Iterazione 1

1.0 Introduzione

La prima iterazione si basa sull'analisi e progettazione dei primi tre casi d'uso, ritenuti fondamentali ai fini del funzionamento dell'applicativo.

Scopo di questa, così come delle iterazioni che verranno dopo, è quello di andare ad Implementare in maniera iterativa l'applicativo software raffinando costantemente la sua Visione, identificando e implementando la maggior parte dei requisiti richiesti e test per garantire il corretto funzionamento del codice dell'applicazione.

In particolare, in questa prima iterazione andremo ad implementare gli scenari principali di successo dei primi tre casi d'uso:

- **UC1**: Creazione Torneo
- **UC2**: Tesseramento giocatori
- **UC3**: Iscrizione Torneo

1.1 Aggiornamento dei casi d'uso

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. Se verranno apportate ulteriori modifiche anche a documenti precedenti, queste saranno tramite tramite cronologia.

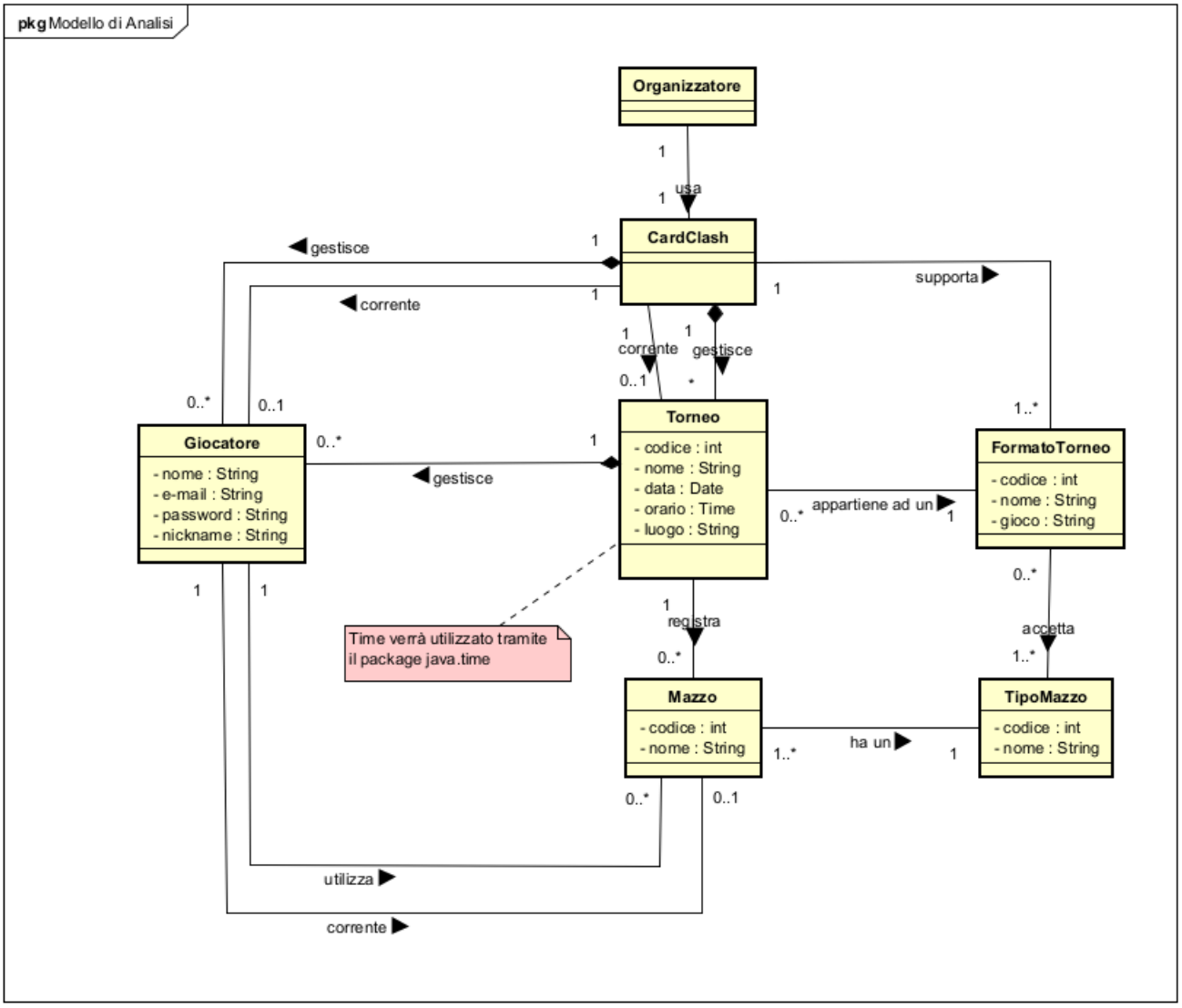
2.0 Fase di Analisi

Le classi concettuali identificate durante questa prima iterazione sono:

- **Organizzatore:** Attore primario che interagisce con il sistema.

- **CardClash:** Rappresenta il sistema CardClash.
- **Torneo:** Rappresenta un torneo, e le relative informazioni come l'id di identificazione, gestite dal sistema.
- **FormatoTorneo:** Tipologie di torneo supportate dal sistema.
- **Giocatore:** Utilizzatore del sistema CardClash, che può effettuare l'iscrizione al torneo e parteciparci.
- **Mazzo:** Rappresenta il Mazzo presentato dall'utente durante l'iscrizione ad un torneo.
- **TipoMazzo:** Rappresenta la tipologia a cui appartiene un Mazzo iscritto e supportato dal sistema.

Da cui tenendo conto di associazioni e attributi, ottenuti analizzando i primi tre casi d'uso, è stato ricavato il seguente Modello di Dominio:

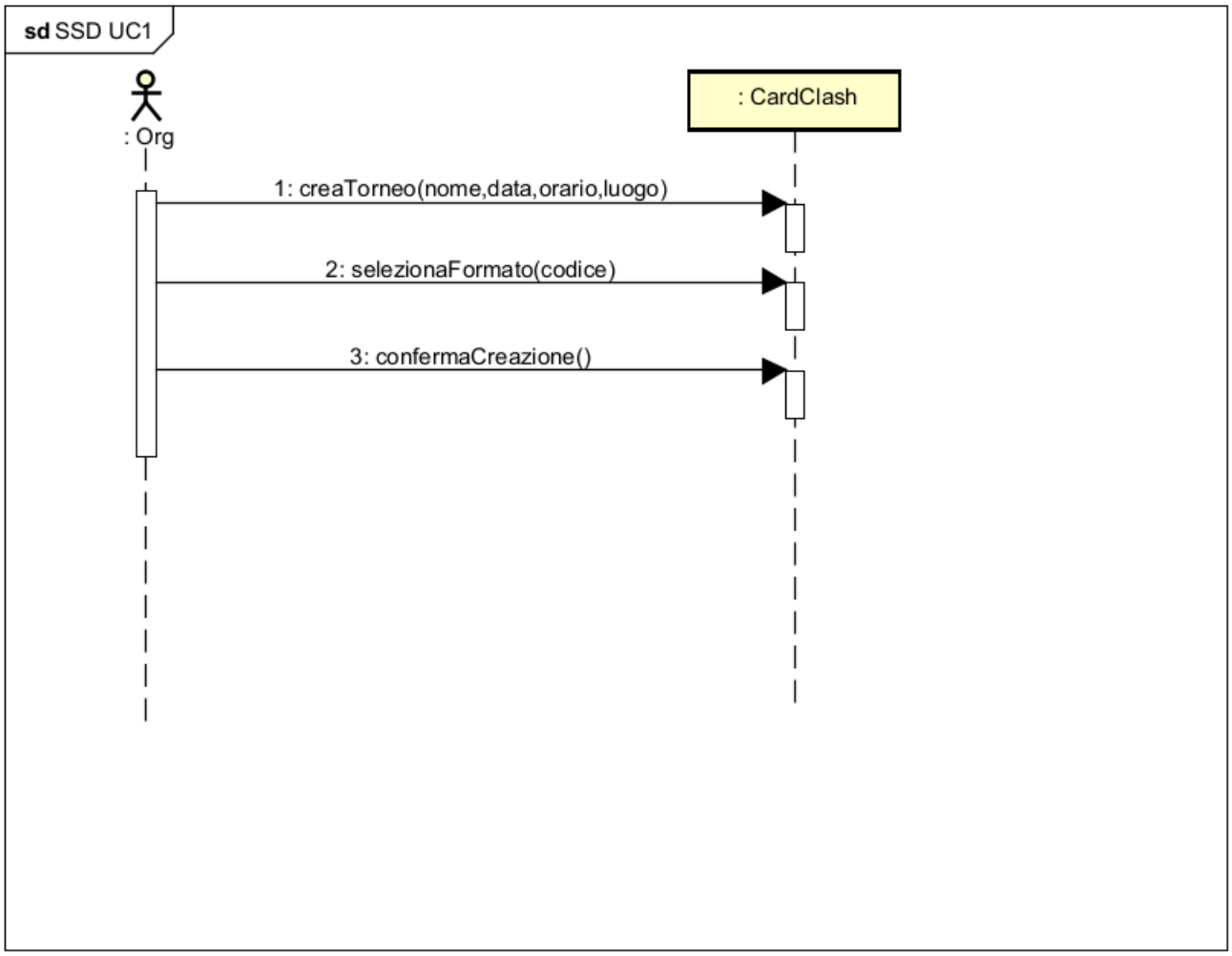


Per come è stato pensato CardClash, non è necessario avere una classe concettuale per i giochi di carte; infatti, essi sono solamente utili per distinguere i formati supportati dal sistema e distinguerli da eventuali loro simili applicati ad un altro gioco (ad esempio, un torneo solo carte comuni su Pokèmon e uno su Yu-Gi-Oh! saranno due formati di torneo diversi). Per tale ragione, abbiamo messo un codice identificativo per facilitare la ricerca del formato nel sistema.

2.1 Diagrammi di sequenza di sistema

- **UC1:** Creazione Torneo

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 UC1: Creazione Torneo

- **Operazione:** creaTorneo(nome,data,orario,luogo)
- **Riferimenti:** Caso d'uso: Creazione Torneo
- **Pre-condizione:** L'organizzatore è autenticato e ha accesso ai permessi di creazione.
- **Post-condizione:**
  - È stata creata un'istanza t di Torneo
  - Gli attributi di t vengono inizializzati
  - t diventa corrente per CardClash

Contratto2 UC1: Creazione Torneo

- **Operazione:** selezionaFormato(codice)
- **Riferimenti:** Caso d'uso: Creazione Torneo
- **Pre-condizione:**
  - È stata già creata una nuova istanza t di torneo
  - Gli attributi di t sono stati inizializzati
  - t è corrente per CardClash
- **Post-condizione:**
  - Viene inserito il formato f per il torneo t corrente

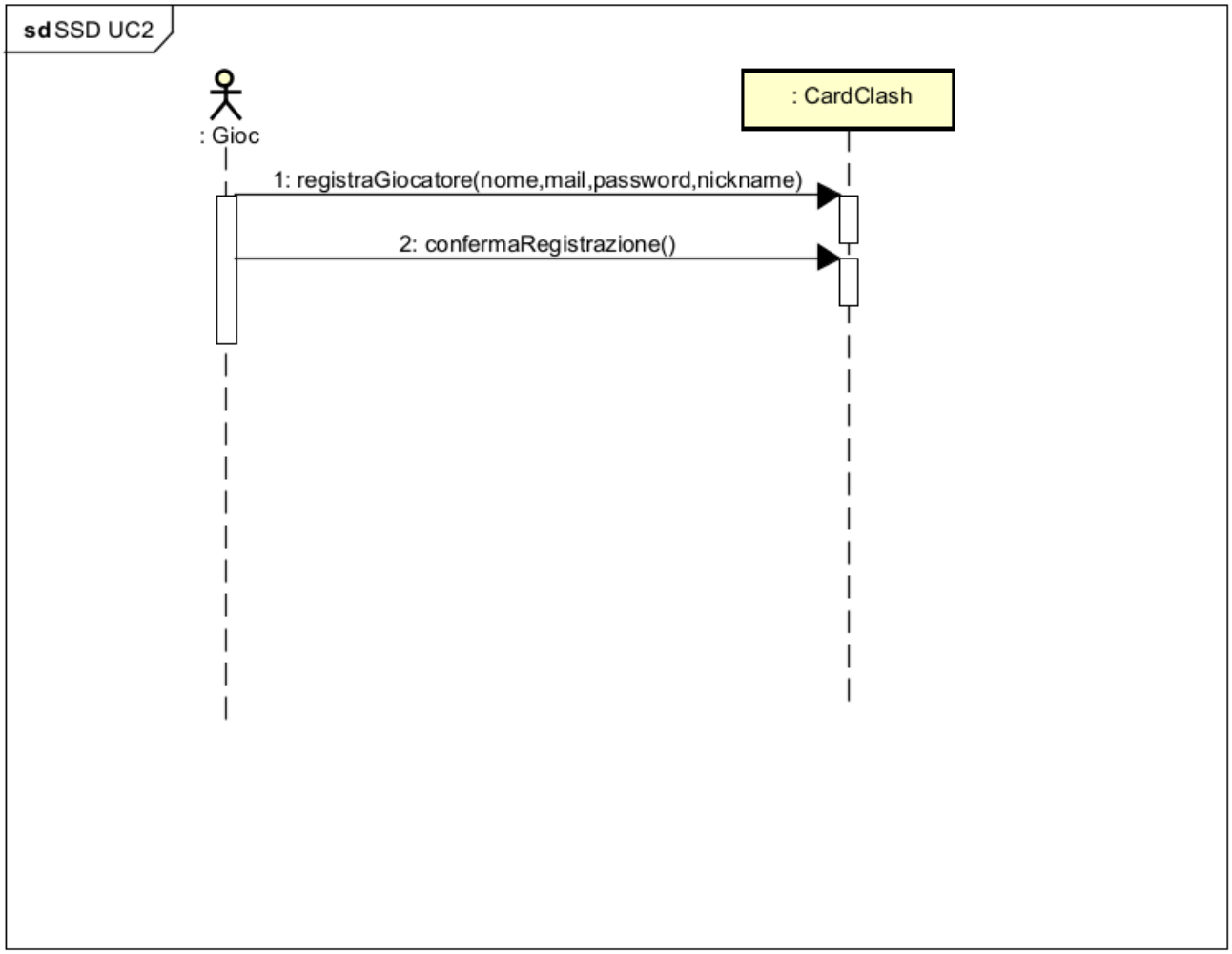
Contratto3 UC1: Creazione Torneo

- **Operazione:** confermaCreazione()
- **Riferimenti:** Caso d'uso: Creazione Torneo
- **Pre-condizione:** Gli attributi di t sono stati inizializzati correttamente e gli è stato associato il formato f di torneo selezionato
- **Post-condizione:**
  - Viene associato a t un id univoco
  - Si salva l'istanza t di Torneo, e la si associa a CardClash

2.2 Diagrammi di sequenza di sistema

- **UC2:** Tesseramento Giocatori

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 UC2: Tesseramento Giocatori

- **Operazione:** registraGiocatore(nome,mail,password,nickname)
- **Riferimenti:** Caso d'uso: Tesseramento Giocatori
- **Pre-condizione:**
  - Il giocatore non deve essere già registrato
- **Post-condizione:**
  - È stata creata un'istanza g di giocatore
  - Sono stati inizializzati gli attributi di g
  - g diventa corrente per CardClash

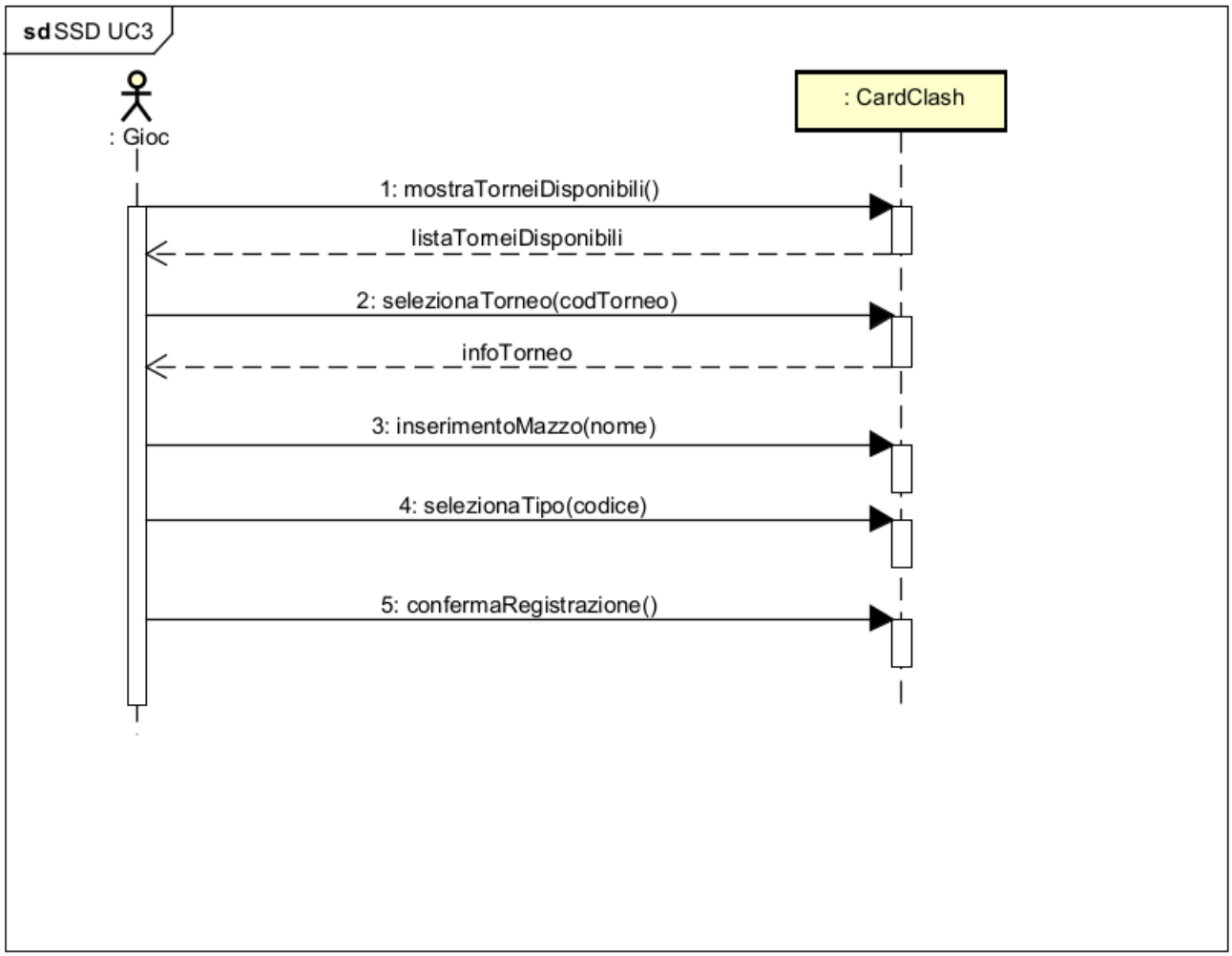
Contratto2 UC2: Tesseramento Giocatori

- **Operazione:** confermaRegistrazione()
- **Riferimenti:** Caso d'uso: Tesseramento Giocatori
- **Pre-condizione:**
  - È stata creata un'istanza g di giocatore
  - g è corrente
- **Post-condizione:**
  - Si salva l'istanza g di giocatore, e la si associa a CardClash

2.3 Diagramma di sequenza di sistema

- **UC3:** Iscrizione 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

Contratto1 UC3: Iscrizione Torneo

- **Operazione:** mostraTornei()
- **Riferimenti:** Caso d'uso: Iscrizione Torneo
- **Pre-condizione:**
  - Sono presenti istanze di torneo
- **Post-condizione:**
  - Viene recuperata la lista dei tornei disponibili a cui è possibile iscriversi

Contratto2 UC3: Iscrizione Torneo

- **Operazione:** selezionaTorneo(codTorneo)
- **Riferimenti:** Caso d'uso: Iscrizione Torneo
- **Pre-condizione:**
  - È in corso un'iscrizione ad un torneo
  - Il giocatore deve essere registrato e autenticato all'interno del sistema

  Post-condizione:
  - Vengono restituite le informazioni del torneo t selezionato

Contratto3 UC3: Iscrizione Torneo

- **Operazione:** inserimentoMazzo(nome)
- **Riferimenti:** Caso d'uso: Iscrizione Torneo
- **Pre-condizione:**
  - Il torneo t selezionato deve essere attivo e aperto ad iscrizioni
- **Post-condizione:**
  - Viene creata un'istanza m di mazzo
  - Gli attributi di m vengono inizializzati
  - m diventa corrente per Giocatore

Contratto4 UC3: Iscrizione Torneo

- **Operazione:** selezionaTipo(codice)
- **Riferimenti:** Caso d'uso: Iscrizione Torneo
- **Pre-condizione:**
  - È stata inizializzata l'istanza m di mazzo e i suoi attributi sono stati settati correttamente
  - Sono state recuperate le tipologie di mazzo consetite per il formato del torneo t selezionato, tramite le quali l'utente potrà associarle ad m
- **Post-condizione:**

- Viene recuperata la tipologia tm di mazzo inserita
- tm viene associata al mazzo m

Contratto5 UC3: Iscrizione Torneo

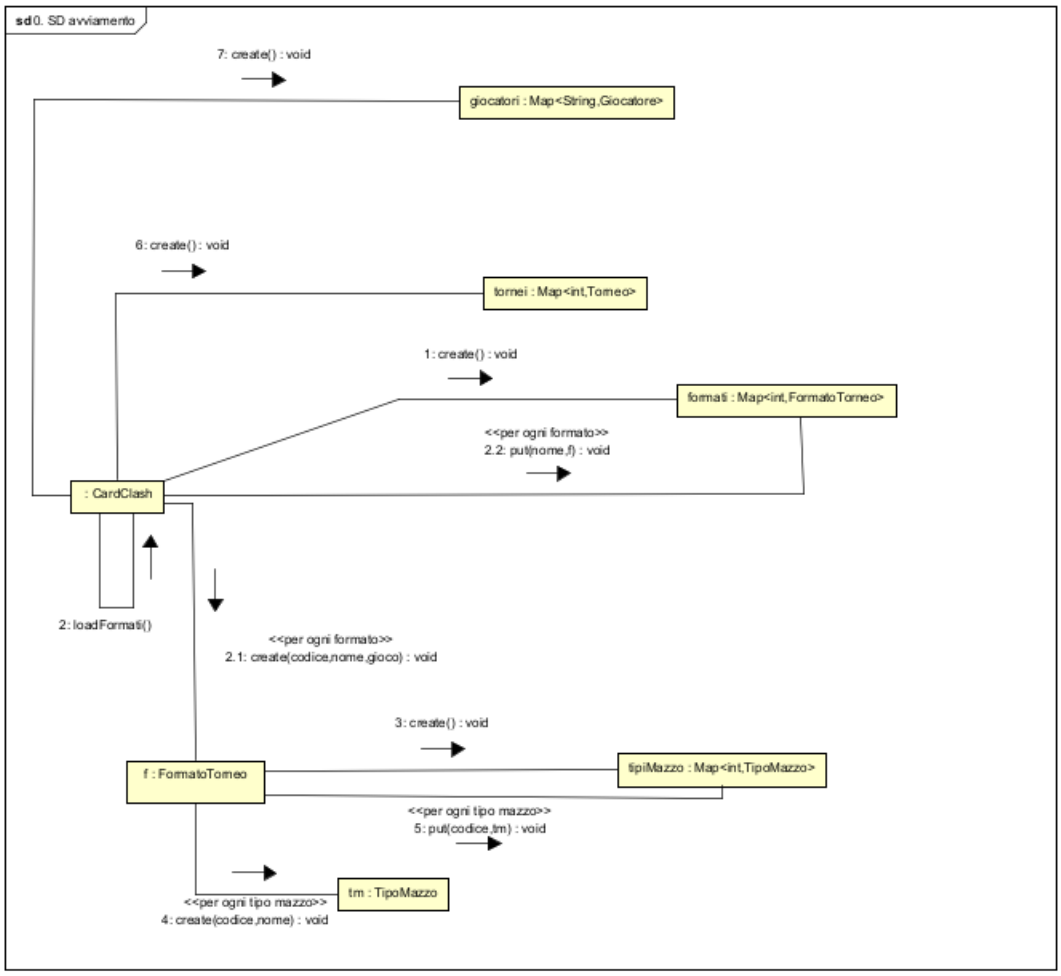
- **Operazione:** confermaRegistrazione()
- **Riferimenti:** Caso d'uso: Iscrizione Torneo
- **Pre-condizione:**
  - Gli attributi di m sono stati inizializzati
  - m è corrente per giocatore
- **Post-condizione:**
  - Si salva l'istanza di m e la si associa al giocatore
  - m viene aggiunto alla lista dei mazzi registrati per il torneo t
  - Il giocatore g viene aggiunto alla lista dei partecipanti del torneo t

3.0 Fase di Progettazione

Gli elaborati principali della fase di progettazione sono i diagrammi di interazione, articolati in diagrammi di sequenza e diagrammi di interazione, con lo scopo di descrivere il comportamento del sistema da un punto di vista dinamico durante i casi d'uso presi in considerazione per questa prima iterazione. Insieme a questi, il diagramma delle classi rappresenta il sistema da un punto di vista statico. Di seguito vengono riportati:

3.1 Diagramma di comunicazione (caso d'uso di avviamento)

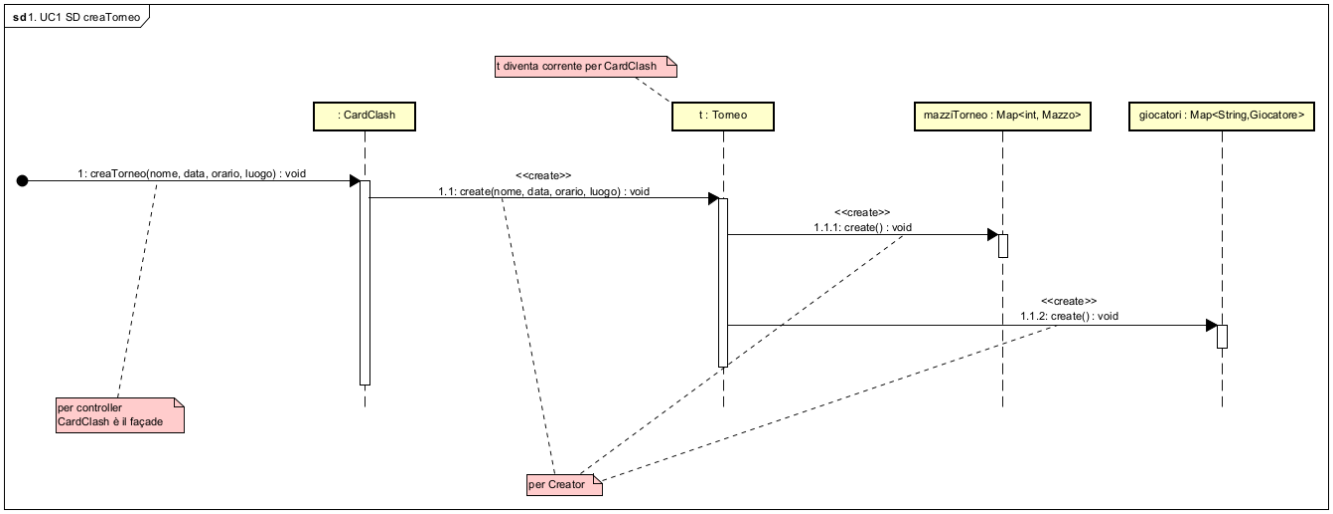
Per funzionare correttamente, il sistema dovrà seguire dei passaggi iniziali di caricamento dati e configurazione relativamente ai formati di torneo supportati da CardClash.



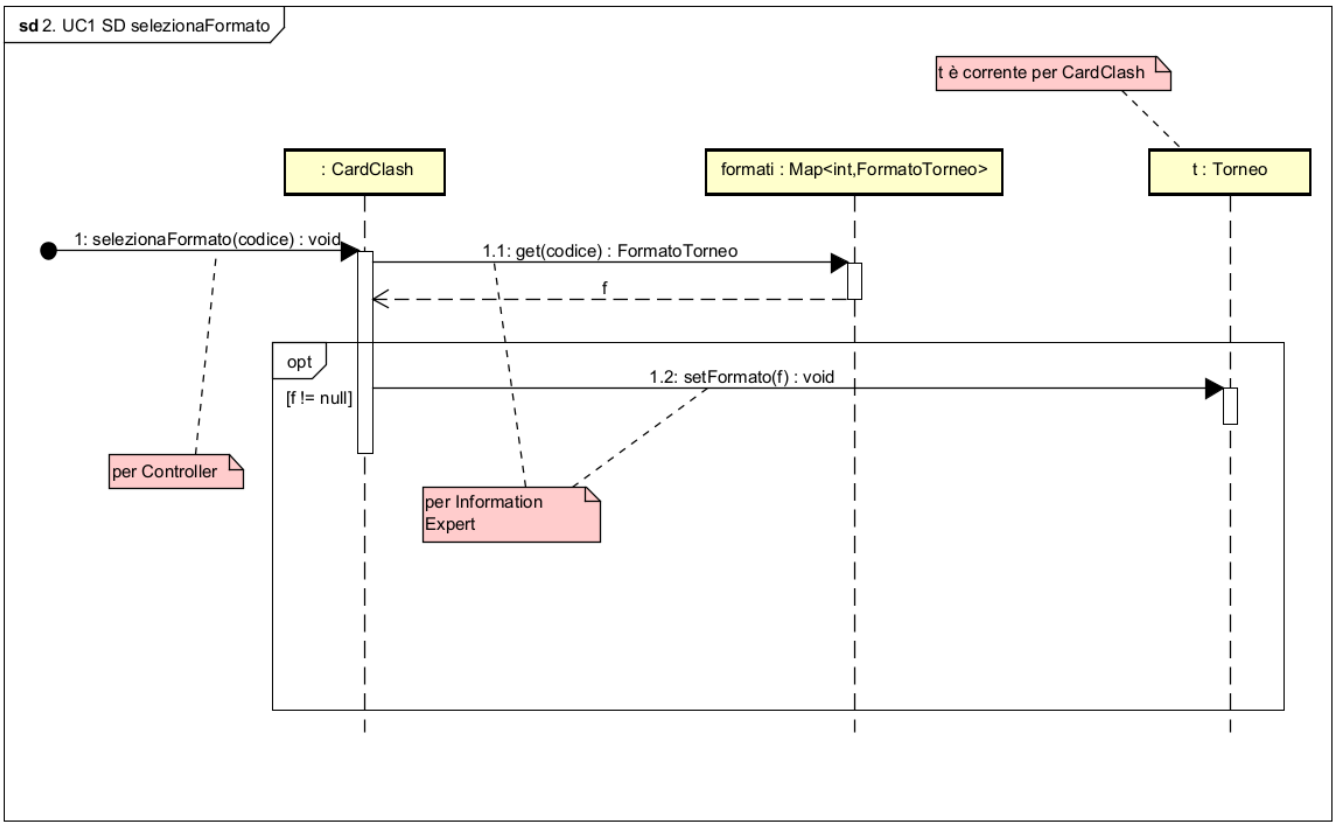
3.2 Diagrammi di sequenza UC1

CardClash è stato progettato secondo il pattern GoF Façade Controller per lavorare come interfaccia del sistema, sia quando viene utilizzato dall'Organizzatore che dai giocatori. Sarà dunque suo il compito di creare (secondo il pattern Creator) le istanze dei tornei.

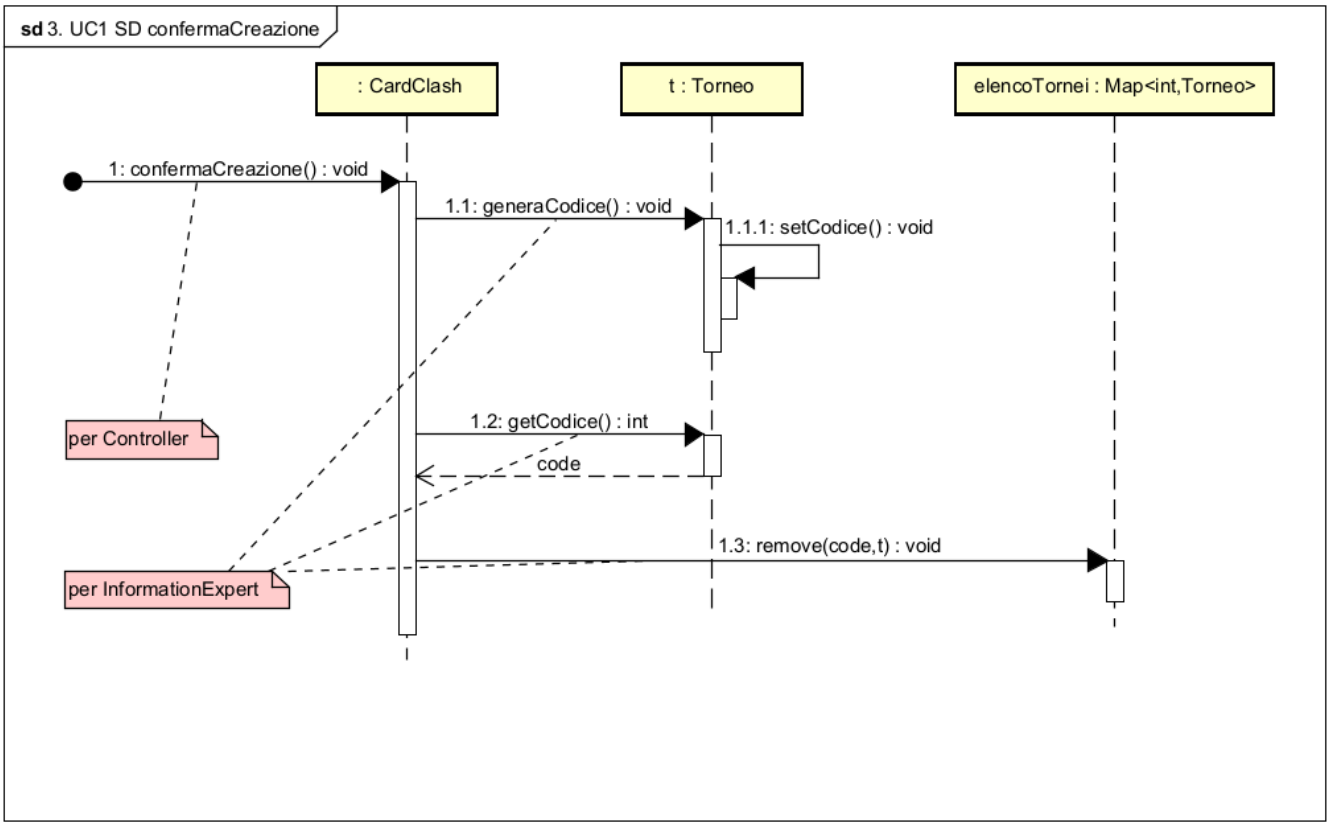
creaTorneo



selezionaFormato



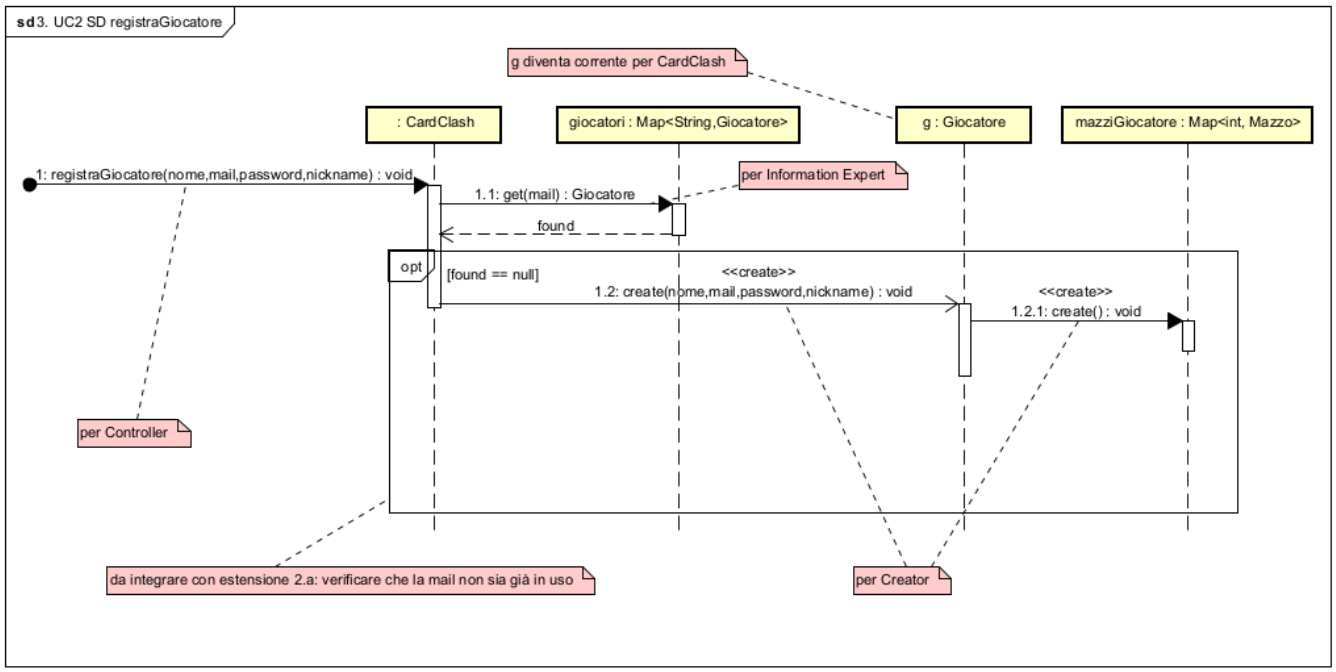
confermaCreazione



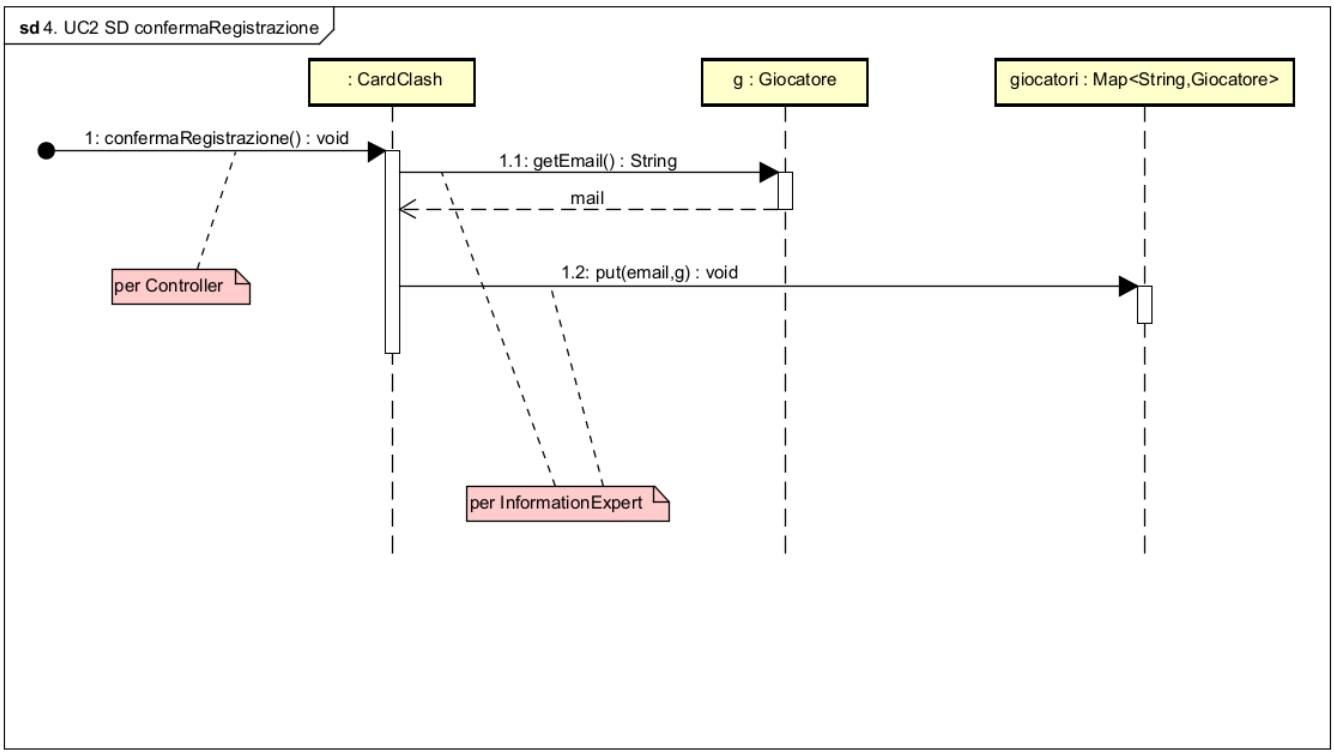
3.2 Diagrammi di sequenza UC2

CardClash tiene traccia di tutti i giocatori iscritti all'applicazione, dunque ha la responsabilità di creare nuovi oggetti di tipo Giocatore (per il pattern Creator) e di aggiungerli alla mappa dei Giocatori, che utilizza per tenerne traccia.

registraGiocatore



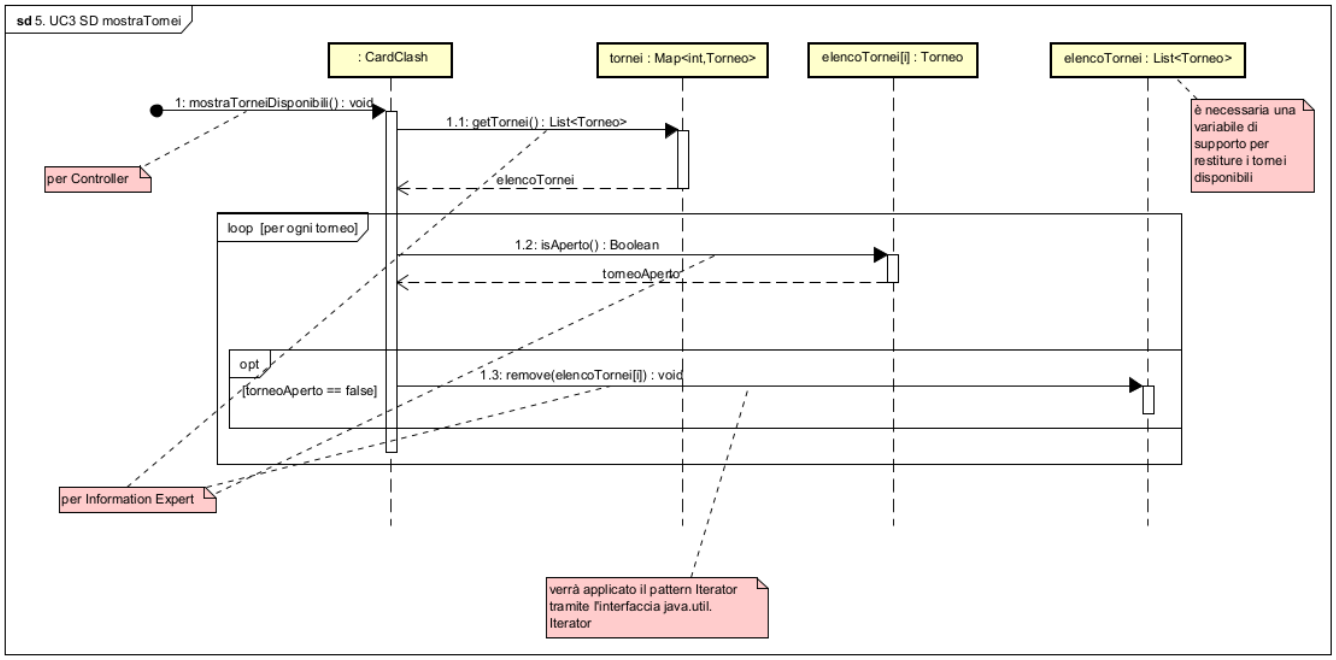
confermaRegistrazione



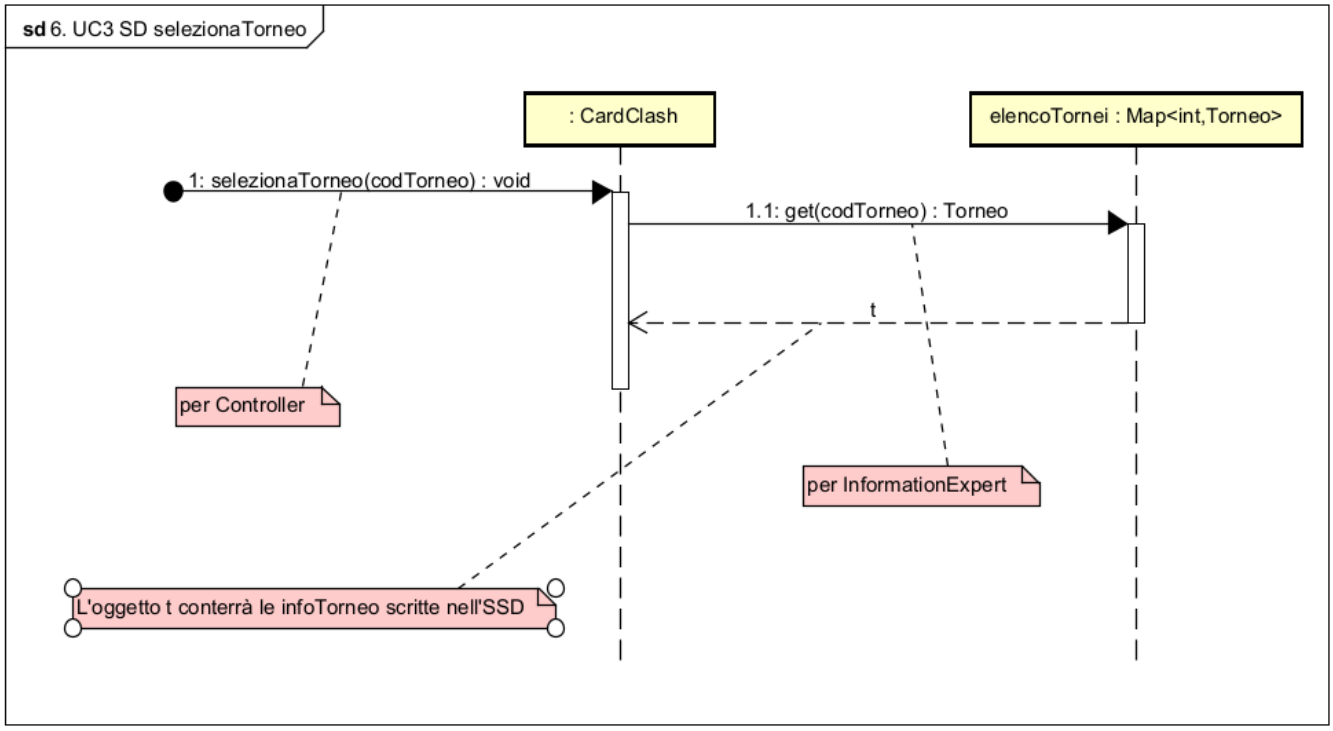
3.3 Diagrammi di sequenza UC3

Questo caso d'uso implementa la possibilità di visualizzare i tornei disponibili, registrare un mazzo con la tipologia a cui appartiene ed iscriversi al torneo.

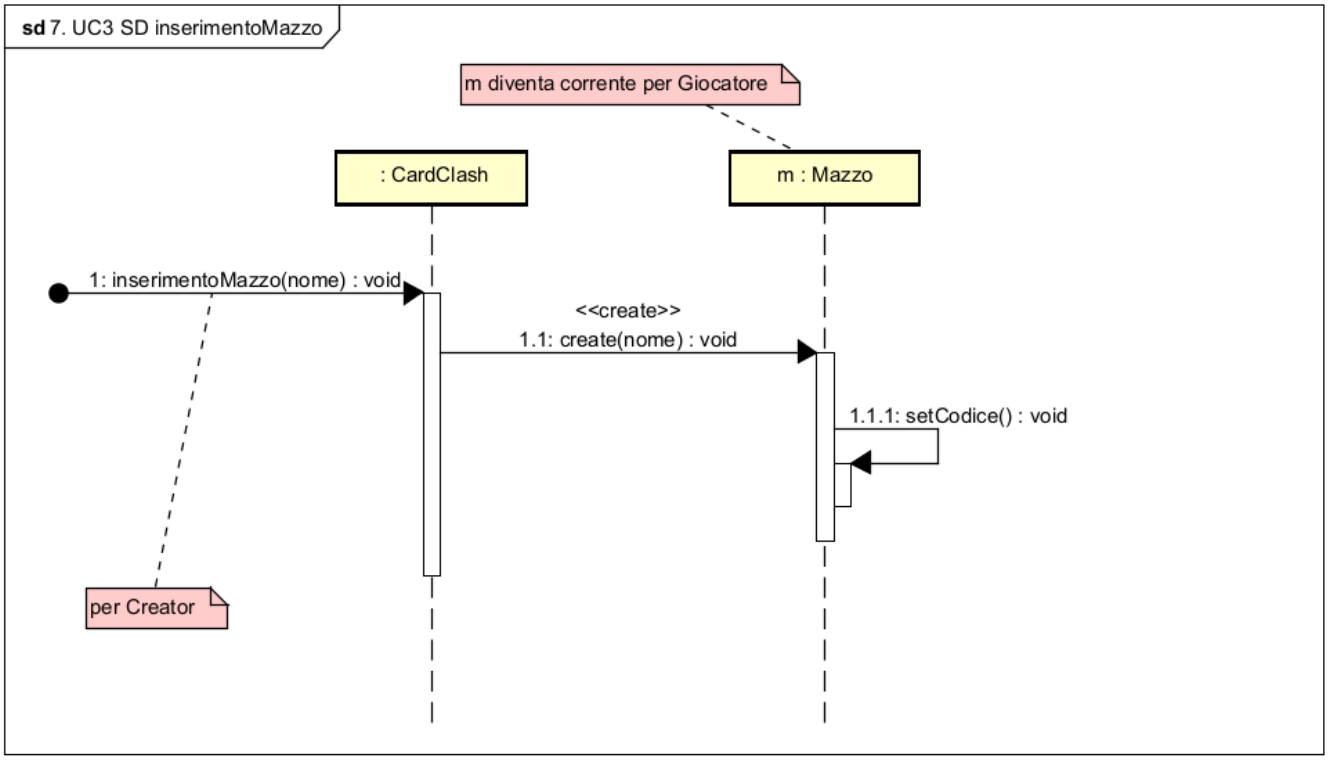
mostraTornei



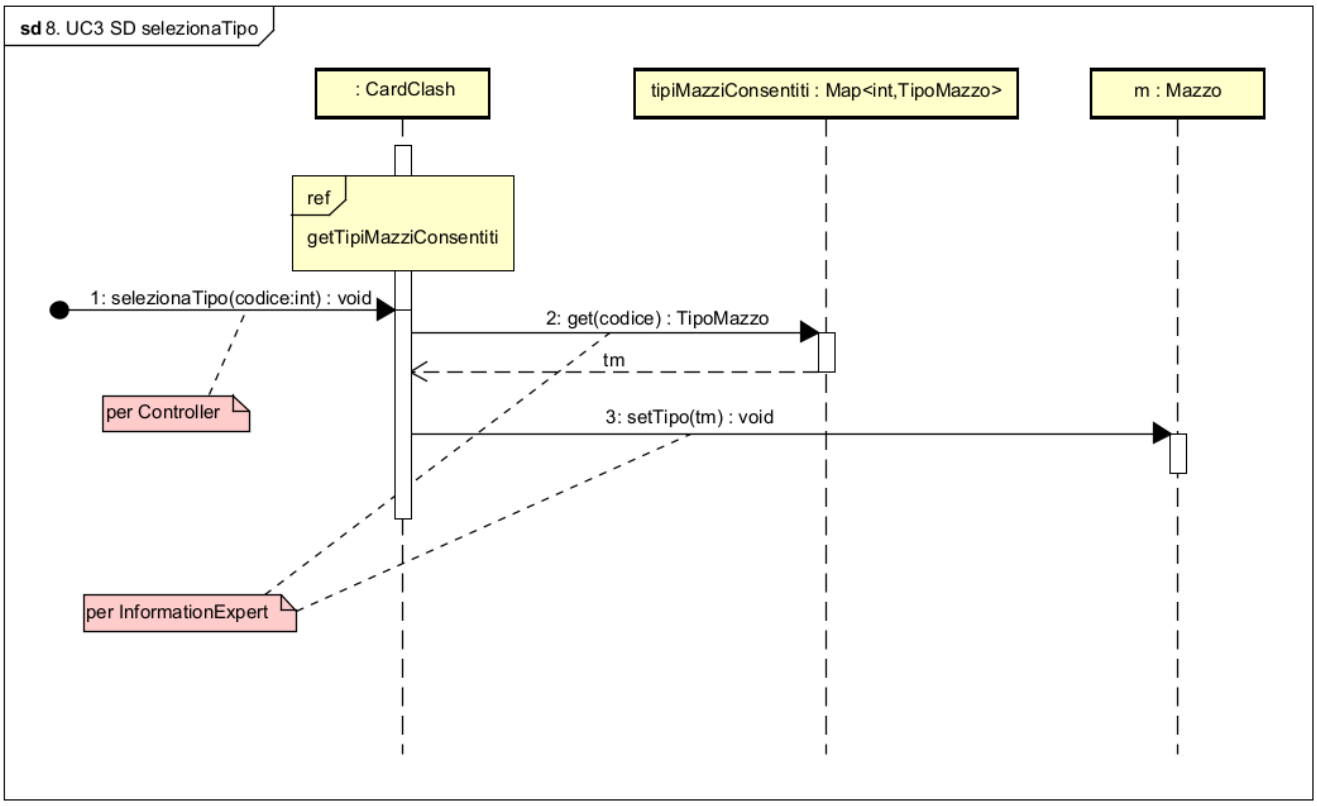
selezionaTorneo



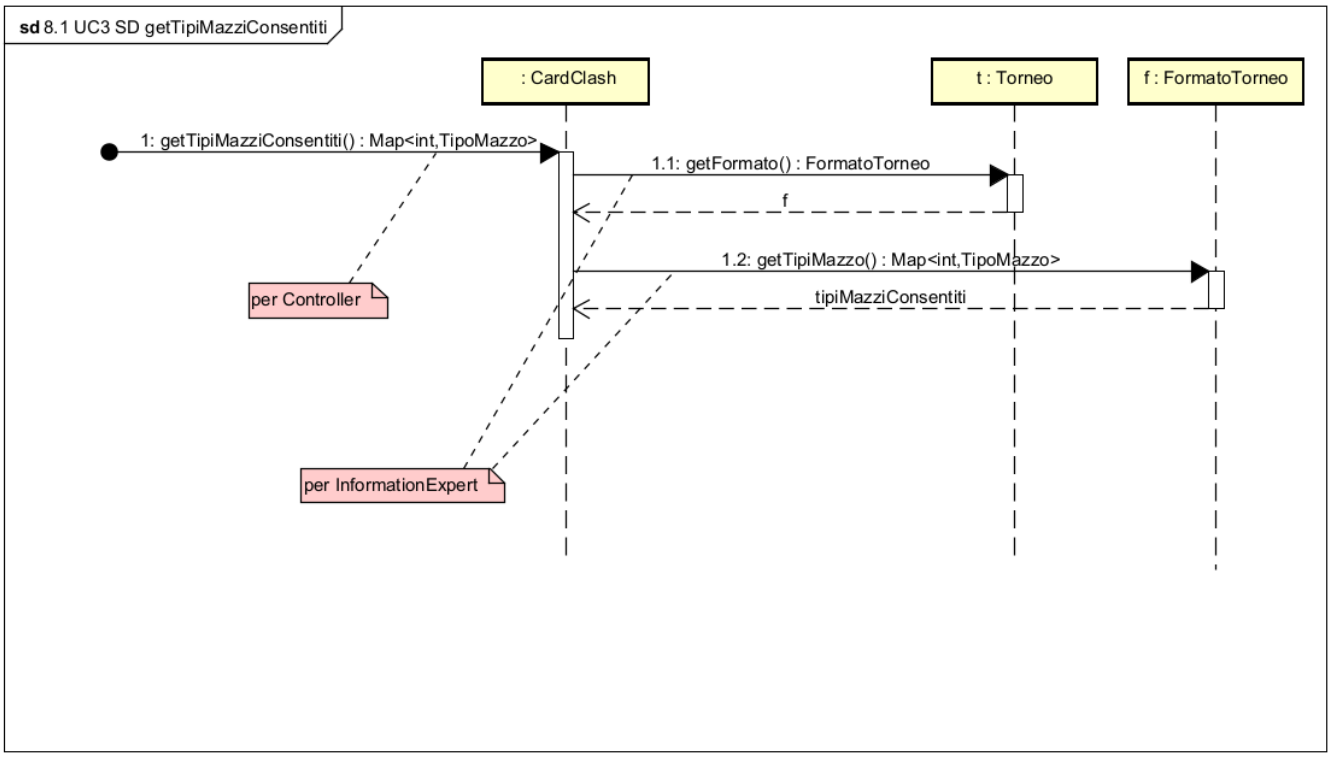
inserimentoMazzo



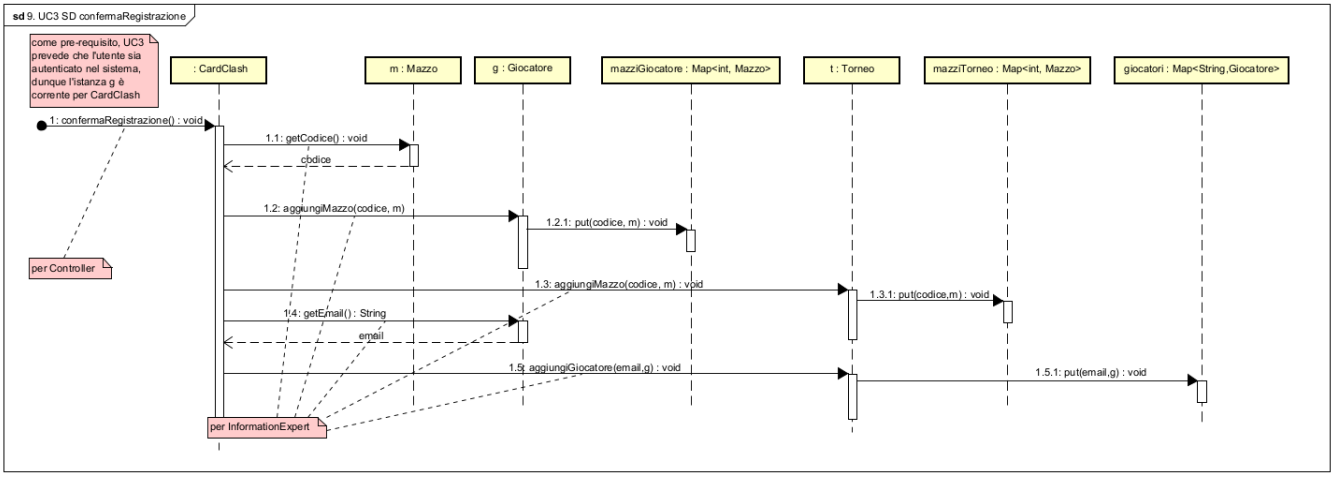
selezionaTipo



getTipiMazzoConsentiti

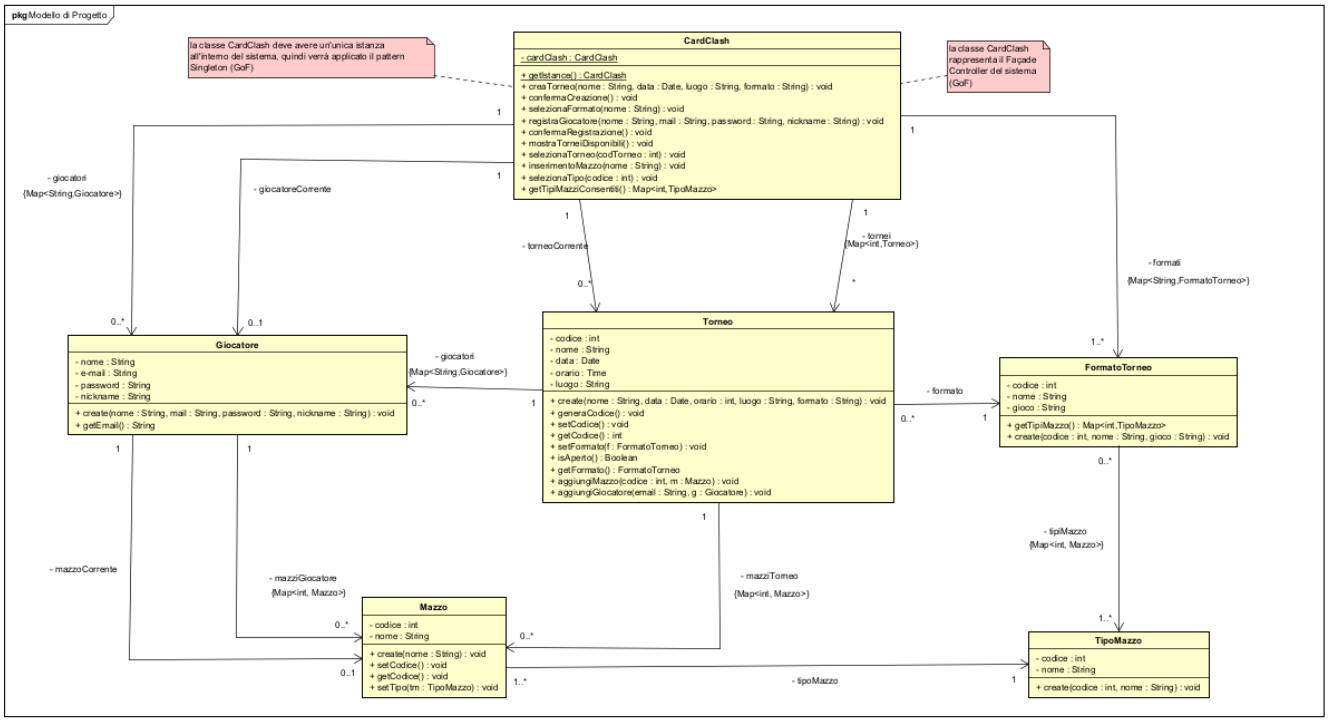


confermaRegistrazione



3.4 Diagramma delle classi

Per la classe CardClash si utilizzerà il pattern GoF Singleton e, come detto in precedenza, servirà da Façade Controller per il sistema.


pkg Modello di Progetto
la classe CardClash deve avere un'unica istanza all'interno del sistema, quindi verrà applicato il pattern Singleton (GoF)
la classe CardClash rappresenta il Façade Controller del sistema (GoF)
CardClash
- cardClash : CardClash
+ getIstance() : CardClash
+ creaTorneo(nome : String, data : Date, luogo : String, formato : String) : void
+ confermaCreazione() : void
+ selezionaFormato(nome : String) : void
+ registraGiocatore(nome : String, mail : String, password : String, nickname : String) : void
+ confermaRegistrazione() : void
+ mostraTorneiDisponibili() : void
+ selezionaTorneo(codTorneo : int) : void
+ inserimentoMazzo(nome : String) : void
+ selezionaTipo(codice : int) : void
+ getTipiMazziConsentiti() : Map<int,TipoMazzo>
- giocatori
{Map<String,Giocatore>}
- giocatoreCorrente
- torneoCorrente
- tornei
{Map<int,Torneo>}
- formati
{Map<String,FormatoTorneo>}
Giocatore
- nome : String
- e-mail : String
- password : String
- nickname : String
+ create(nome : String, mail : String, password : String, nickname : String) : void
+ getEmail() : String
- giocatori
{Map<String,Giocatore>}
Torneo
- codice : int
- nome : String
- data : Date
- orario : Time
- luogo : String
+ create(nome : String, data : Date, orario : int, luogo : String, formato : String) : void
+ generaCodice() : void
+ setCodice() : void
+ getCodice() : int
+ setFormato(f : FormatoTorneo) : void
+ isAperto() : Boolean
+ getFormato() : FormatoTorneo
+ aggiungiMazzo(codice : int, m : Mazzo) : void
+ aggiungiGiocatore(email : String, g : Giocatore) : void
- formato
FormatoTorneo
- codice : int
- nome : String
- gioco : String
+ getTipiMazzo() : Map<int,TipoMazzo>
+ create(codice : int, nome : String, gioco : String) : void
- mazzoCorrente
- mazziGiocatore
{Map<int, Mazzo>}
- mazziTorneo
{Map<int, Mazzo>}
- tipiMazzo
{Map<int, Mazzo>}
Mazzo
- codice : int
- nome : String
+ create(nome : String) : void
+ setCodice() : void
+ getCodice() : void
+ setTipo(tm : TipoMazzo) : void
- tipoMazzo
TipoMazzo
- codice : int
- nome : String
+ create(codice : int, nome : String) : void