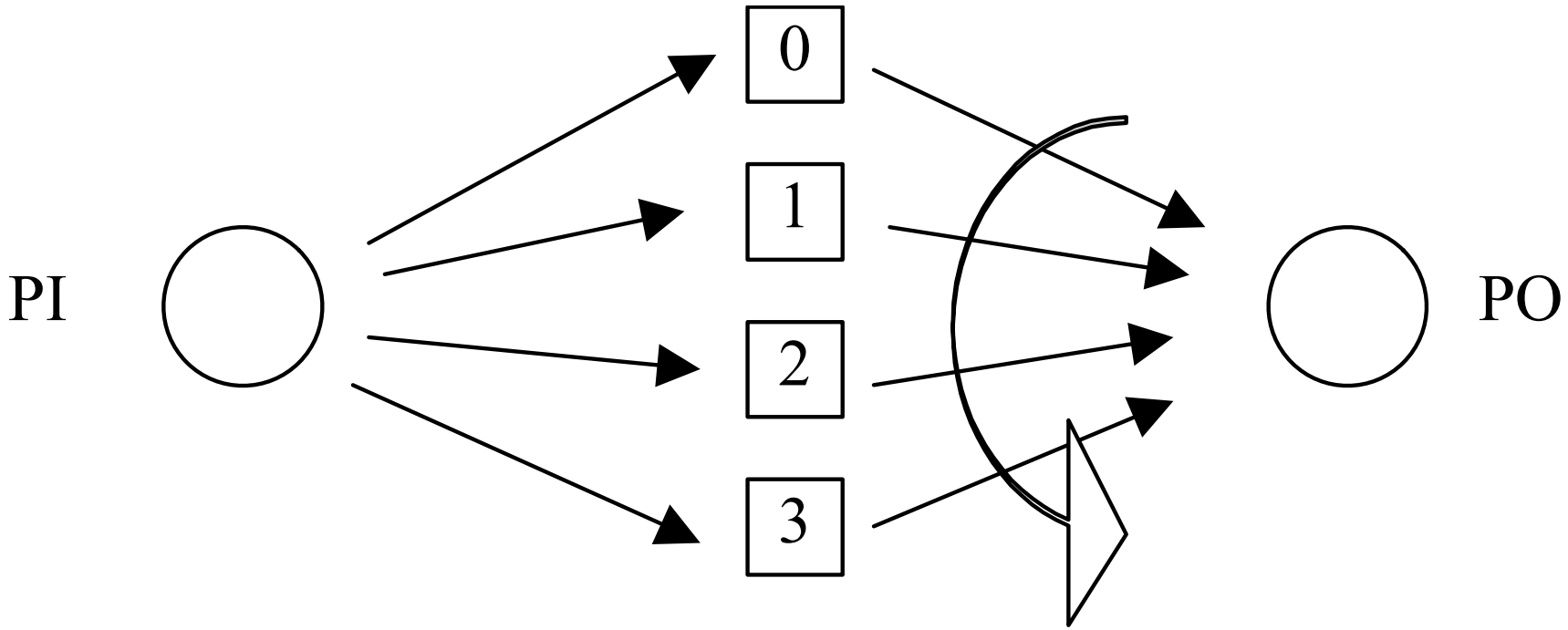
Dato un sistema con quattro buffer di caratteri.

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>

Un impianto con valvola

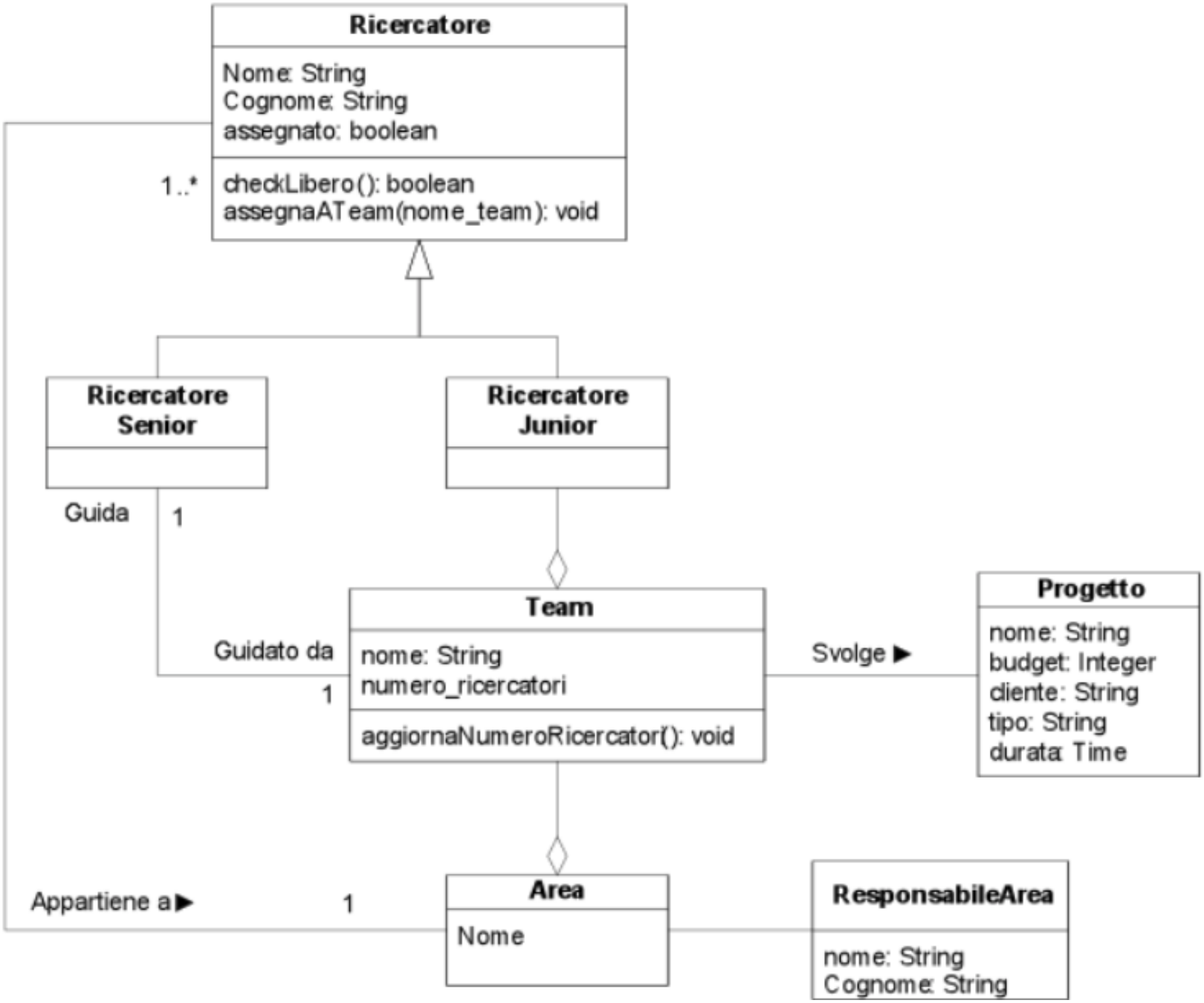
Un impianto può portarsi dallo stato di funzionamento normale N in uno stato di gestione di malfunzionamento M. Entrato in tale stato, entro 5s deve essere aperta una valvola di scarico. Se non si apre, l'impianto passa ad uno stato di fermo (F). Se la valvola viene aperta, essa rimane in tale stato per un tempo non inferiore a 5s e non superiore a 10s, poi l’impianto ritorna nello stato N.

UML

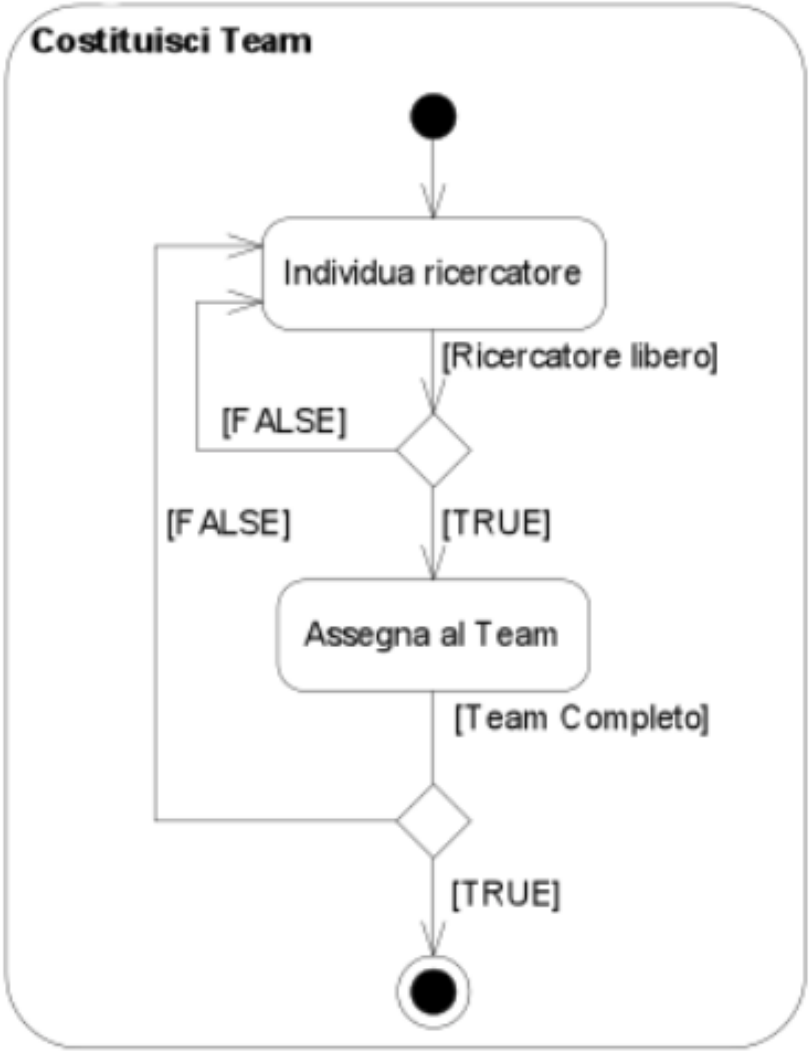
Centro di Ricerca

Un centro di ricerca è organizzato in team di ricercatori. Ogni team è guidato da un ricercatore senior e include uno o più ricercatori junior. I team sono raggruppati in aree con un responsabile di area. Ogni team ha come compito quello di svolgere un progetto.

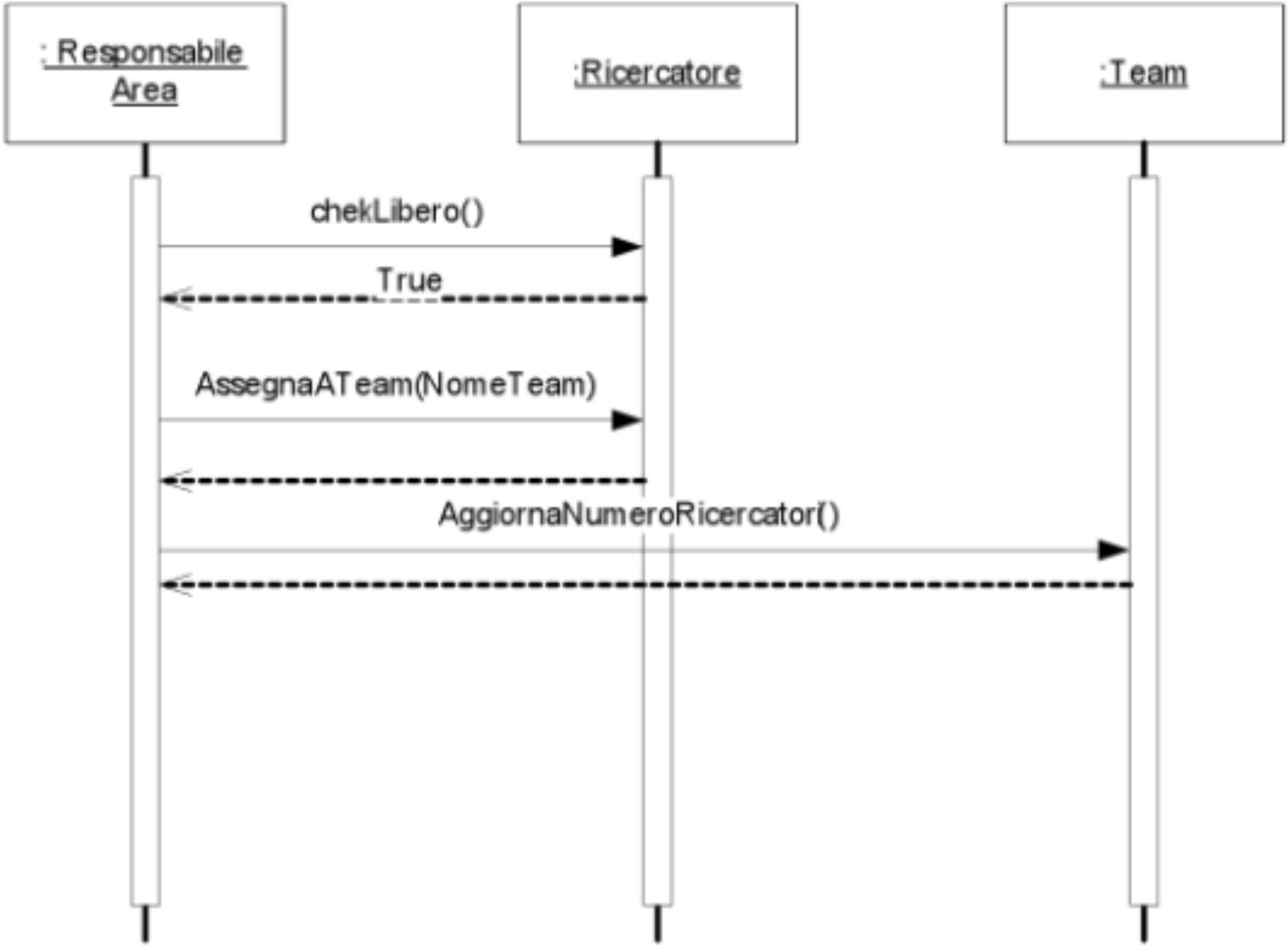
L'attivazione di un progetto prevede la selezione dell'area alla quale verrà assegnato il progetto, la costituzione del team e la scelta del responsabile del team. I ricercatori da assegnare al team sono scelti tra quelli disponibili nell'area (non assegnati ad alcun team). Al termine del progetto, le risorse del team tornano ad essere disponibili per altri progetti della stessa area.


Ricercatore
Nome: String
Cognome: String
assegnato: boolean
checkLibero(): boolean
assegnaATeam(nome_team): void
1..*
Ricercatore Senior
Ricercatore Junior
Guida
1
Team
nome: String
numero_ricercatori
aggiornaNumeroRicercator(): void
Guidato da
1
Svolge ▶
Progetto
nome: String
budget: Integer
cliente: String
tipo: String
durata: Time
Appartiene a ▶
1
Area
Nome
ResponsabileArea
nome: String
Cognome: String


Seleziona area
Associa progetto
Costituisci Team
Seleziona responsabile Team


Costituisci Team
Individua ricercatore
[Ricercatore libero]
[FALSE]
[FALSE]
[TRUE]
Assegna al Team
[Team Completo]
[TRUE]

Formazione team



Impiegati (TDE)

Si consideri l'insieme di impiegati di un'azienda. Gli impiegati espongono tre metodi String getName() e String getOffice() che ritornano nome e ufficio degli impiegati e String getDescrizione() che ritorna le mansioni dell'impiegato.
Ci sono vari tipi di impiegato, per esempio gli ingegneri. Le responsabilità degli impiegati (si considerino per semplicità solo gli ingegneri) possono cambiare dinamicamente. In particolare, un ingegnere può avere la responsabilità di manager amministrativo o manager di progetto. Il comportamento del metodo String getDescrizione() viene modificato opportunamente.

Impiegati (TDE)

Per esempio, se un ingegnere *ing1* è manager amministrativo di un'area A la stringa "Manager area: A" viene concatenata alla descrizione ritornata dall'invocazione del metodo getDescrizione(). Se a *ing1* viene anche aggiunta la funzionalità di manager amministrativo dell'area B,, il metodo getDescrizione() invocato su *ing1* concatena la stringa "Manager area: B" alla descrizione di *ing1*.
Infine se a *ing1* viene aggiunta la funzionalità manager del progetto P1, la stringa ottenuta invocando il metodo getDescrizione() sull'oggetto *ing1* sarà concatenata la stringa "Manager di Progetto P1" a ciò che ritornava getDescrizione().
Come evidenziato dagli esempi, un ingegnere può essere manager amministrativo di più aree e/o manager di più progetti.

Impiegati (TDE)

1. Si modelli, attraverso un diagramma delle classi UML e un design pattern opportuno, una soluzione al problema sopra presentato.

2. Si scriva anche la struttura del codice Java risultante. Ovvero, si definiscano le classi identificate al passo precedente, le loro relazioni e le intestazioni dei metodi principali.

3. Si scriva il codice Java che crea un'istanza di un oggetto ingegnere con funzionalità di manager amministrativo per l'area A e per l'area B e project manager del progetto P1.

Impiegati (TDE)

È possibile usare il pattern Decorator