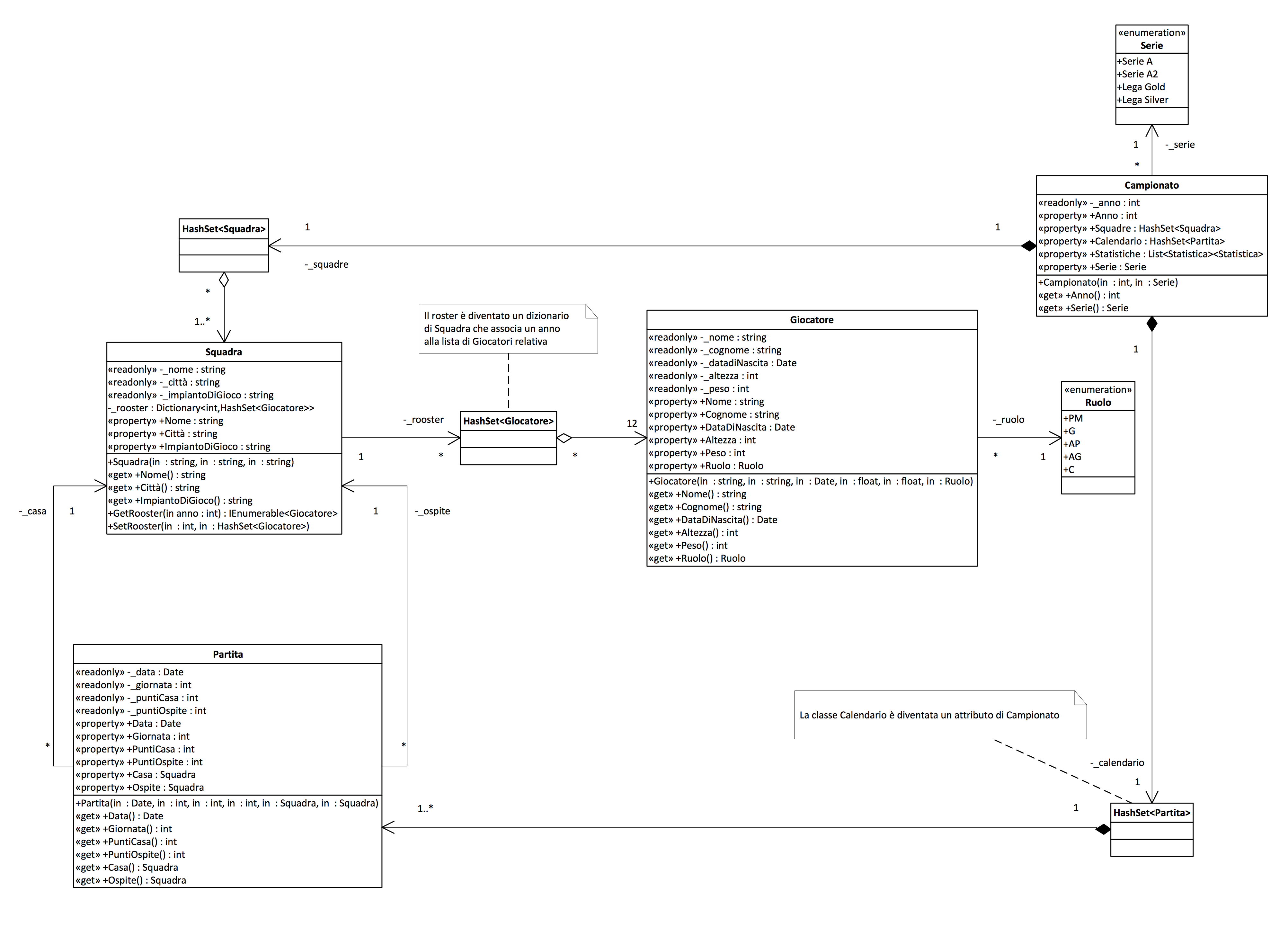
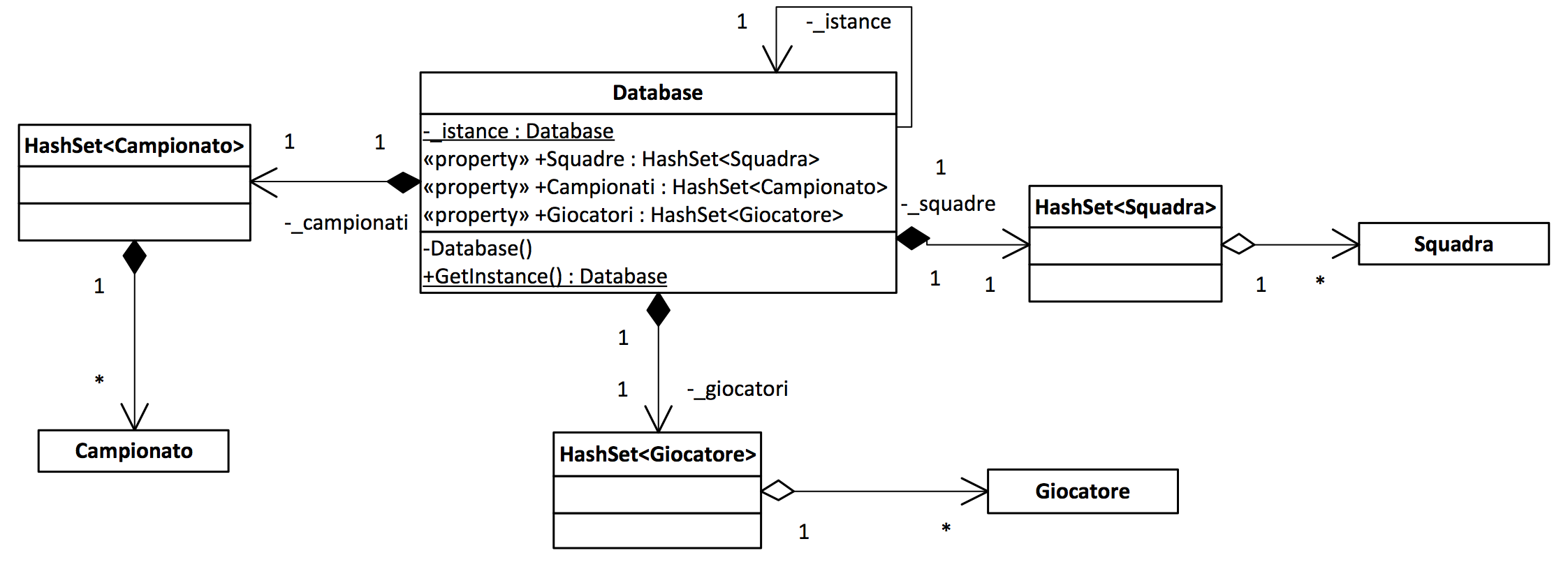
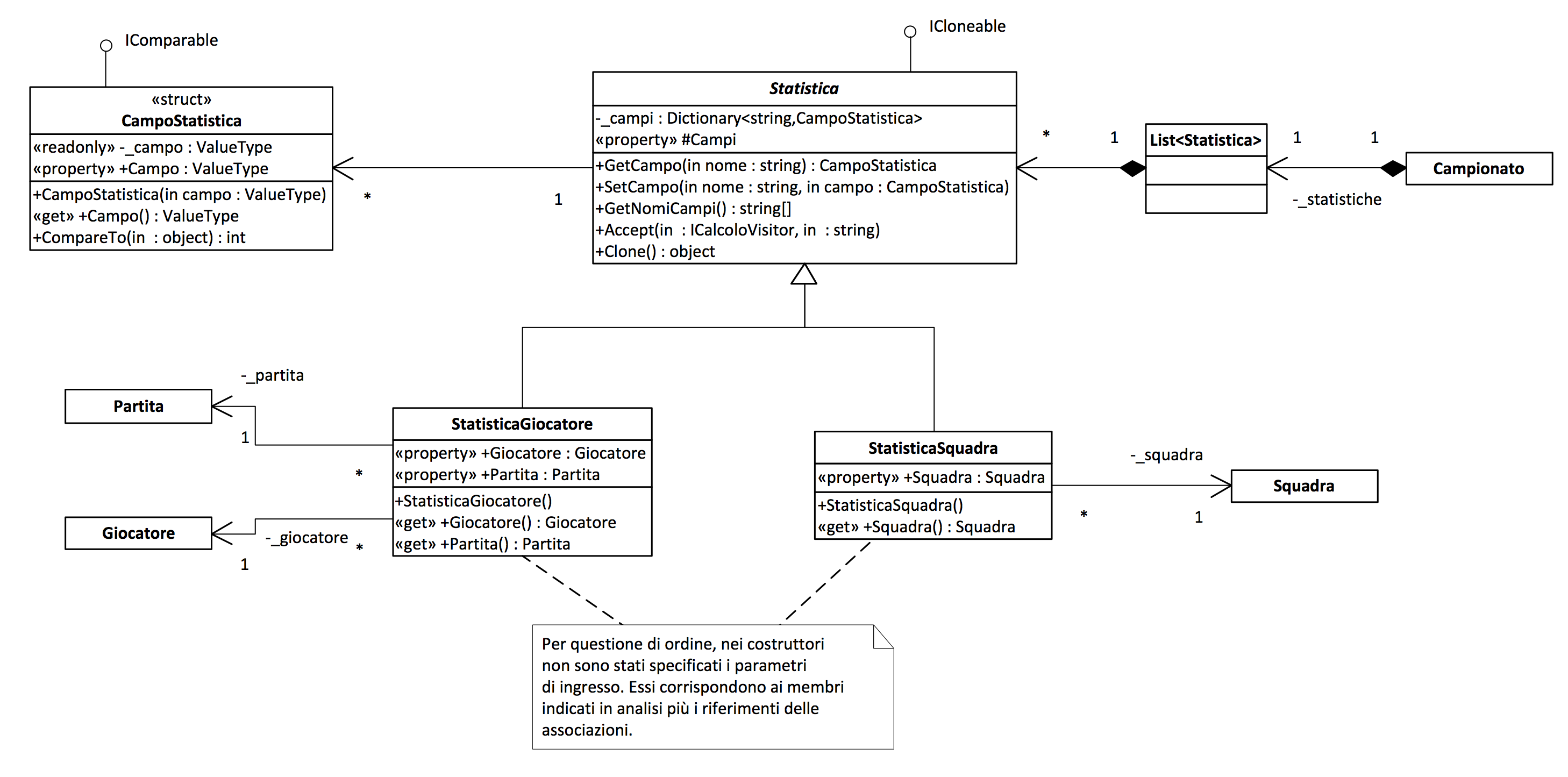
**DIAGRAMMI DELLE CLASSI DI PROGETTAZIONE**

****

****

Il Singleton Database mantiene in memoria i dati utilizzati durante l’esecuzione dell’applicazione. All’avvio del programma, vengono caricati e parsificati i documenti XML che contengono i dati immutabili.

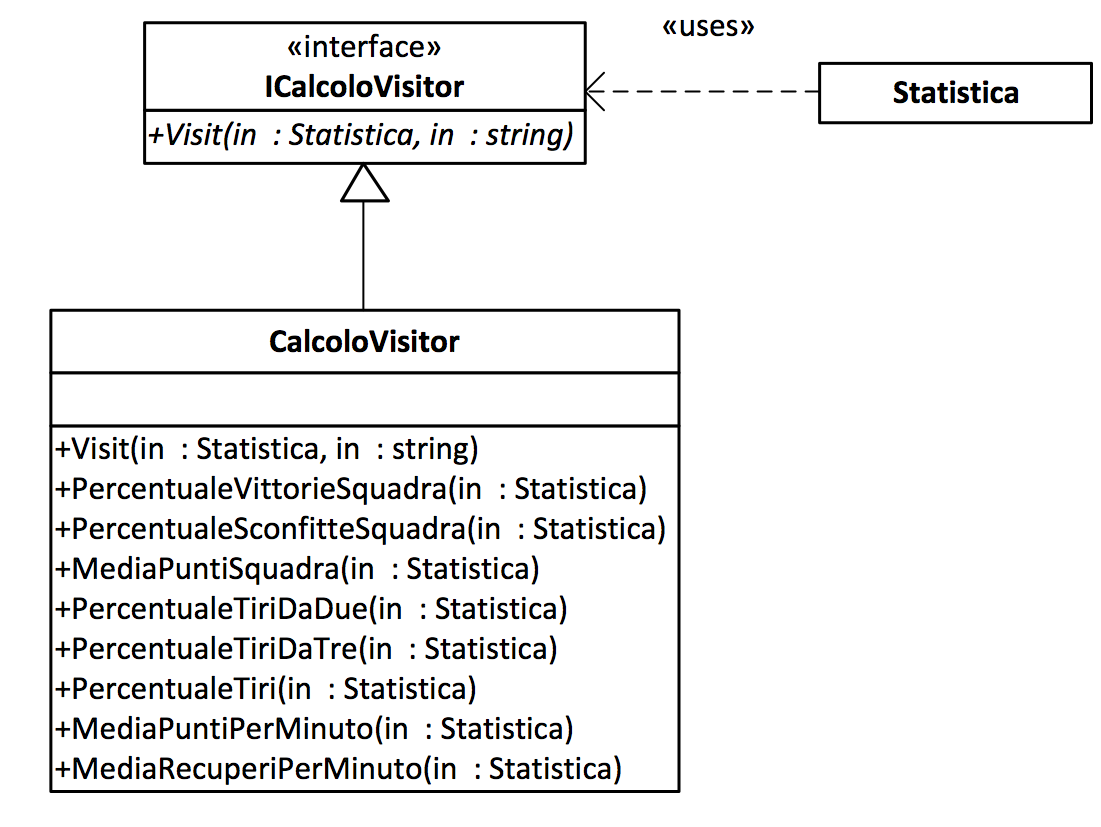
Squadre, giocatori e campionati sono salvati in collezioni contenute direttamente nel Database; le statistiche e le partite vengono poi inserite nei rispettivi campionati.

****

Le statistiche di squadra e di giocatore vengono uniformate grazie alla classe astratta Statistica. Dato che l’insieme dei campi che costituiscono una statistica è un requisito molto instabile, si è scelto di implementare Statistica come un contenitore di strutture CampoStatistica, le quali racchiudono un qualsiasi ValueType che sia comparabile. L’accesso ai campi della statistica è garantito da un dizionario che associa il nome del campo al corrispondente CampoStatistica.

Questa architettura permette un utilizzo uniforme di StatisticaGiocatore e StatisticaSquadra ed è un’applicazione del principio Open/Closed: le statistiche sono facilmente estensibili aggiungendo ulteriori campi al dizionario senza modifiche al codice già presente.

Dato che gli oggetti Statistica possono essere modificate durante l’esecuzione dell’applicazione (ad esempio durante un calcolo), il metodo Clone dell’interfaccia ICloneable permette di effettuare una deep copy del dizionario di CampoStatistica per non modificare le statistiche persistenti presenti all’interno di Campionato.

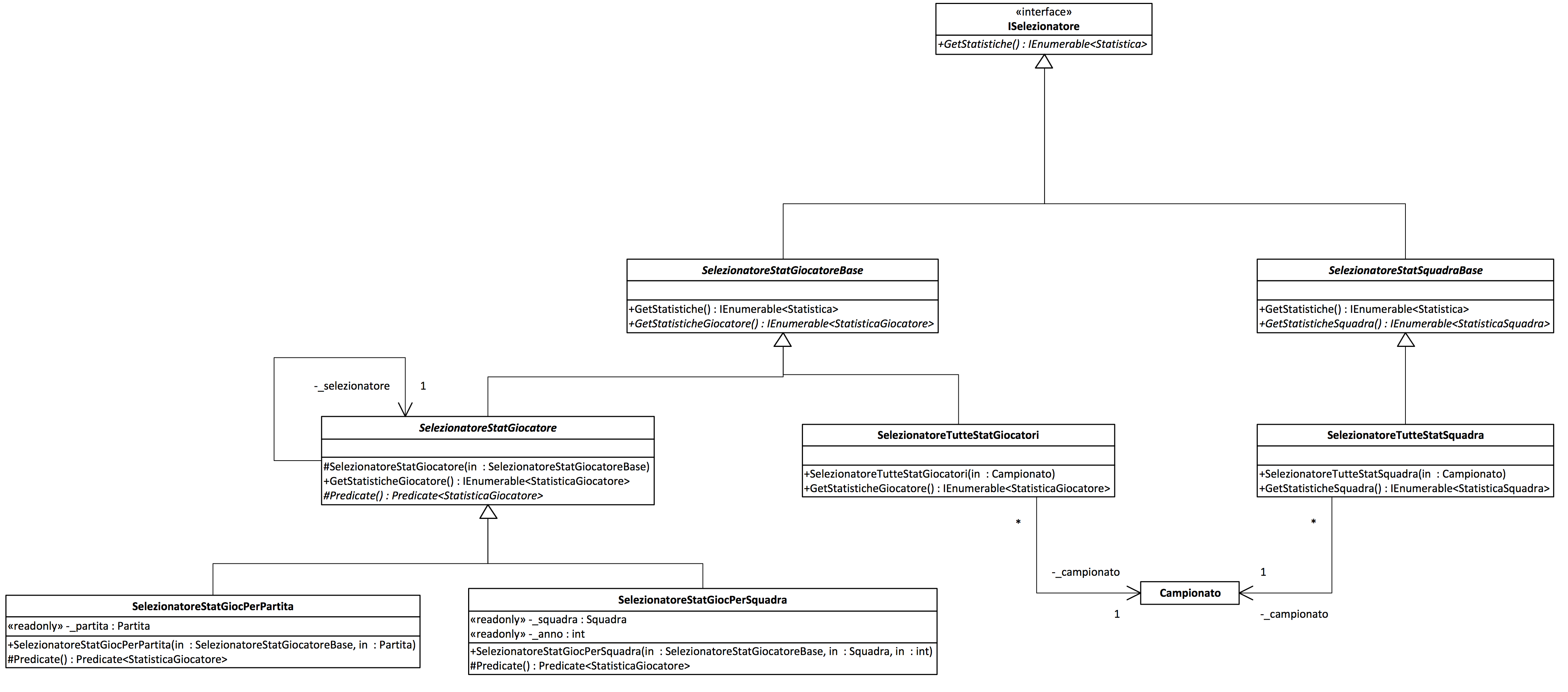


I calcoli effettuabili sulle statistiche sono stati raccolti nella classe CalcoloVisitor. La Statistica invoca Accept passando come argomento il nome del calcolo che deve essere effettuato e il visitor si occupa di modificare lo stato interno della Statistica selezionando i campi su cui effettuare l’operazione e inserendo il risultato in un nuovo CampoStatistica.

Così come per i campi delle statistiche, anche l’insieme dei calcoli effettuabili è un requisito molto instabile: per questo motivo si è optato per la scelta del pattern Visitor, il quale permette di separare nettamente la responsabilità di mantenimento dello stato della classe Statistica da quella di effettuare operazioni sui suoi campi; inoltre l’insieme dei calcoli è facilmente estensibile implementando una sottoclasse di CalcoloVisitor.

Per effettuare il popolamento dei menù della GUI che contengono l’elenco dei calcoli disponibili, si fa uso di un attributo custom che marca ogni metodo del visitor. L’attributo indica a quali campi delle statistiche si applica il calcolo e permette, tramite Reflection, di popolare i menù relativi. Per questa ragione i metodi devono essere pubblici, ma grazie all’interfaccia ICalcoloVisitor essi sono nascosti e non invocabili direttamente.

Anche il metodo Visit fa uso di Reflection per effettuare il dispatch del metodo: utilizzando i metadati della propria classe, cerca il metodo che corrisponde alla stringa passata in ingresso e lo invoca dinamicamente.



Per effettuare le selezioni è necessario applicare dei criteri relativi ai riferimenti delle particolari statistiche: non è dunque possibile operare sull’oggetto Statistica, ma è il selezionatore deve conoscere se si tratta di una StatisticaGiocatore o una StatisticaSquadra. Il risultato della selezione, però, può essere trattato come una generica collezione di Statistica.

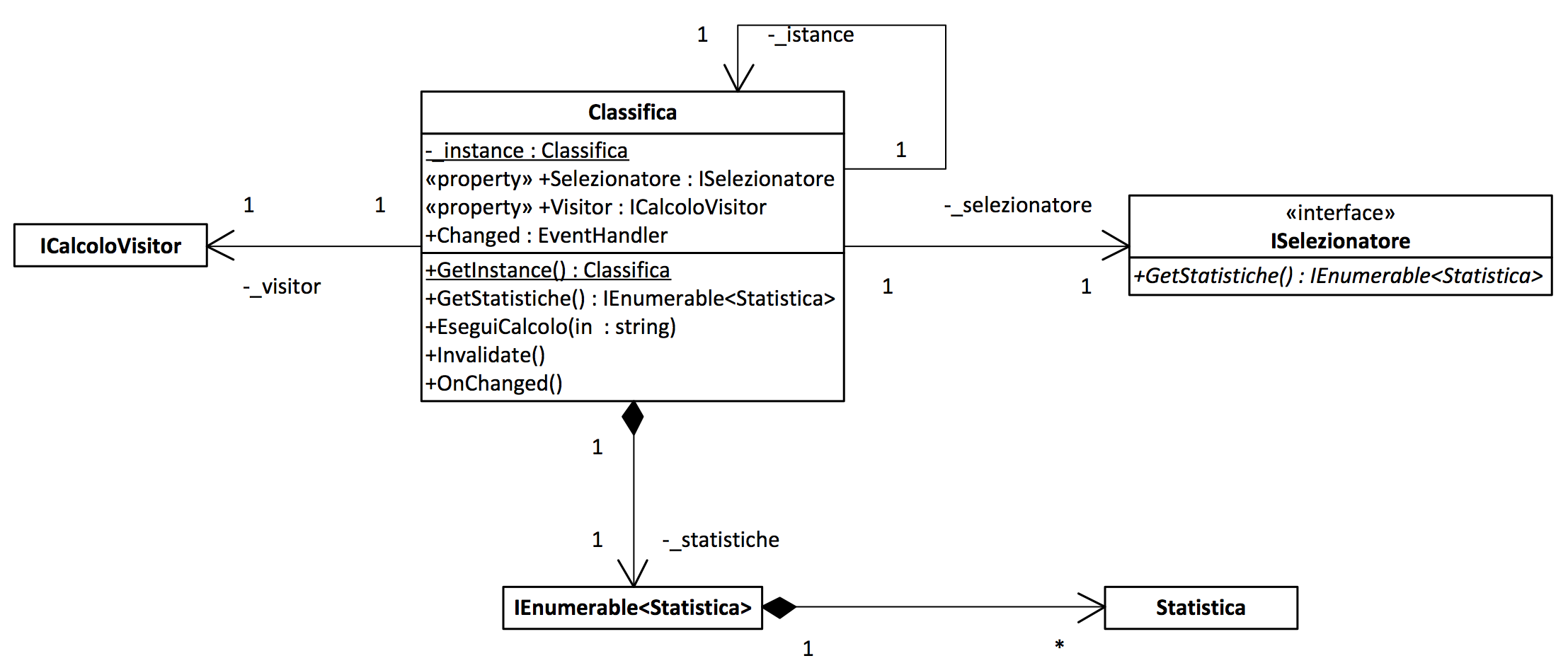
Per questi motivi, si è fatto uso del pattern Strategy per nascondere dietro all’interfaccia ISelezionatore le due diverse strategie di selezione, che si concretizzano come due gerarchie simmetriche.

Per quanto riguarda le statistiche dei giocatori, si è sfruttato il pattern Decorator per applicare i criteri di selezione come dei filtri sul risultato di SelezionatoreTutteStatGiocatori, il quale estrae dalla collezione di Statistica del Campionato in esame gli oggetti di tipo StatisticaGiocatore.

Per le statistiche di squadra non sono presenti nei requisiti dei particolari criteri di selezione, oltre a quello obbligatorio del Campionato. Si è comunque scelta un’implementazione speculare a quella dei selezionatori delle StatisticaGiocatore per aprire a un’eventuale estensione della StatisticaSquadra e a una conseguente introduzione di criteri di selezione.

Le selezioni nascondono un’operazione di “normalizzazione”: la collezione di Statistica restituita deve contenere una sola Statistica per ogni Giocatore o Squadra. Dunque le statistiche della selezione che si riferiscono allo stesso Giocatore vengono condensate in un’unica Statistica tramite somma dei campi. Per StatisticaSquadra il problema non si pone dato che all’interno di un Campionato è presente un solo oggetto per Squadra.

È compito di un builder effettuare la scelta della strategia di selezione, che dipende dal tipo di Statistica su cui si sta operando in quel momento, e l’istanziazione degli eventuali decoratori a seconda dei parametri passati in ingresso.



Il Singleton Classifica rappresenta il risultato della selezione corrente che viene visualizzata nell’interfaccia grafica. Al suo interno è presente la collezione degli oggetti Statistica che si sta visualizzando, copie clonate restituite dal metodo GetStatistiche dell’ISelezionatore.

Classifica è il ponte di comunicazione principale tra la GUI e il modello sottostante. Espone l’evento Changed che viene fatto scattare quando cambiano i criteri di selezione specificati dall’utente nell’interfaccia grafica e che forza l’aggiornamento della view. Inoltre è il responsabile dell’applicazione dei calcoli richiesti sui campi delle statistiche, che viene fatta passando il proprio ICalcoloVisitor al metodo Accept di ogni Statistica della selezione corrente.