SAPIENZA Università di Roma

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Esercitazioni di

PROGETTAZIONE DEL SOFTWARE

(Corso di Laurea in Ingegneria Informatica ed Automatica

Corso di Laurea in Ingegneria dei Sistemi Informatici

A.A. 2010/11

Soluzione dell'esercitazione basata sul compito d'esame del 26 febbraio 2010

Requisiti

L'applicazione da progettare riguarda la gestione di squadre e giocatori di una lega professionistica. I giocatori sono caratterizzati da un nome (una stringa) ed un anno di nascita (un intero). Le squadre sono caratterizzate da un nome (una stringa). In una squadra giocano almeno 15 giocatori. Tra i giocatori che giocano in una squadra, uno gioca nel ruolo di capitano. Delle squadre alcune sono neopromosse e in esse giocano esattamente 15 giocatori (vincolo non esprimibile). Le squadre si incontrano tra di loro in partite, di cui interessa conoscere quale squadra gioca in casa e quale in trasferta, ed il risultato (due interi: uno per la squadra di casa, ed uno per quella in trasferta). Non possono esserci due partite in cui le stesse due squadre giocano con gli stessi ruoli in entrambe. Data una squadra è d'interesse conoscere le squadre con cui si è incontrata sia in casa che in trasferta.

Requisiti (cont.)

Siamo interessati a progettare la seguente attività: ripetutamente fino a quando l'utente lo richiede, l'utente tramite un'attività di I/O indica una partita restituendo il link corrispondente. Quindi concorrentemente si procede con le seguenti sottoattività: (i) si chiede tramite una operazione di I/O se operare sulla squadra in trasferta o sulla squadra di casa e si calcola la media delle età dei giocatori della squadra indicata, dopodichè si chiede all'utente se vuole operare con l'altra squadra e, in caso affermativo, si calcola la media delle età dei giocatori anche per questa; (ii) si calcola il numero delle partite vinte (escluse quella indicata) dalla squadra in trasferta e il numero delle partire vinte (esclusa quella indicata) dalla squadra in casa. Una volta completate entrambe queste sottoattività, si produce una pagina html con il report che mostra le informazioni calcolate.

Requisiti (cont.)

Domanda 1. Basandosi sui requisiti riportati sopra, effettuare la fase di analisi producendo lo schema concettuale in UML per l'applicazione, comprensivo del diagramma delle classi, diagramma delle attività, specifica delle attività atomiche che operano sul diagramma delle classi (i task), motivando, qualora ce ne fosse bisogno, le scelte effettuate. La specifica delle attività di I/O non è richiesta.

Domanda 2. Effettuare la fase di progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte effettuate. È richiesto di definire solo le responsabilità su tutte le associazioni del diagramma delle classi ed il progetto dell'algoritmo dell'attività atomica (task) di calcolo della media delle età dei giocatori di una squadra.

Domanda 3. Effettuare la fase di realizzazione, producendo un programma Java e motivando, qualora ce ne fosse bisogno, le scelte effettuate. È richiesto di realizzare in Java solo i seguenti aspetti dello schema concettuale:

- La classe Squadra e tutte le associazioni a cui partecipa (ignorando i vincoli di subset sulle associazioni a cui partecipa).

- L'attività principale, le eventuali sottoattività non atomiche, l'attività atomica (task) di calcolo della media delle età dei giocatori di una squadra –si assuma che l'anno corrente sia 2010. Le altre sottoattività non vanno realizzate.

Fase di analisi

Diagramma delle classi

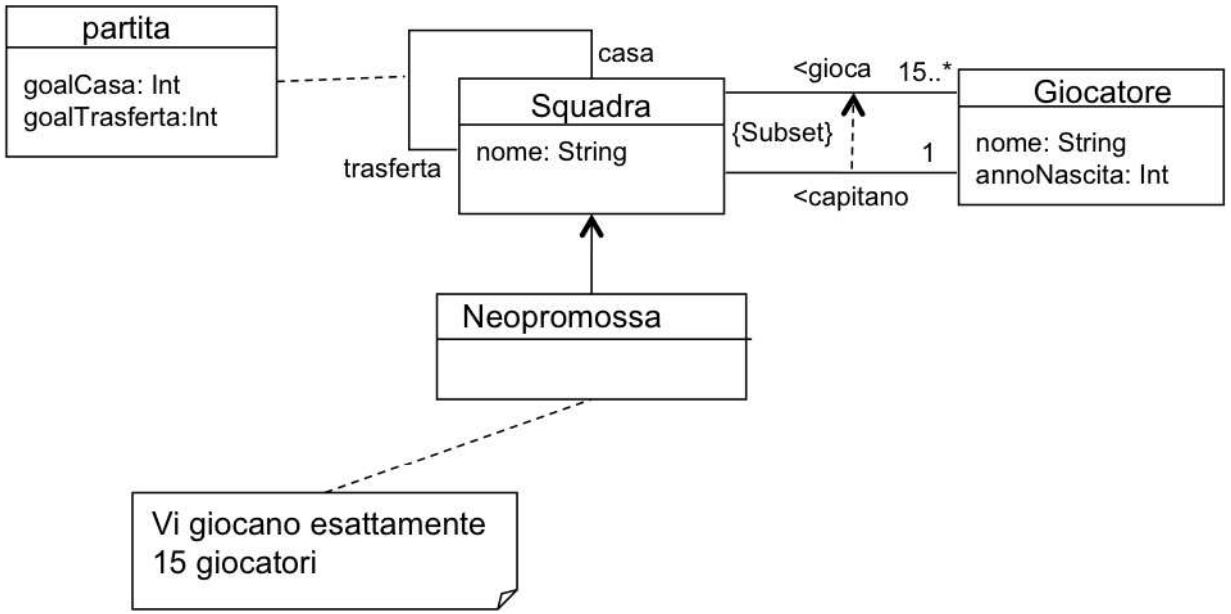
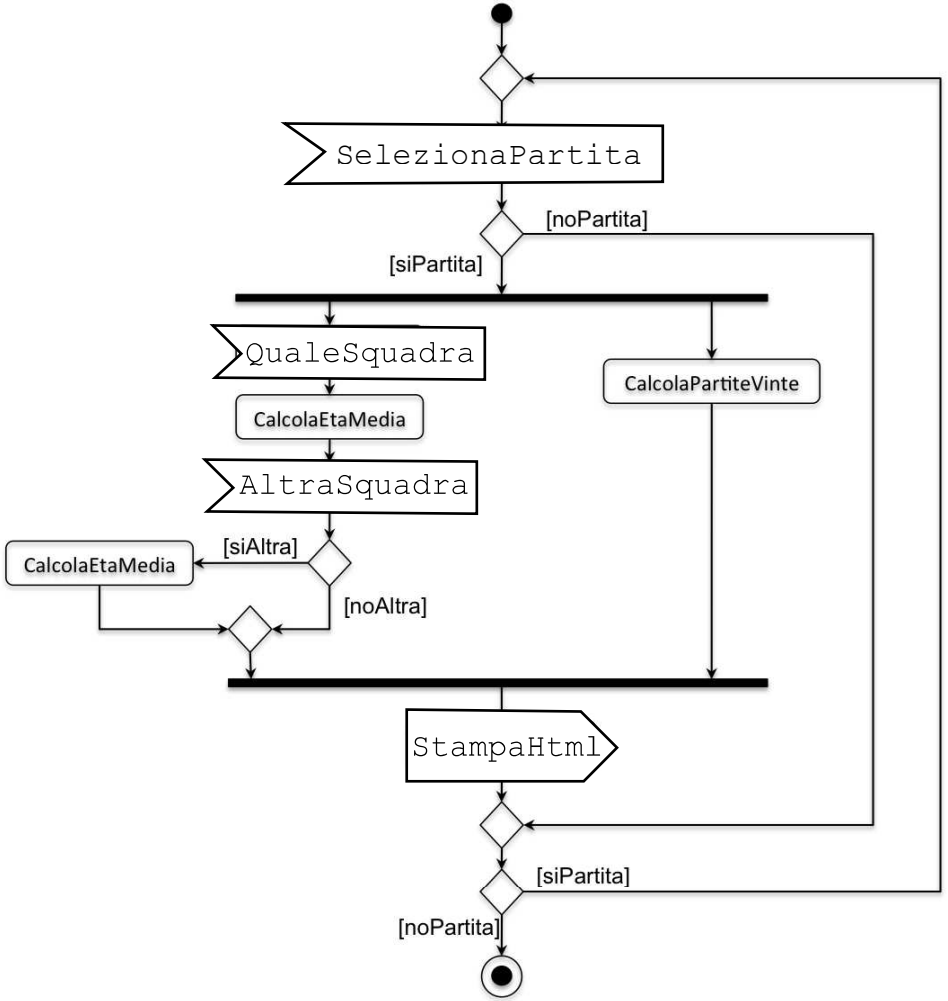


Diagramma delle attività



Specifica delle classi

Non sono presenti operazioni di classe.

Specifica delle attività

Attività di I/O

InizioSpecificaAttivitàAtomica LeggiPartita
LeggiPartita () : (Partita)
pre: –
post: mostra la lista di tutte le partite,
se l'utente seleziona una partita, `result` è la partita selezionata
se l'utente non seleziona alcuna partita, `result=null`
FineSpecifica

InizioSpecificaAttivitàAtomica QualeSquadra
QualeSquadra (p:partita) : (Squadra)
pre: –
post: mostra all'utente le opzioni *casa* e *trasferta*
se l'utente sceglie *casa*, `result` è la squadra con ruolo *casa* in p
se l'utente sceglie *trasferta*, `result` è la squadra con ruolo *trasferta* in p
FineSpecifica

Specifica delle attività (cont.)

InizioSpecificaAttivitàAtomica AltraSquadra
    AltraSquadra () : (Bool)
        pre: –
        post: mostra all'utente le opzioni *sì* e *no*
            se l'utente sceglie *sì*, `result=true`
            se l'utente sceglie *no*, `result=false`
FineSpecifica

InizioSpecificaAttivitàAtomica StampaHTML
    StampaHTML (vinteCasa : Int, vinteTrasferta : Int,
                mediaEtaCasa: Real, mediaEtaTrasferta: Real)
        pre: mediaEtaCasa $> 0 \vee$ mediaEtaTrasferta > 0
        post: mostra le informazioni in HTML
            se mediaEtaCasa$<= 0$ non mostra questa informazione
            se mediaEtaTrasferta$<= 0$ non mostra questa informazione
FineSpecifica

Specifica delle attività (cont.)

Attività atomiche (task)

InizioSpecificaAttivitàAtomica CalcolaEtaMedia

CalcolaEtaMedia (s:Squadra) : (Real)

pre: –

post: sia $G = \{g \in Giocatore \mid \langle g, s\rangle \in gioca\}$

$$\texttt{result} = \frac{\sum_{g \in G} (2010 - g.annoNascita)}{|G|}$$

FineSpecifica

Specifica delle attività (cont.)

InizioSpecificaAttivitàAtomica CalcolaPartiteVinte

CalcolaPartiteVinte (p:partita) : (vinteCasa : Int, vinteTrasferta : Int)

pre: –

post:

sia c la squadra che partecipa al link p con ruolo *casa*, e siano:

$P_c = \{q \in partita \setminus \{p\} \mid q.casa = c \vee q.trasferta = c\}$ (partite diverse da p giocate da c)

$V_c^c = \{q \in P_c \mid q.casa = c \wedge q.goalCasa > q.goalTrasferta\}$ (partite in P_c vinte da c, in casa)

$V_c^t = \{q \in P_c \mid q.trasferta = c \wedge q.goalTrasferta > q.goalCasa\}$
(partite in P_c vinte da c, in trasferta)

inoltre, sia t la squadra che partecipa al link p con ruolo *trasferta*, e siano:

$P_t = \{q \in partita \setminus \{p\} \mid q.casa = t \vee q.trasferta = t\}$ (partite diverse da p giocate da t)

$V_t^c = \{q \in P_t \mid q.casa = t \wedge q.goalCasa > q.goalTrasferta\}$ (partite in P_t vinte da t, in casa)

$V_t^t = \{q \in P_t \mid q.trasferta = t \wedge q.goalTrasferta > q.goalCasa\}$
(partite in P_t vinte da t, in trasferta)

allora:

vinteCasa= $|V_c^c \cup V_c^t|$

vinteTrasferta= $|V_t^c \cup V_t^t|$

FineSpecifica

Specifica delle attività (cont.)

Attività complesse

InizioSpecificaAttività sottoramoSx

```
    sottoramoSx (p:partita) : (mediaEtaCasa: Real, mediaEtaTrasferta: Real)

    Variabili Processo
        squadraSelezionata : Squadra
        altra : Bool

    InizioProcesso
        mediaEtaCasa = -1;
        mediaEtaTrasferta = -1;
        QualeSquadra(p) : (squadraSelezionata);
        if (p.casa == squadraSelezionata)
            CalcolaEtaMedia(squadraSelezionata) : (mediaEtaCasa);
        else
            CalcolaEtaMedia(squadraSelezionata) : (mediaEtaTrasferta);
        AltraSquadra() : (altra);
        if (altra){
            if (p.casa == squadraSelezionata)
                CalcolaEtaMedia(p.fuori) : mediaEtaTrasferta;
            else
                CalcolaEtaMedia(p.casa) : mediaEtaCasa;
        }

    FineProcesso
FineSpecifica
```

Specifica delle attività (cont.)

InizioSpecificaAttività sottoramoDx

sottoramoDx (p:partita) : (vinteCasa: Int, vinteTrasferta: Int)

Variabili Processo
–

InizioProcesso
CalcolaPartiteVinte(p) : (vinteCasa,vinteTrasferta);
FineProcesso

FineSpecifica

Specifica delle attività (cont.)

```
InizioSpecificaAttività attivitaPrincipale
    attivitaPrincipale () : ()

    Variabili Processo
        partitaSelezionata : Partita
        mediaEtaCasa : Real
        mediaEtaTrasferta : Real
        vinteCasa : Int
        vinteTrasferta : Int

    InizioProcesso
        do{
            LeggiPartita():(partitaSelezionata);
            if(partitaSelezionata != null){
                fork {
                    thread t1 : sottoramoSx(partitaSelezionata):(mediaEtaCasa,mediaEtaTrasferta);
                    thread t2 : sottoramoDx(partitaSelezionata):(vinteCasa,vinteTrasferta);
                }
                join t1, t2;
                StampaHTML (vinteCasa, vinteTrasferta, mediaEtaCasa, mediaEtaTrasferta);
            }
        } while(partitaSelezionata != null)
    FineProcesso
FineSpecifica
```

Fase di progetto

Responsabilità sulle associazioni

La seguente tabella delle responsabilità si evince da:

- 1. requisiti
- 2. operazioni di classe ed attività
- 3. vincoli di molteplicità nel diagramma delle classi

Associazione	Classe (o ruolo)	ha resp.
partita	*casa*	Sì[1,2]
	trasferta	Sì[1,2]
gioca	*Squadra*	Sì[3]
	Giocatore	NO
capitano	*Squadra*	Sì[3]
	Giocatore	NO

Strutture di dati

Abbiamo la necessità di rappresentare collezioni omogenee di oggetti, a causa:

- dei vincoli di molteplicità 15..* delle associazioni,
- delle variabili necessarie per vari algoritmi.

Per fare ciò, utilizzeremo le classi del Java Collection Framework : `Set`, `HashSet`.

Corrispondenza fra tipi UML e Java

Riassumiamo le nostre scelte nella seguente tabella di corrispondenza dei tipi UML.

Tipo UML	Rappresentazione in Java
Int	`int`
Real	`float`
String	`String`
boolean	`boolean`
Insieme	`HashSet`

Tabelle di gestione delle proprietà di classi UML

Riassumiamo le nostre scelte differenti da quelle di default mediante la *tabella delle proprietà immutabili* e la *tabella delle assunzioni sulla nascita*.

Classe UML	**Proprietà immutabile**
Squadra	*nome*
Giocatore	*nome*
	annoNascita

Classe UML	**Proprietà**	
	nota alla nascita	**non nota alla nascita**
Squadra	–	*capitano*

Altre considerazioni

Molteplicità: Dobbiamo assicurare che le istanze di Neopromossa contengano esattamente 15 giocatori.

Sequenza di nascita degli oggetti: Non dobbiamo assumere una particolare sequenza di nascita degli oggetti.

Valori alla nascita: Non sembra ragionevole assumere che per qualche proprietà esistano valori di default validi per tutti gli oggetti.