Iterazione 3

1.0 Introduzione

La terza iterazione si concentrerà sui seguenti aspetti:

- Implementazione di estensioni dei casi d'uso già implementati.
- Implementazione di nuovi casi d'uso.
- Refactoring del codice e testing.

In questa terza iterazione andremo ad implementare gli scenari principali di successo dei seguenti casi d'uso:

- **UC7**: Visualizza Classifica
- **UC8**: Aggiorna ELO
- **UC9**: Aggiungi formato torneo

Essendo stato effettuato un refactoring del codice, si veda il diagramma delle classi nella sua versione aggiornata e il progetto nella directory per vedere le modifiche apportate.

1.1 Aggiornamento dei casi d'uso

Come per la prima e seconda iterazione, durante questa fase sono stati specificati e rivisti dei dettagli dei casi d'uso implementati. Per visionare le modifiche apportate ai suddetti, si consulti il documento "Modello dei casi d'uso" riportato nella cartella in versione aggiornata per questa iterazione.

2.0 Fase di Analisi

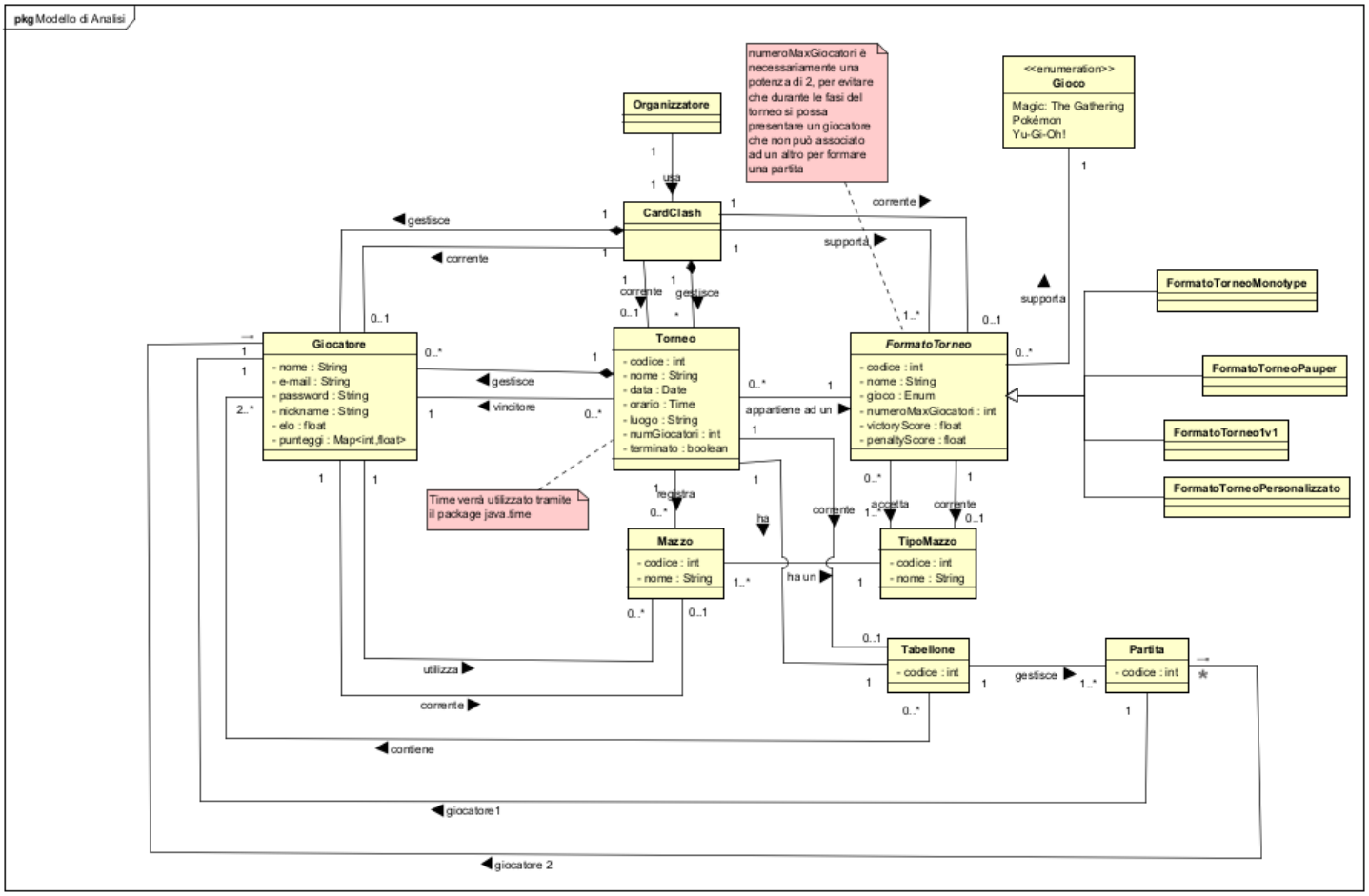
Nella fase di Analisi della terza iterazione, sono state effettuate delle variazioni alle classi già implementate nei precedenti casi d'uso. Di seguito le modifiche effettuate:

- Aggiunta l'attributo booleano "terminato" sulla classe concettuale Torneo, necessario per modellare lo stato del torneo in vista dei casi d'uso implementati. Riteniamo che l'utilizzato di questo attributo sia più che sufficiente per il corretto funzionamento dell'applicazione, non ci serve dunque applicare il pattern GoF State. In altre parole, non abbiamo la necessità di avere una transizione dinamica e indipendente degli stati che giustifichi la creazione di classi separate per ogni stato, poiché il comportamento dell'oggetto non cambia radicalmente in funzione dello stato, ma segue una logica predefinita e facilmente controllabile. Inoltre, la sua

aggiunta è stata seguita dalla modifica di alcuni SD fatti in precedenza, che in seguito saranno riportati in versione aggiornata.

- Aggiunto l'attributo "vincitore" su torneo, per permettere di salvare il giocatore vincitore del torneo.
- Modellata una nuova classe chiamata FormatoTorneoPersonalizzato, per permettere la corretta implementazione del caso d'uso UC9 per l'aggiunta di un nuovo formato torneo.

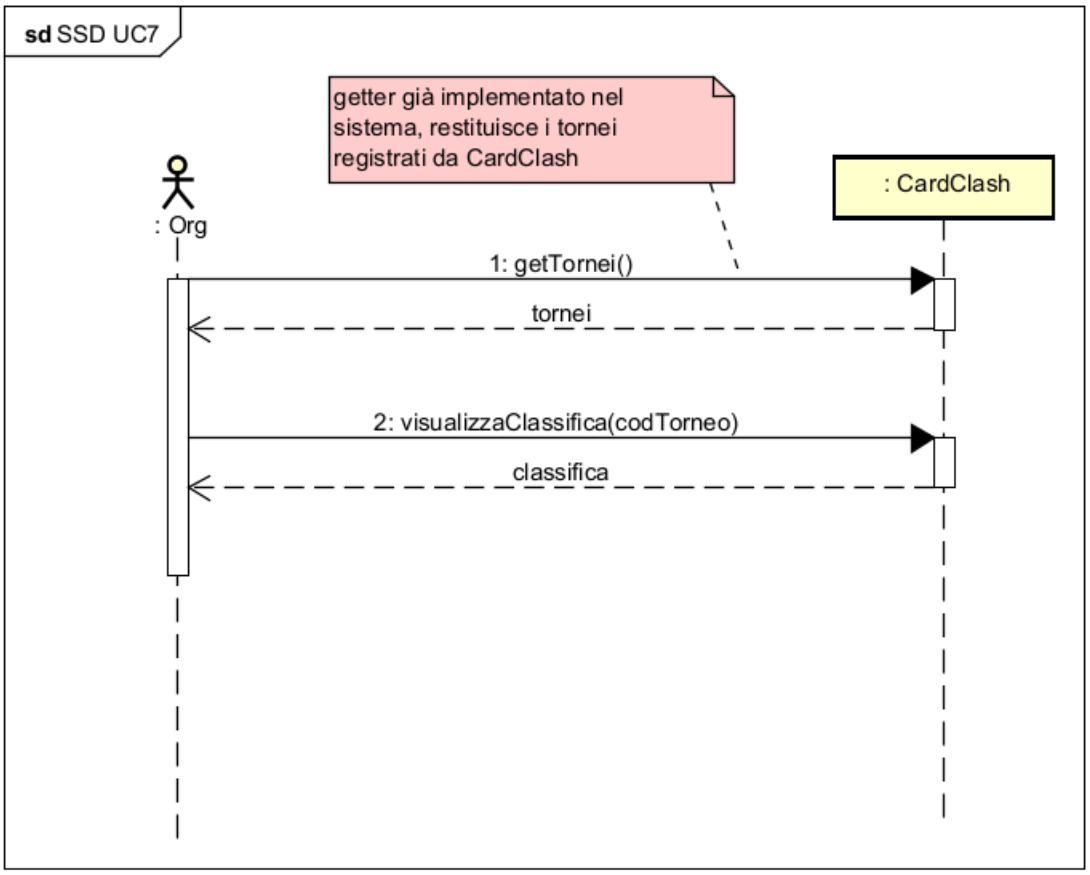
Si riporta di seguito il Modello di Dominio aggiornato:



2.1 Diagramma di sequenza di sistema

- **UC7:** Visualizza Classifica

Il seguente Diagramma di Sequenza di Sistema (SSD) illustra il corso di eventi I/O tra l'organizzatore e il sistema CardClash.



2.1.1 Contratti delle operazioni

I seguenti Contratti descrivono le principali operazioni eseguite dal sistema sulla base degli eventi individuati nell'SSD.

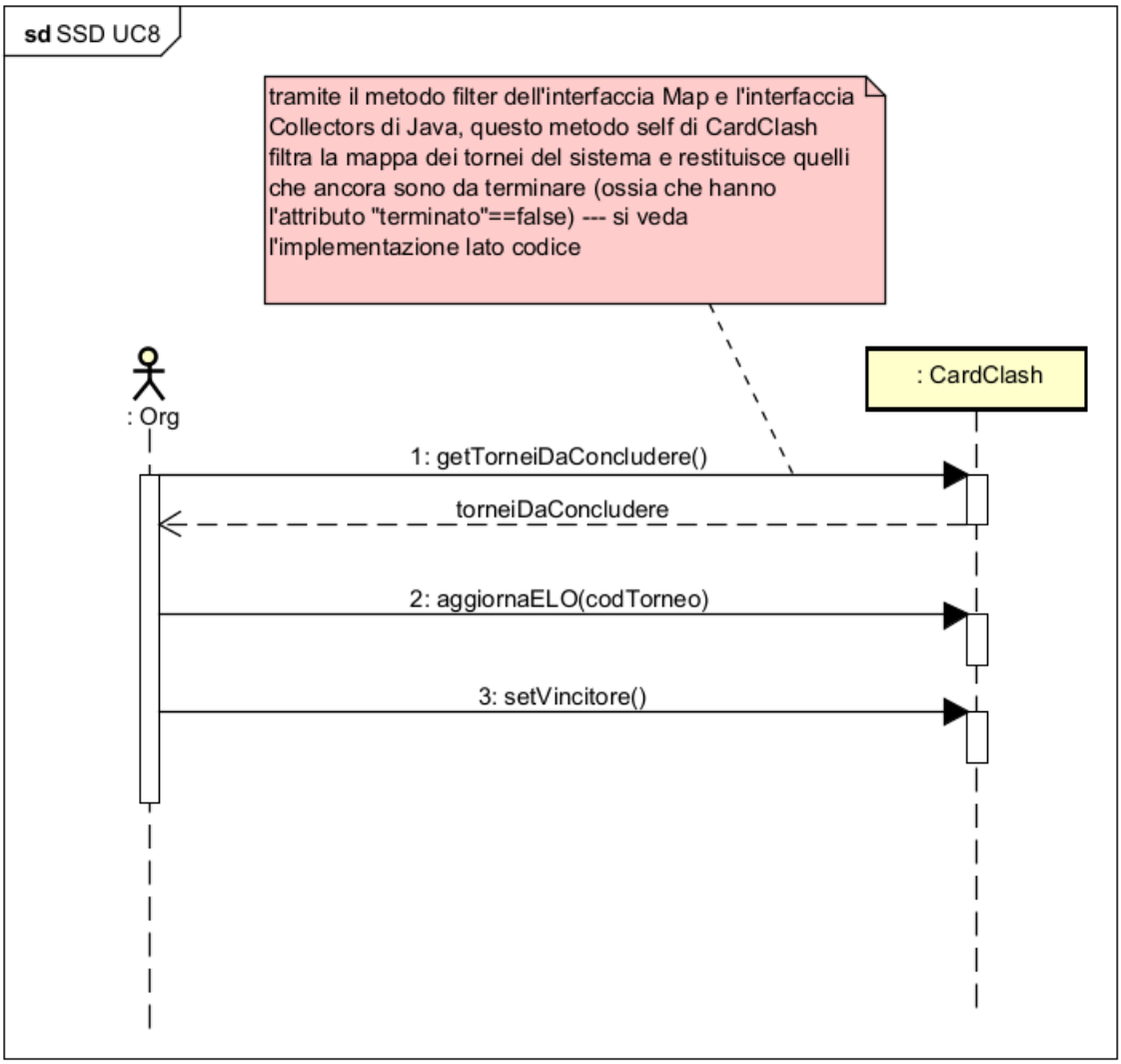
Contratto1 UC7: VIsualizza Classifica

- **Operazione:** visualizzaClassifica(codTorneo)
- **Riferimenti: Caso d'uso:** Visualizza Classifica
- **Pre-condizione:**
  - L'amministratore è autenticato nel sistema.
  - I tornei sono stati recuperati con successo
- **Post-condizione:**
  - Viene ritornata la classifica per il torneo selezionato.

2.2 Diagramma di sequenza di sistema

- **UC8:** Aggiorna ELO

Il seguente Diagramma di Sequenza di Sistema (SSD) illustra il corso di eventi I/O tra l'organizzatore e il sistema CardClash.



2.2.1 Contratti delle operazioni

I seguenti Contratti descrivono le principali operazioni eseguite dal sistema sulla base degli eventi individuati nell'SSD.

Contratto1 UC8: Aggiorna ELO

- **Operazione:** aggiornaELO(codTorneo)
- **Riferimenti: Caso d'uso:** Aggiorna ELO

- **Pre-condizione:**
- L’amministratore è autenticato nel sistema.
- I tornei da concludere sono stati recuperati con successo
- **Post-condizione:**
  - Viene recuperata la lista dei giocatori del torneo, e viene aggiornato il loro ELO sulla base del valore corrente e del punteggio ottenuto durante il torneo.

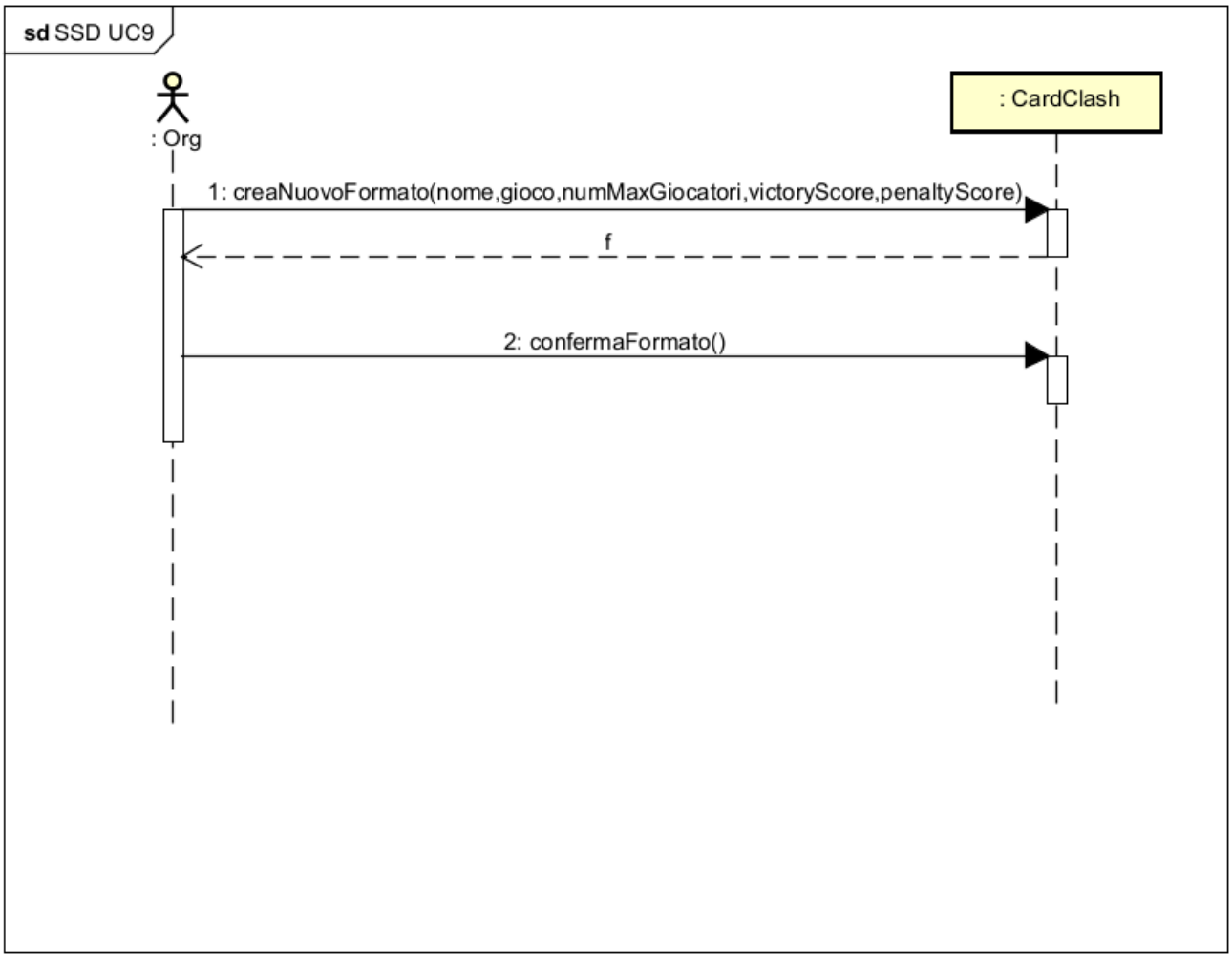
Contratto1 UC8: Aggiorna ELO

- **Operazione:** setVincitore()
- **Riferimenti: Caso d’uso:** Aggiorna ELO
- **Pre-condizione:** L’ELO dei giocatori è stato aggiornato correttamente
- **Post-condizione:**
  - Viene terminato il torneo settando true il boolean omonimo
  - Viene inizializzato il vincitore del torneo, ossia il primo giocatore in classifica

2.3 Diagramma di sequenza di sistema

- **UC9:** Aggiungi formato torneo

Il seguente Diagramma di Sequenza di Sistema (SSD) illustra il corso di eventi I/O tra l'organizzatore e il sistema CardClash.



2.3.1 Contratti delle operazioni

I seguenti Contratti descrivono le principali operazioni eseguite dal sistema sulla base degli eventi individuati nell'SSD.

Contratto1 UC9: Aggiungi formato torneo

- **Operazione:** creaNuovoFormato()
- **Riferimenti: Caso d'uso:** Aggiungi Formato Torneo
- **Pre-condizione:** L'amministatore è autenticato
- **Post-condizione:**
  - Il formato f è stato creato correttamente

- Gli attributi di f sono stati inizializzati correttamente
- f è corrente per CardClash

Contratto2 UC9: Aggiungi formato torneo

- **Operazione:** confermaFormato()
- **Riferimenti: Caso d'uso:** Aggiungi Formato Torneo
- **Pre-condizione:** gli attributi di f sono stati inizializzati correttamente, e il check dell'estensione 9b è risultato false.
- **Post-condizione:**
  - Si salva l'istanza di f all'interno della mappa dei formati.

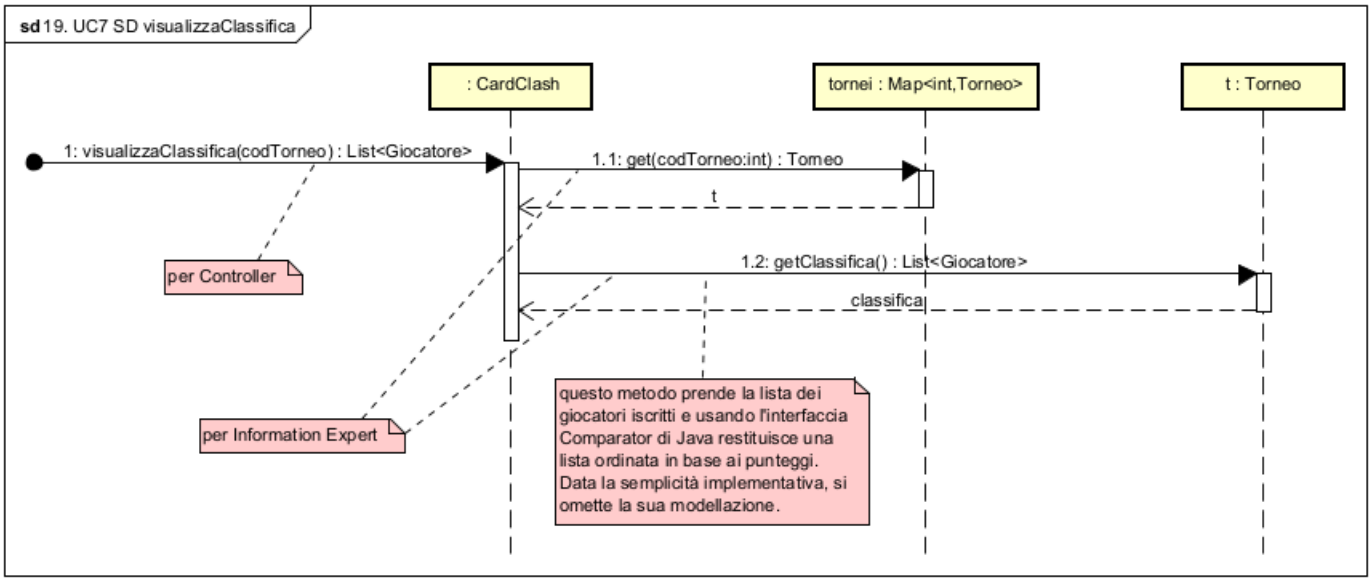
3.0 Fase di Progettazione

Di seguito vengono riportati gli elaborati prodotti. SI noti, come già citato nella fase di analisi, l'aggiunta della classe formatoTorneoPersonalizzato che risulta essere un'altra implementazione della classe astratta FormatoTorneo, sempre attenendoci al pattern GoF Template Method; questo per permettere la creazione di nuovi formati di torneo personalizzati dall'amministratore. Inoltre, sono state implementate le estensioni 9b e 9c. Infine, sono stati modificati gli SD di: mostraTorneiDisponibili e CreaTabellone per adattarsi all'attributo "terminato" su torneo, e l'SD aggiornaTabellone, per mostrare il metodo inizializzaPartita implementato da Tabellone.

3.1 Diagramma di sequenza UC7

Il seguente caso d'uso permette la visualizzazione delle classifiche dato il codice di un dato torneo.

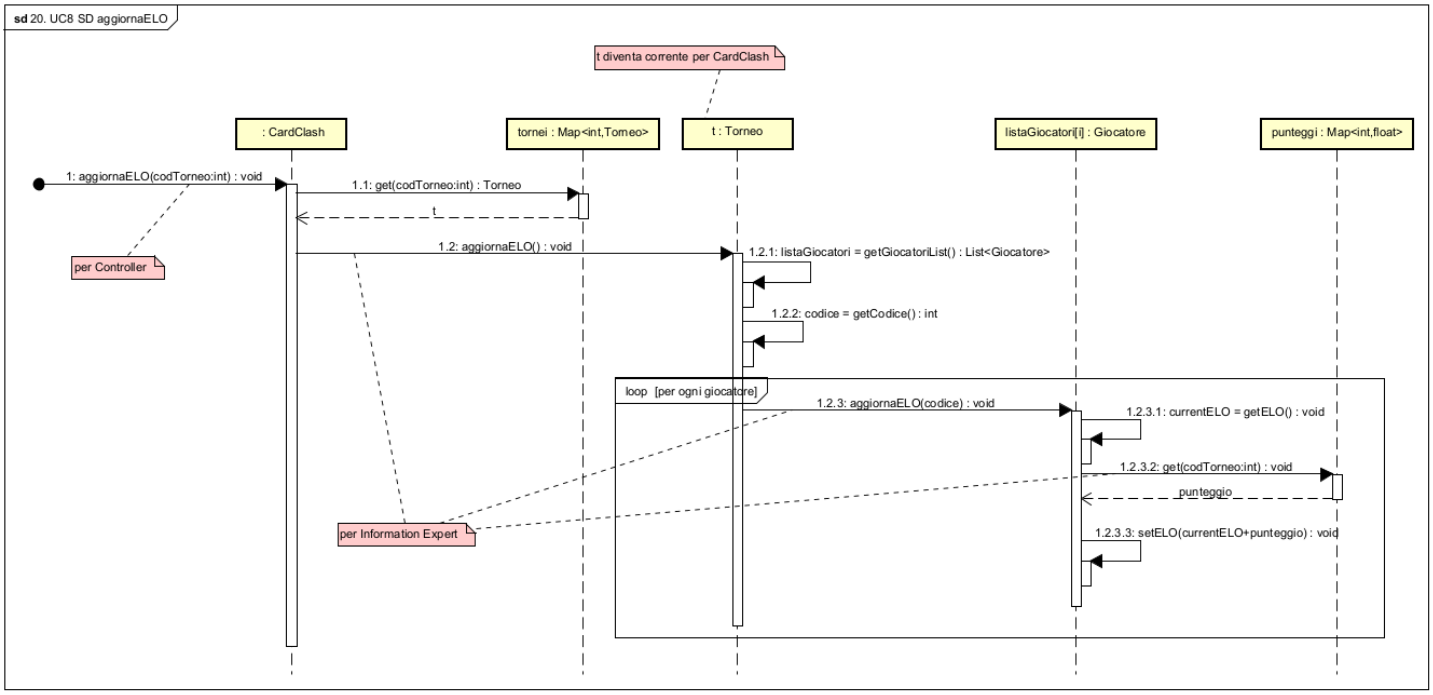
visualizzaClassifica



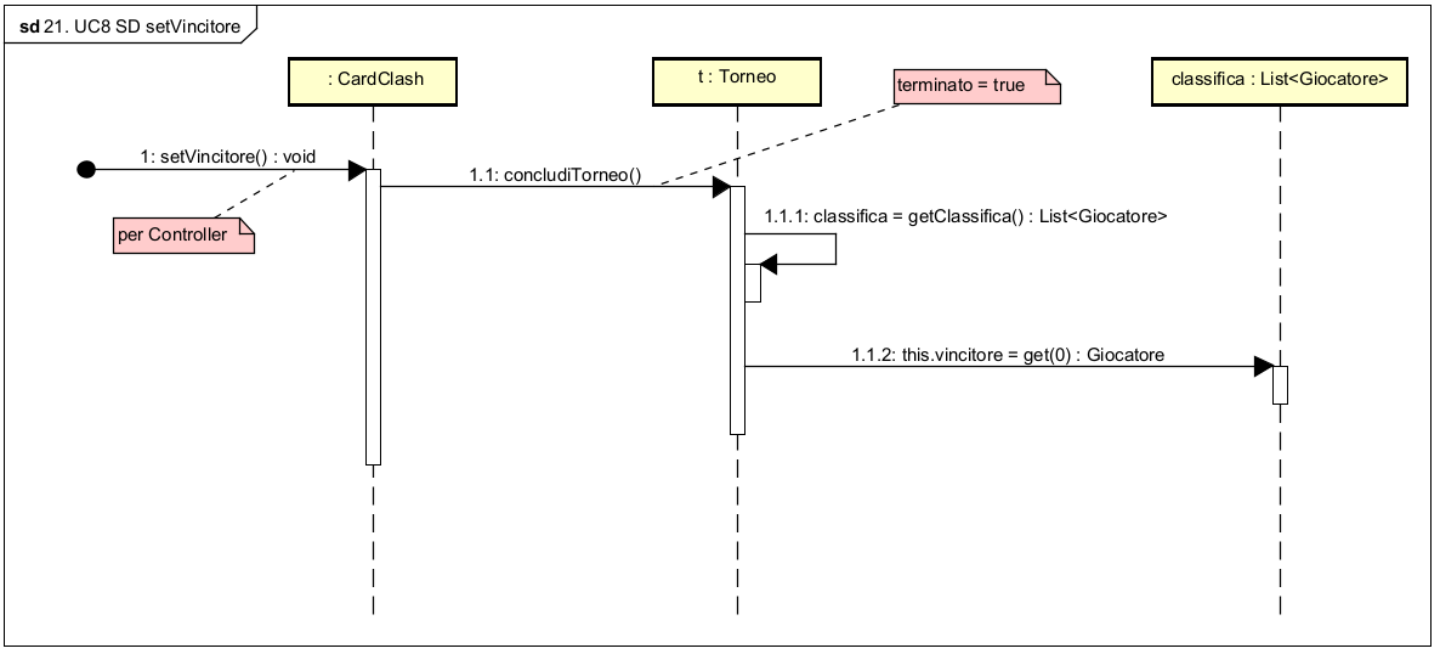
3.2 Diagramma di sequenza UC8

Questo caso d'uso consente di aggiornare l'ELO dei giocatori iscritti a un torneo, dato il codice del torneo. Inoltre, permette di assegnare un vincitore e di concludere il torneo.

aggiornaELO



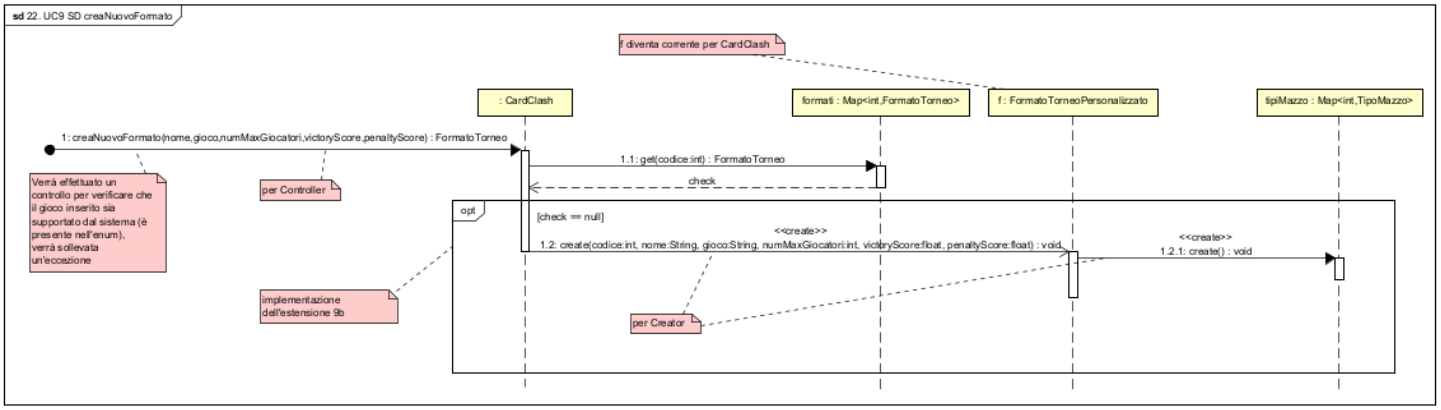
setVincitore



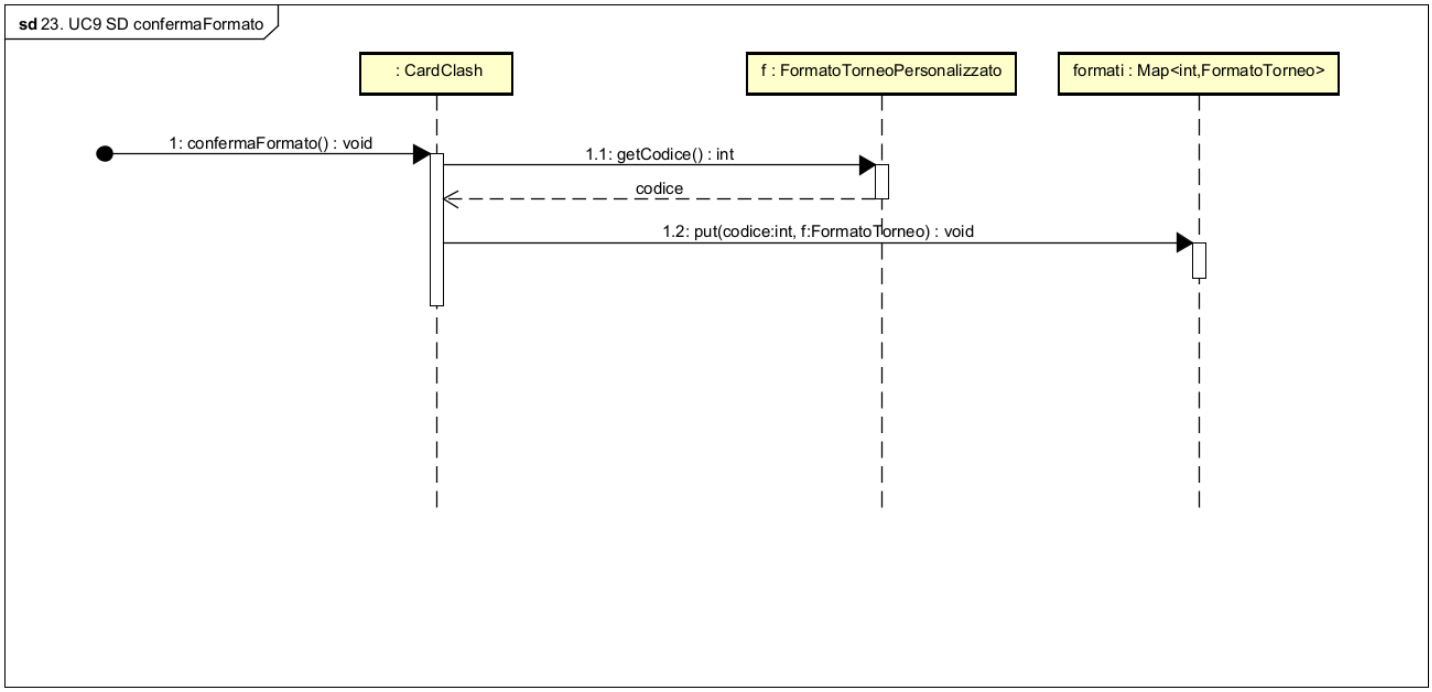
3.3 Diagramma di sequenza UC9

Questo caso d'uso permette la creazione di un formato di torneo personalizzato. Come detto in precedenza, sono state implementate le estensioni 9b e 9c.

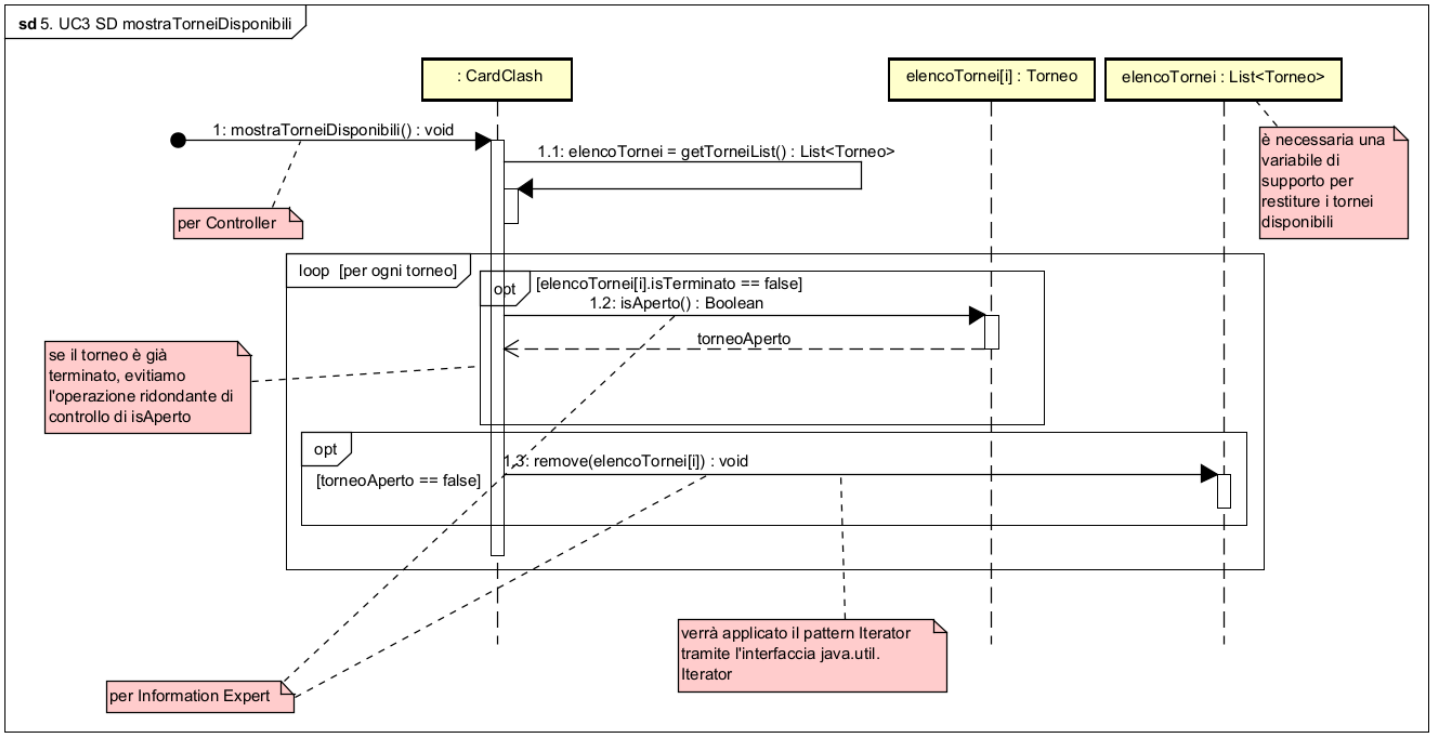
creaNuovoFormato



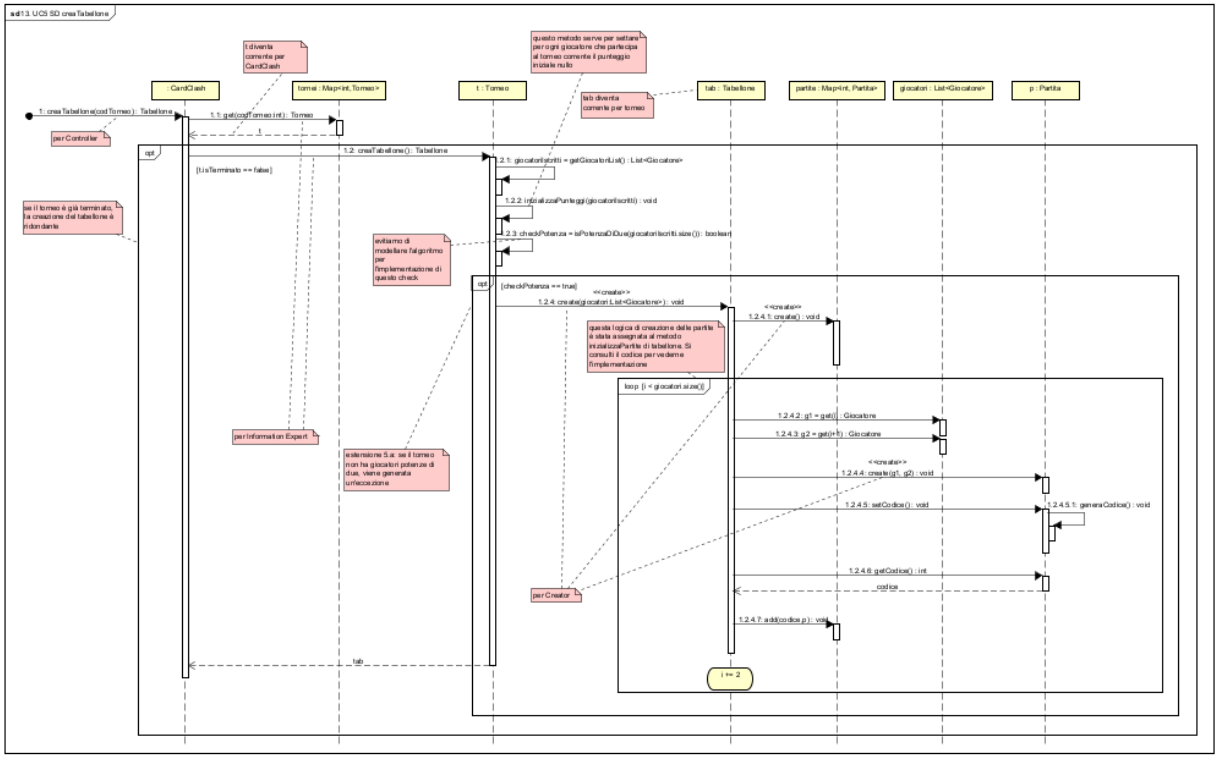
confermaFormato



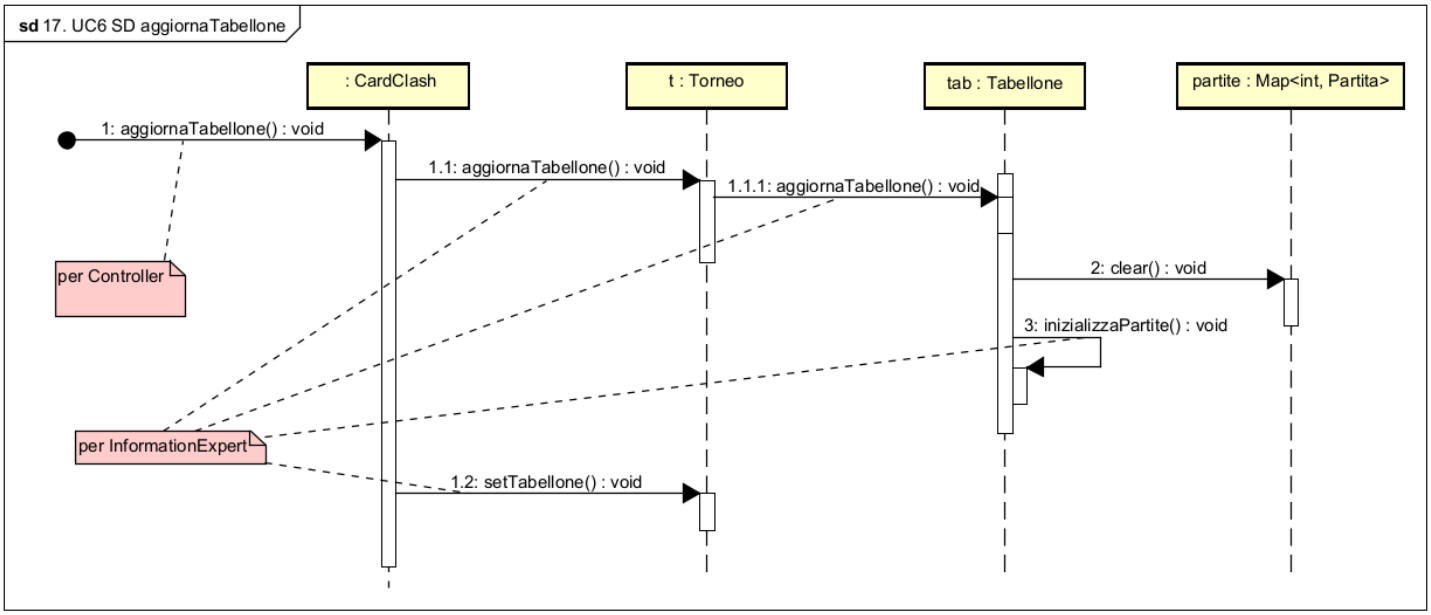
3.4 Modifica SD mostraTorneiDisponibili (UC3)



3.5 Modifica SD creaTabellone (UC5)

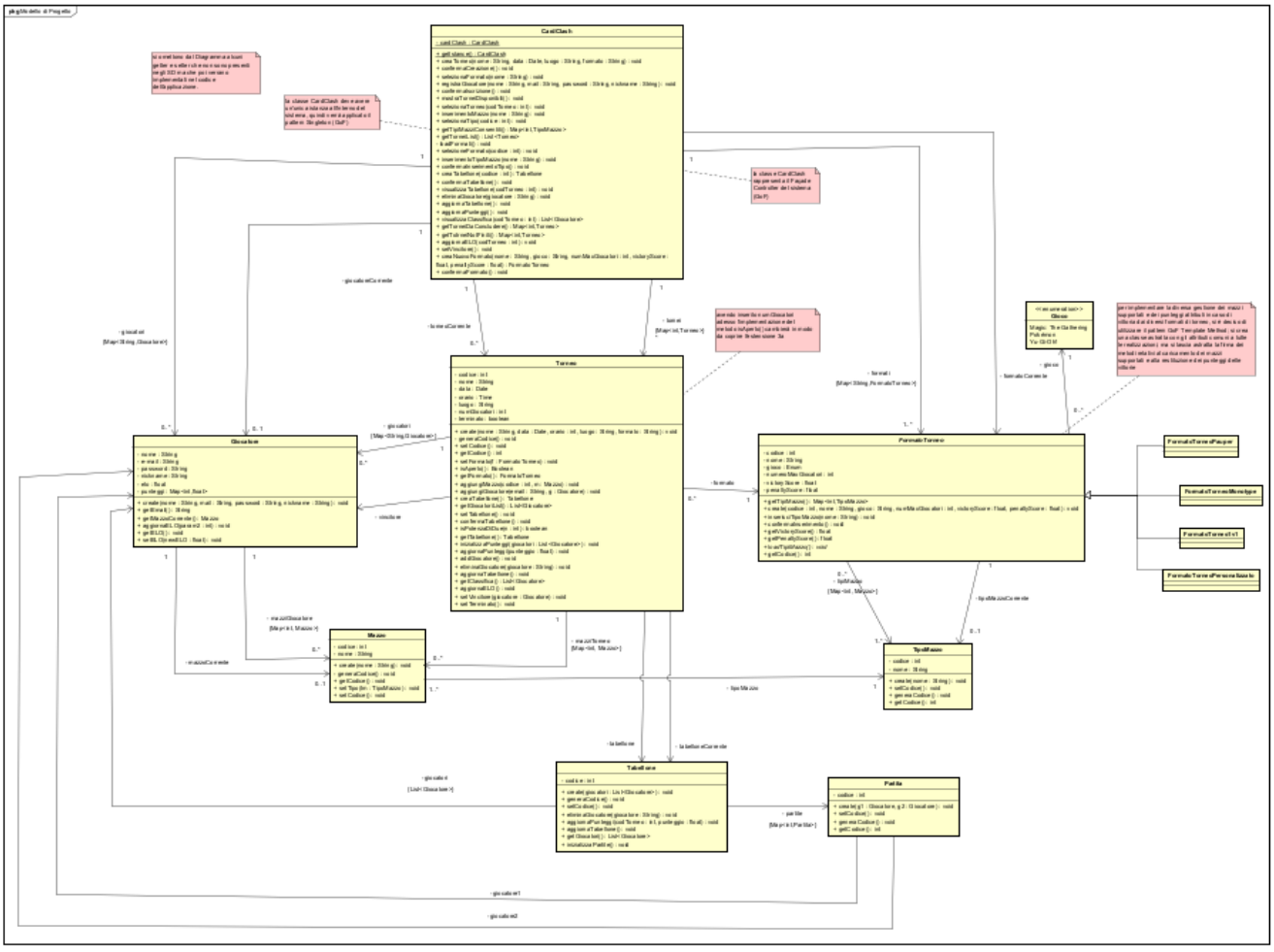


3.6 Modifica SD aggiornaTabellone (UC6)



4.0 Diagramma delle classi

Si riporta il nuovo diagramma delle classi per la corrente iterazione:



5.0 Testing

Durante la terza iterazione, alcuni metodi di testing definiti in precedenza sono stati aggiornati per adattarsi ai cambiamenti introdotti nel sistema durante questa fase. Inoltre, sono stati testati i nuovi metodi implementati per coprire i casi d'uso modellati in questa iterazione. Riportiamo di seguito i nuovi metodi di test:

CardClashTest

1. **testVisualizzaClassifica()**
   - Verifica che la classifica venga visualizzata correttamente, in base ai giocatori presenti nel torneo.
2. **testGetTorneiDaConcludere()**
   - Verifica che un torneo non concluso sia correttamente presente nella mappa dei tornei da concludere e che, un torneo che invece è stato concluso, non sia presente all'interno della mappa.
3. **testAggiornaELO()**
   - Verifica che l'elo di ogni giocatore, dato il codice di un torneo, venga correttamente aggiornato.
4. **testSetVincitore()**
   - Verifica che una volta che un torneo viene concluso, il giocatore in testa alla classifica venga correttamente settato come vincitore.
5. **testCreaNuovoFormato()**
   - Verifica che la creazione di un nuovo formato vada a buon fine, verificando inoltre che il gioco per il quale si sta creando il nuovo formato sia supportato dal sistema.
6. **testConfermaFormato()**
   - Verifica che, una volta creato un nuovo formato per un gioco supportato, la conferma inserisca correttamente il nuovo formato tra quelli supportati dal sistema.

GicatoreTest

1. **testAggiornaELO()**
   - Verifica che l'elo dei giocatori, dato il codice di un torneo, venga correttamente aggiornato.

FormatoTorneoTest

1. **testLoadTipiMazzoPersonalizzato()**
   - Verifica il corretto funzionamento della classe FormatoTorneoPersonalizzato al momento della sua creazione.

TorneoTest

1. **testIsAperto()**
   - Verifica che il torneo sia aperto non appena viene creato(con una data futura). Viene anche verificato che quando il numero dei giocatori raggiunge il numero massimo consentito il torneo non sia più aperto.
2. **testIsPotenzaDiDue()**
   - Verifica che la funzione restituisca correttamente i risultati del controllo per vari numeri forniti in ingresso.
3. **testEliminaGiocatore()**
   - Verifica che l'eliminazione di un giocatore presente in un torneo avvenga correttamente.
4. **testGetClassifica()**
   - Verifica che una classifica venga creata nell'ordine corretto dei giocatori.
5. **testAggiornaELO()**
   - Verifica il corretto aggiornamento dell'ELO dei giocatori.
6. **testConcludiTorneo()**
   - Verifica che venga correttamente settato l'attributo "terminato" e che il vincitore sia settato.