2.0 ITERAZIONE 1

2.1 Introduzione

In questa iterazione approfondiremo i casi d' uso già introdotti, individuando le classi concettuali, gli attributi e le loro relazioni per la costruzione del modello di dominio. Verranno inoltre inseriti il diagramma di sequenza (SSD) e i contratti (CO) delle operazioni del caso d'uso (UC) di riferimento. In seguito verrà effettuata la progettazione orientata agli oggetti, descritta dai diagrammi di interazione e dei diagrammi delle classi (DCD) analizzati nel domino delle classi.

2.2 UC1: Modello di dominio

In questa iterazione sono state identificate le seguenti classi concettuali:

Giocatore: rappresenta l'attore primario, colui che interagisce con il sistema;

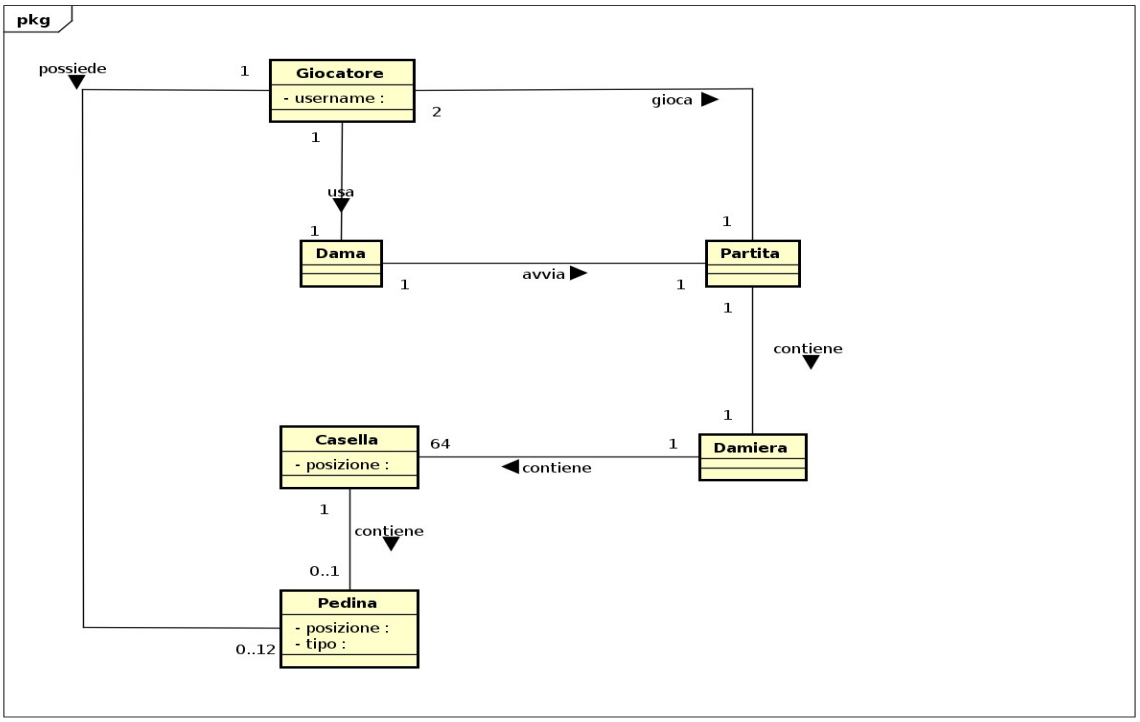
Dama: rappresenta il sistema Dama;

Partita: contiene tutte le componenti dello scenario di esecuzione della partita;

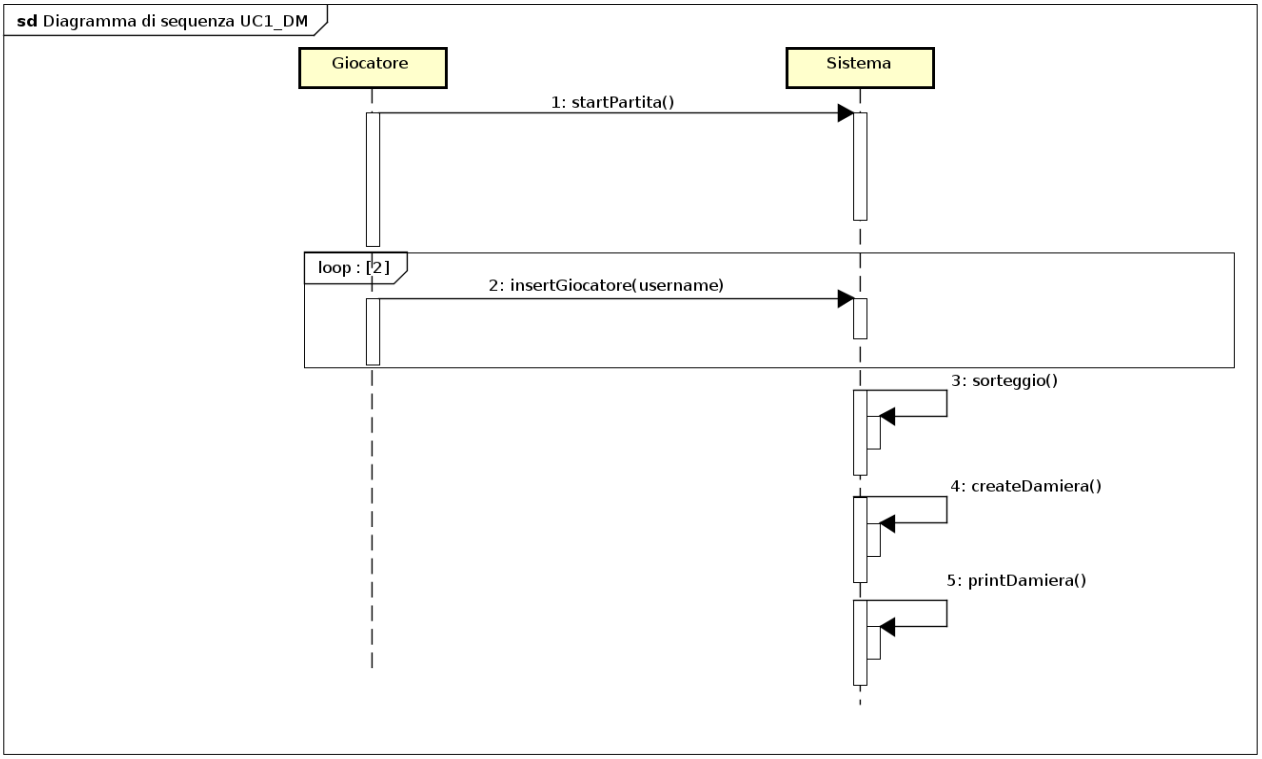
Pedina: rappresenta il pezzo su cui verrà eseguita la mossa. Contiene informazione relative alla posizione e al tipo di pedina;

Casella: componente della damiera. Può contenere una pedina. Contiene informazioni relative alla posizione;

Damiera: contiene più caselle;



2.2.1 Diagramma di Sequenza SSD



2.2.2 Contratti delle operazioni

Tramite i contratti verranno descritte le operazioni individuate grazie al diagramma di sequenza.

Contratto CO1: startPartita
Riferimenti: UC1
Operazione: startPartita()
Precondizioni: Un giocatore digita il comando per avviare una nuova partita
PostCondizioni: -Viene creata l'istanza Partita p
-Vengono create le istanze Giocatore g1 e Giocatore g2
-g1.username , g2.username vengono inizializzate
-Vengono associati g1 e g2 alla Partita tramite l'associazione gioca

Contratto CO2: insertGiocatore
Riferimenti: UC1
Operazione: insertGiocatore(username : String)
Precondizioni: -La partita è avviata
-I giocatori inseriscono gli username
PostCondizioni: -Viene creata l'istanza Giocatore g
-L'attributo g.username viene inizializzato
-L'attributo g.pedine viene inizializzato
-L'attributo g.countPedine viene inizializzato

Contratto CO3: sorteggio
Riferimenti: UC1
Operazione: sorteggio()
PostCondizioni: -L’attributo num di ogni istanza di giocatori[i] viene settato
-L’attributo colore di ogni istanza di giocatori[i] viene settato
-Viene settato l’attributo turno di ogni istanza p.giocatori[i] in base al colore delle pedine assegnate

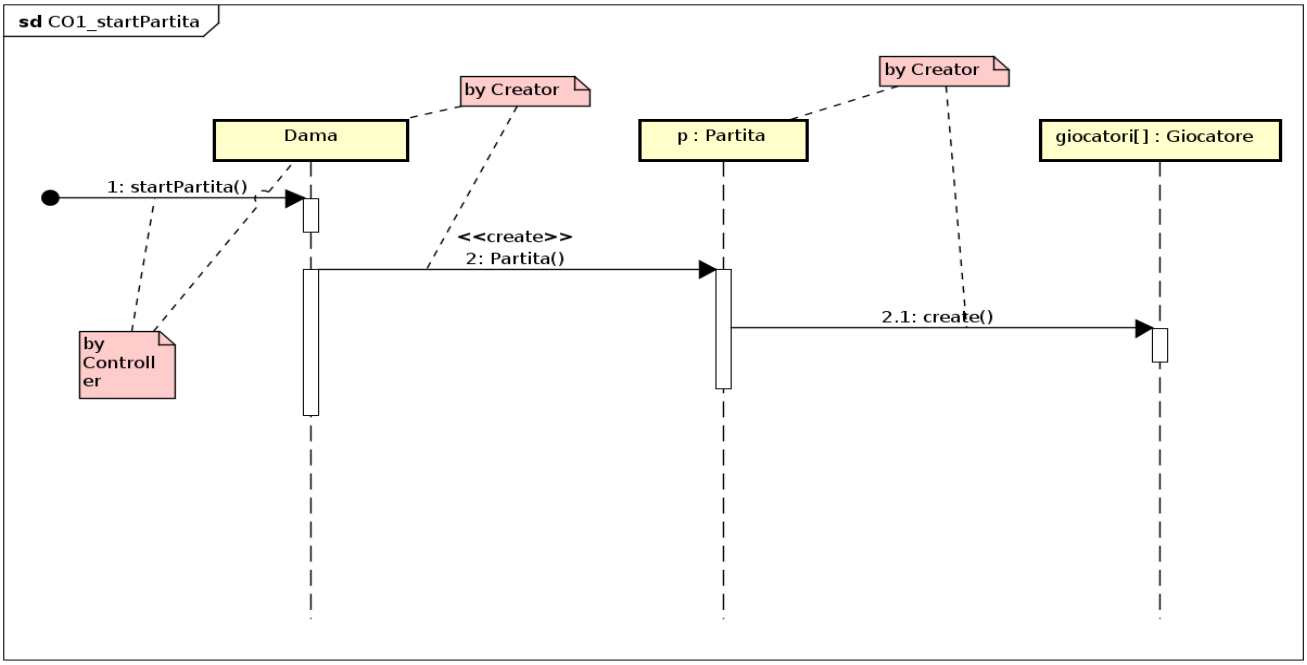
Contratto CO4: createDamiera
Riferimenti: UC1
Operazione: createDamiera()
PostCondizioni: -É stata creata l’istanza Damiera d
-É stata creata l’istanza Casella cpn, cpb, cv
-Gli attributi *simbolo, riga e colonna* di cpn, cpb e cv vengono inizializzati

Contratto CO5: printDamiera
Riferimenti: UC1
Operazione: printDamiera()
PostCondizioni: -La damiera è visibile nello scenario della partita
-Le pedine sono visibili sulla Damiera

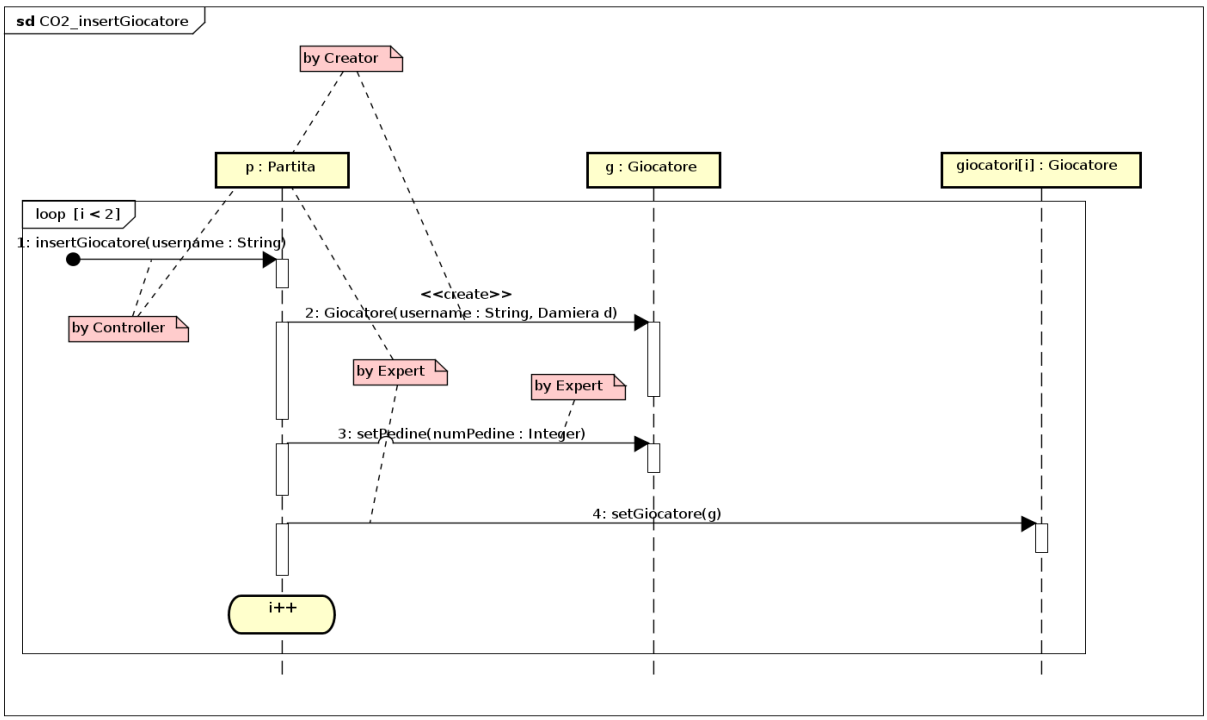
2.3 UC1: Progettazione orientata agli oggetti

2.3.1 Diagrammi di Interazione:

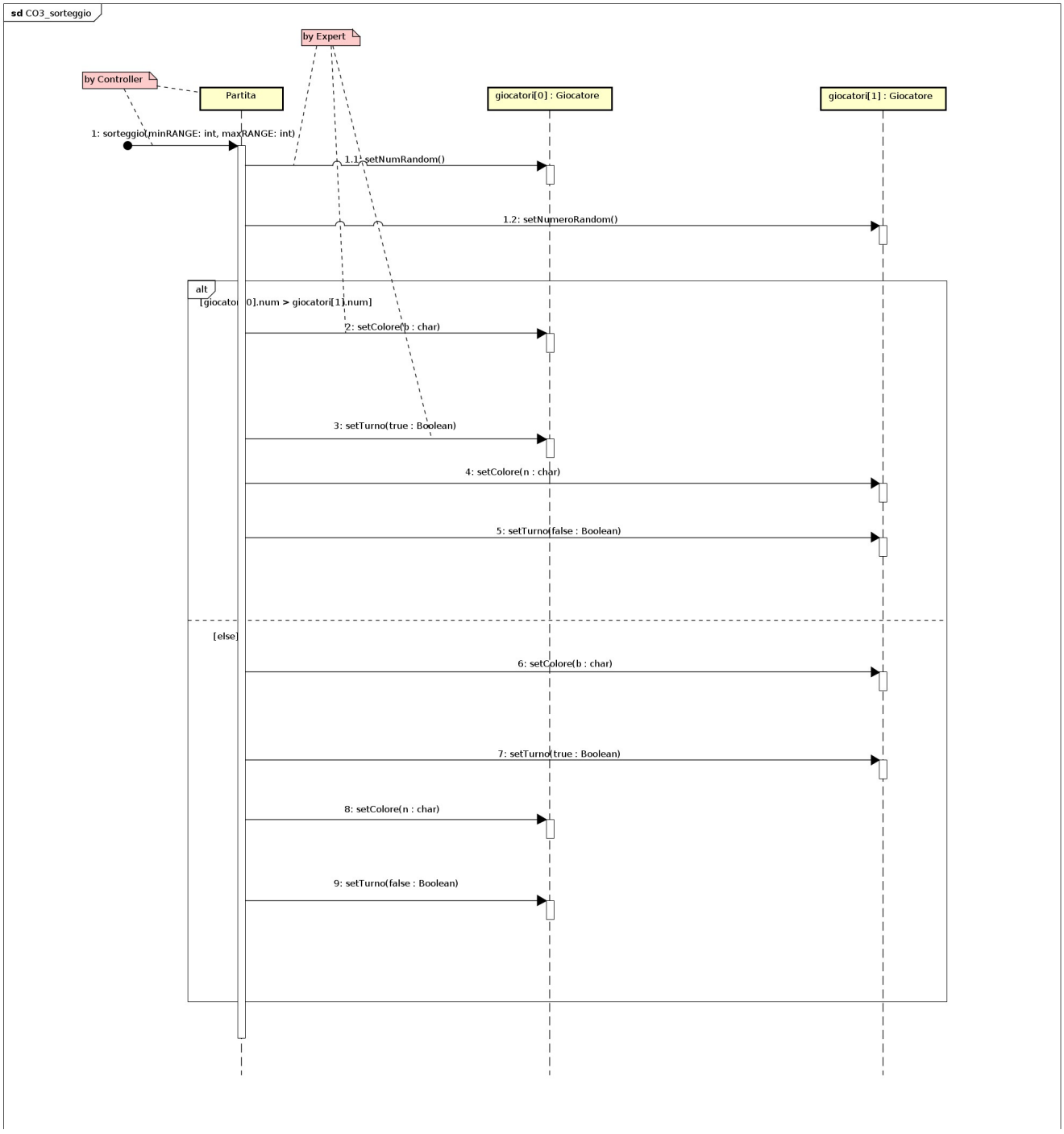
- **startPartita**



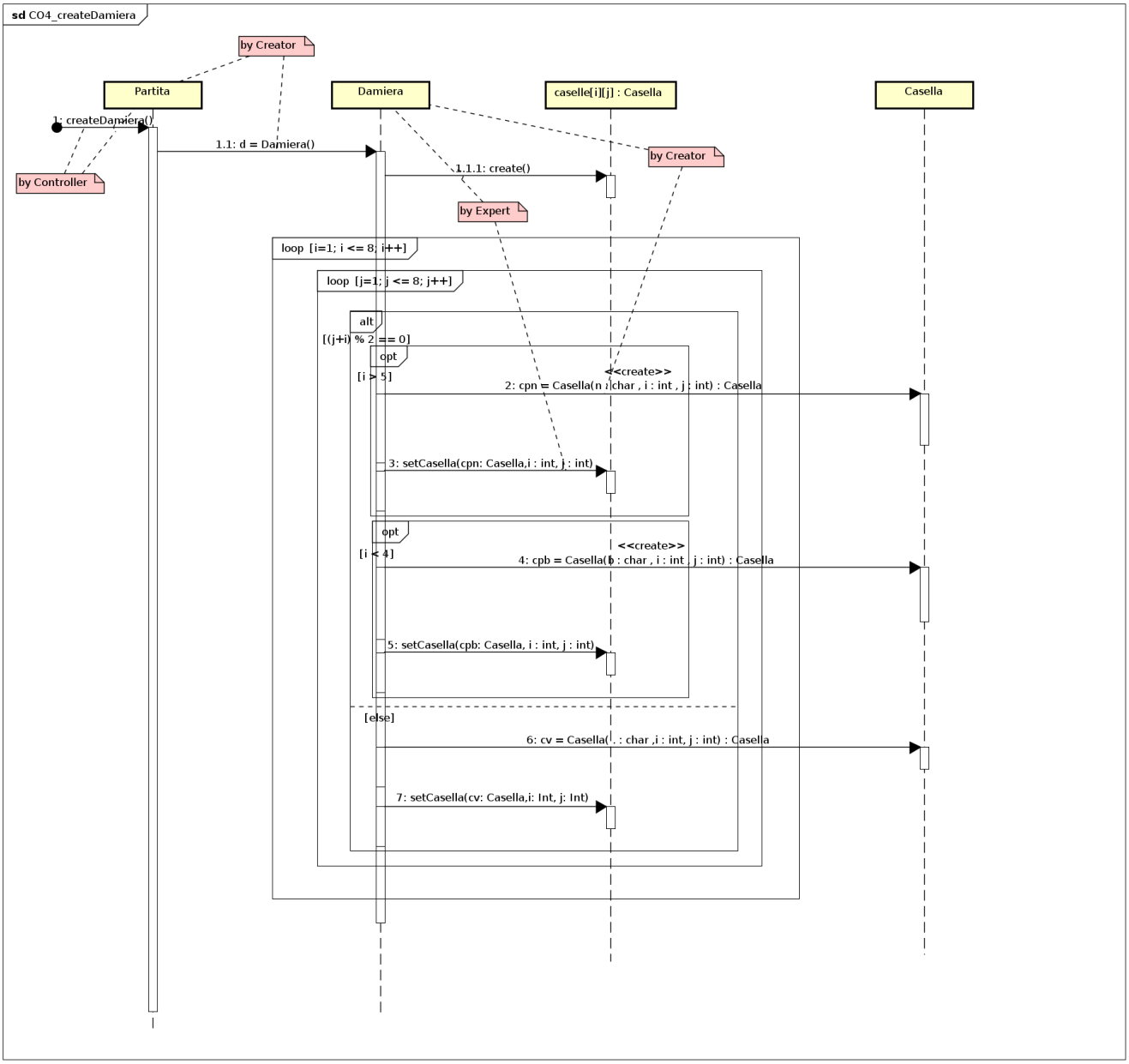
- **InsertGiocatore**



- **sorteggio**


sd CO3_sorteggio
by Expert
by Controller
Partita
giocatori[0] : Giocatore
giocatori[1] : Giocatore
1: sorteggio(minRANGE: int, maxRANGE: int)
1.1: setNumRandom()
1.2: setNumeroRandom()
alt
[giocatori[0].num > giocatori[1].num]
2: setColore(b : char)
3: setTurno(true : Boolean)
4: setColore(n : char)
5: setTurno(false : Boolean)
[else]
6: setColore(b : char)
7: setTurno(true : Boolean)
8: setColore(n : char)
9: setTurno(false : Boolean)

- **createDamiera**



2.3.2 DCD:

Sono state identificate le seguenti classi progettuali:

Giocatore;

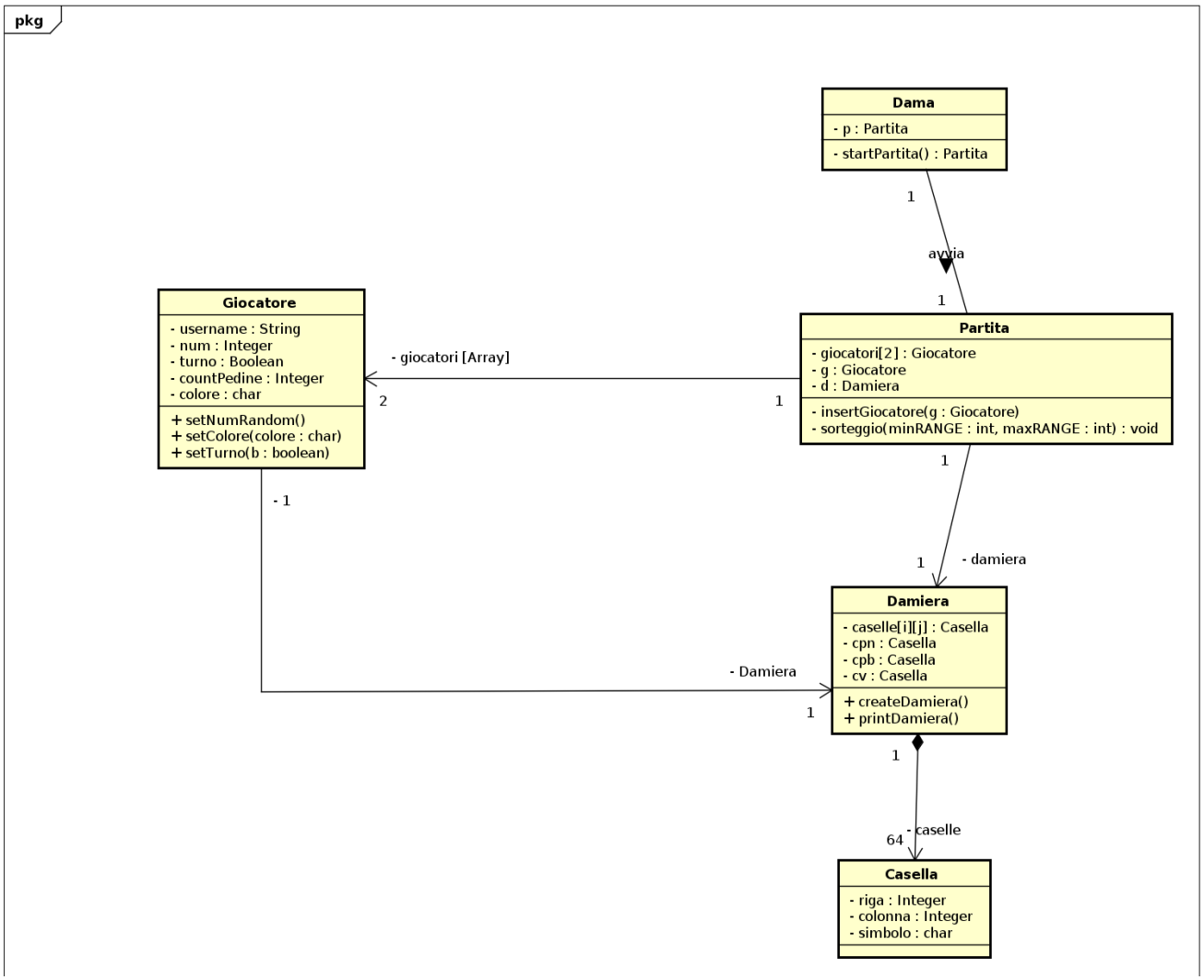
Dama: rappresenta il sistema Dama complessivo. È il primo oggetto a ricevere e coordinare una operazione di sistema. Per questo motivo la scelta del Facade Controller ricade sulla classe Dama;

Partita: Controller del caso d'uso Nuova Partita

Casella;

Damiera;

La classe ***Pedina*** non è stata inserita poichè il gioco non prevede la gestione di pedine differenti da quelli della dama. Inoltre dato che Pedina avrebbe avuto un solo attributo in più rispetto la classe Casella (simbolo), è stata maturata la decisione di integrare tale attributo nella classe Casella.



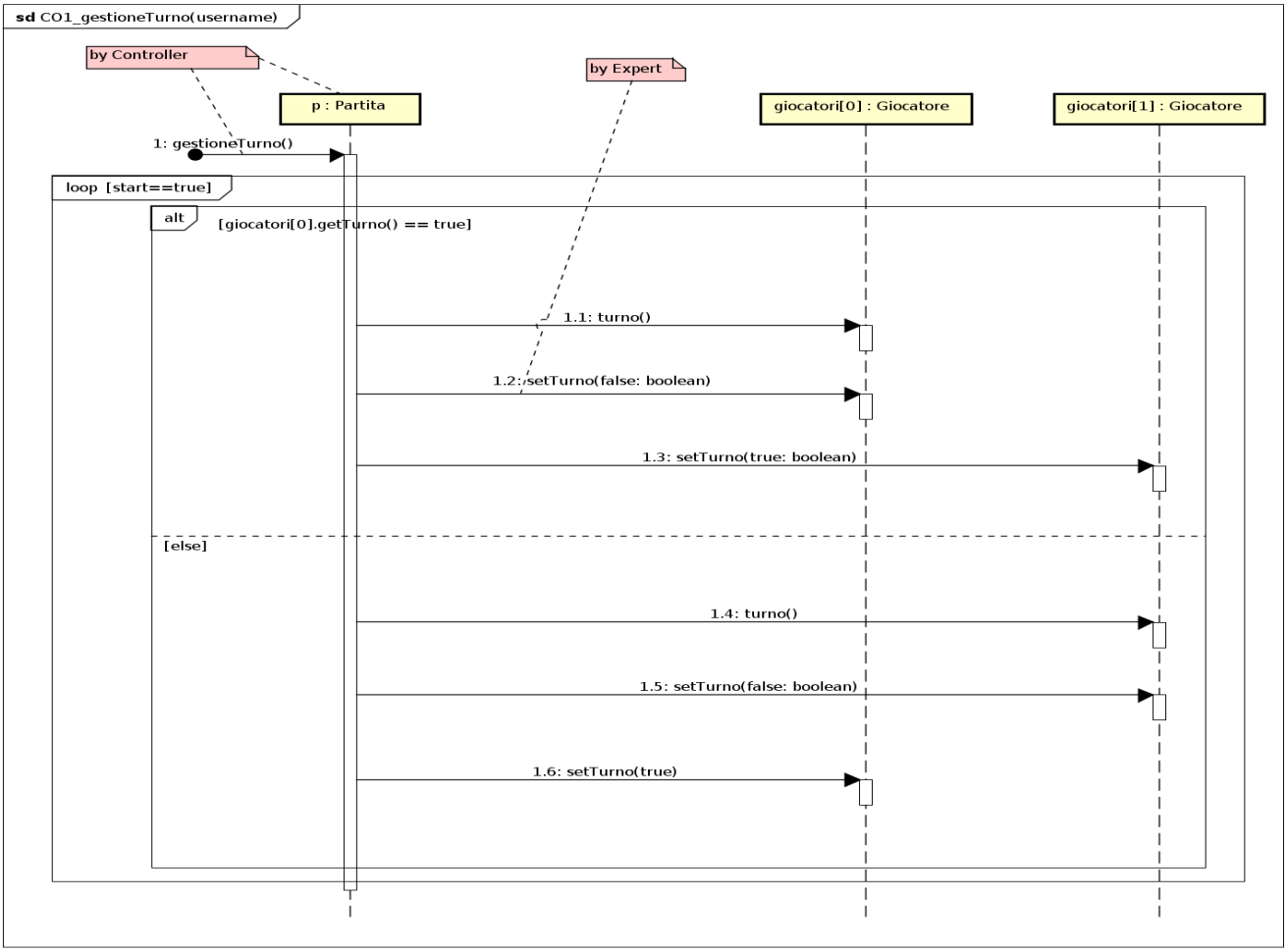
2.4 UC2: Modello di dominio, Gestione Turno

2.4.2 Contratti delle operazioni

Contratto CO1: gestioneTurno
Riferimenti: UC2
Operazione: gestioneTurno()
Precondizioni: - Entrambi i giocatori hanno un numero di pedine diverso da zero
- Entrambi i giocatori hanno l'attributo turno settato a true o false
PostCondizioni: -viene assegnato il turno al giocatore con attributo turno settato a true
-L'attributo turno del giocatore corrente viene settato false
-L'attributo turno dell'avversario viene settato true

2.5 UC2: Progettazione orientata agli oggetti

2.5.1 Diagramma di Interazione:



2.5.2 DCD:

Sono state identificate le seguenti classi progettuali:
Giocatore;
Partita: Controller del caso d'uso Gestione Turno. Viene assegnata la responsabilità del ciclo di gioco;

2.6 UC3: Aggiornamento analisi dei requistiti, Turno

- Portata: Applicazione Dama
- Attore primario: Giocatore 1 / Giocatore 2
- Parti interessate :
  - Giocatore 1 o Giocatore 2
- Precondizioni:
  - La partita è in esecuzione
  - Giocatore 1 o Giocatore 2 ha il turno abilitato
- Scenari :
  - **3.1 Mossa semplice**
  - **3.2 Mossa con presa**
  - **3.3 Mossa con presa multipla**
- Garanzia di successo:
  - Viene aggiornato visivamente lo scenario di gioco

3.1 Mossa semplice

- Il giocatore inserisce la coordinata della propria ***pedina*** da muovere
- Il giocatore inserisce la coordinata della casella accessibile per la destinazione della pedina
- Il sistema verifica e abilita la mossa
- La posizione della pedina viene aggiornata

- *Estensioni:*
  - Il giocatore inserisce una coordinata di **origine** errata:
    a) la coordinata non coincide con una propria pedina
    b) la coordinata inserita coincide con una propria pedina, ma tale pedina non ha una casella successiva vuota nella diagonale in avanti (o indietro)
  - Il giocatore inserisce una coordinata di **destinazione** non corrispondente ad una casella successiva vuota nella diagonale in avanti (o indietro)

3.2 Mossa con presa

- Il giocatore seleziona la coordinata della propria pedina da muovere
- Il giocatore seleziona la coordinata di una ***casella*** libera, successiva ad un pedina avversaria vicina, nella direzione diagonale in avanti.
- Il numero delle pedine del giocatore che ha subito la presa viene decrementato.
- La posizione della pedina mangiante verrà aggiornata, mentre la pedina mangiata verrà tolta dalla damiera.

- *Estensioni:*
  - Il giocatore inserisce una coordinata di **origine** errata:
    a) la coordinata non coincide con una propria pedina
  - Il giocatore inserisce una coordinata di **destinazione** errata:
    a) la pedina di origine può effettuare sia una mossa semplice che una mossa con presa: il giocatore seleziona la coordinata per effettuare una mossa semplice
    b) Il giocatore inserisce una coordinata di destinazione non corrispondente ad una casella libera, successiva ad un pedina avversaria vicina, nella direzione diagonale

3.3 Mossa con presa multipla

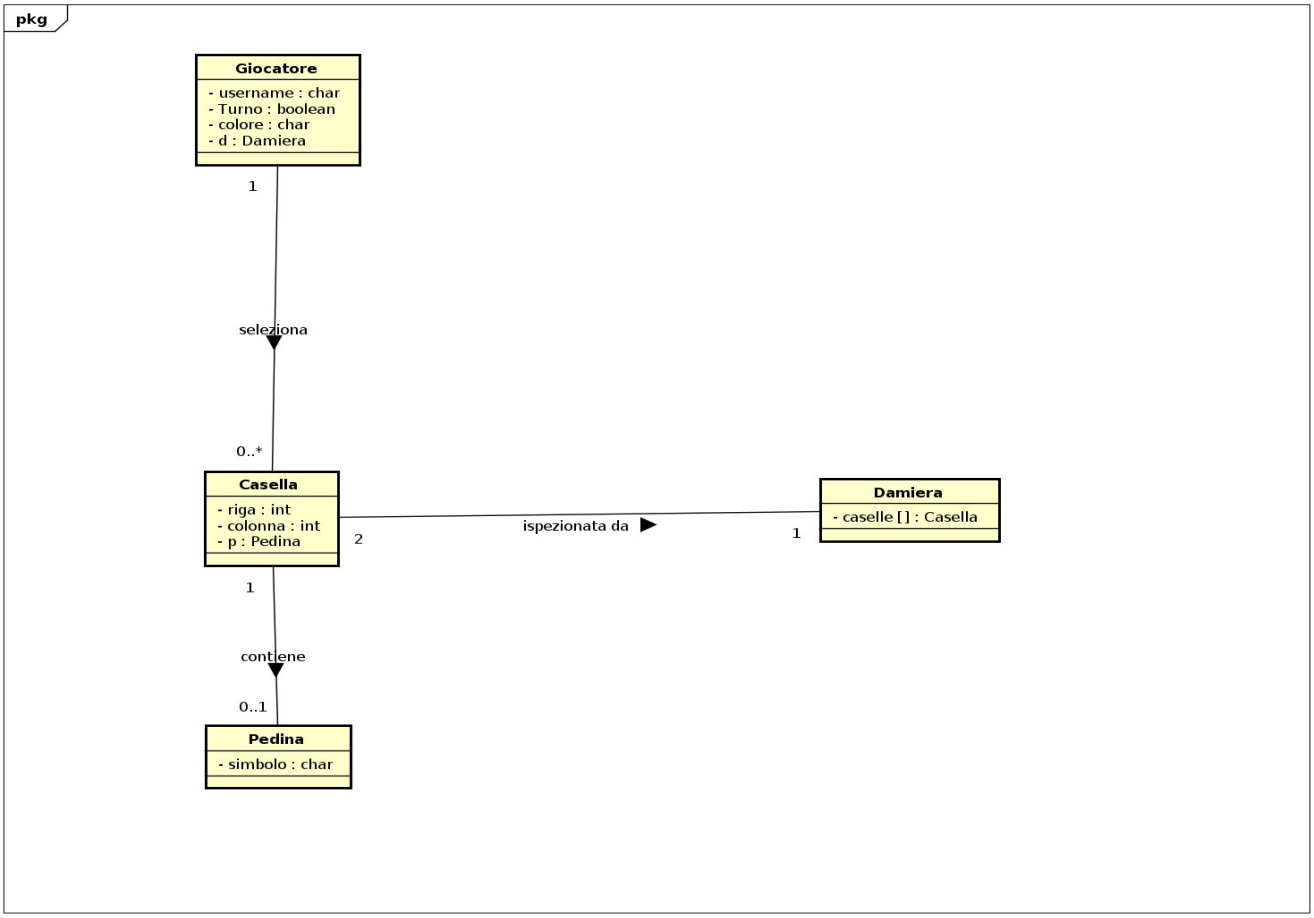
- Il giocatore ha appena effettuato una mossa con presa
- La pedina incontra in diagonale *un' altra* pedina avversaria con la successiva ***casella accessibile*** libera (nella direzione diagonale in avanti).
- Il giocatore seleziona la successiva destinazione fino al verificarsi della precedente condizione.
- Il numero delle pedine del giocatore che ha subito la presa multipla viene decrementato in base al numero di prese subite.
- La posizione della pedina mangiante verrà aggiornata, mentre le pedine mangiate verranno tolte dalla damiera.

- *Estensioni:*

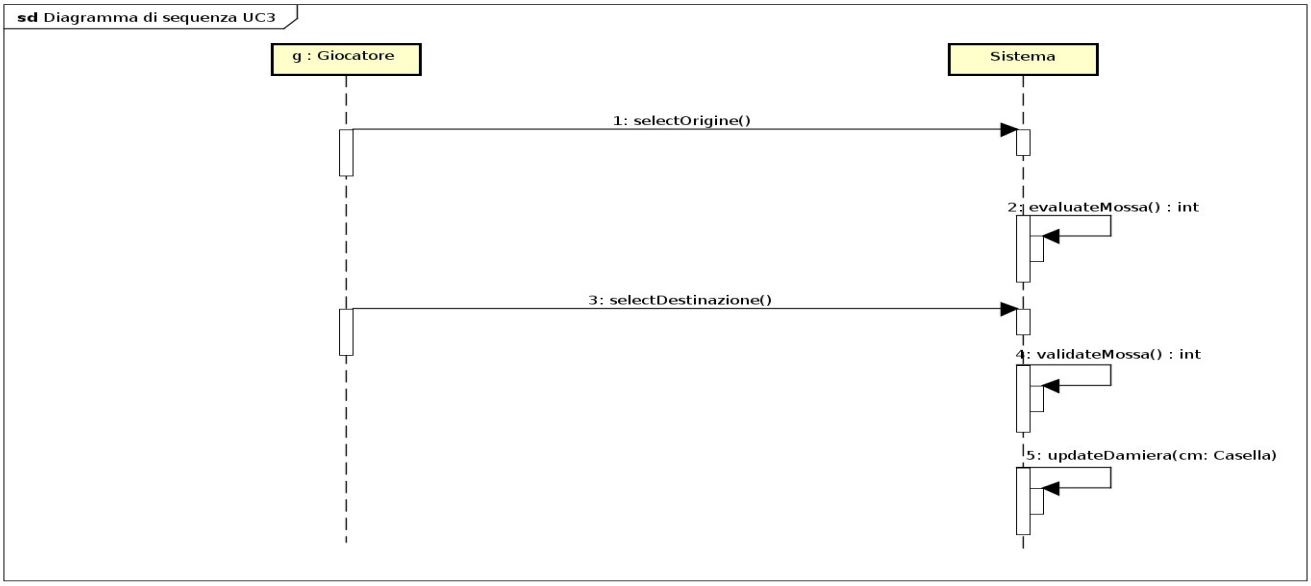
  -Il giocatore inserisce una coordinata di destinazione errata:

  La coordinata di destinazione non corrispondente ad una casella libera, successiva ad un pedina avversaria vicina, nella direzione diagonale

2.6 UC3: Modello di dominio, Turno



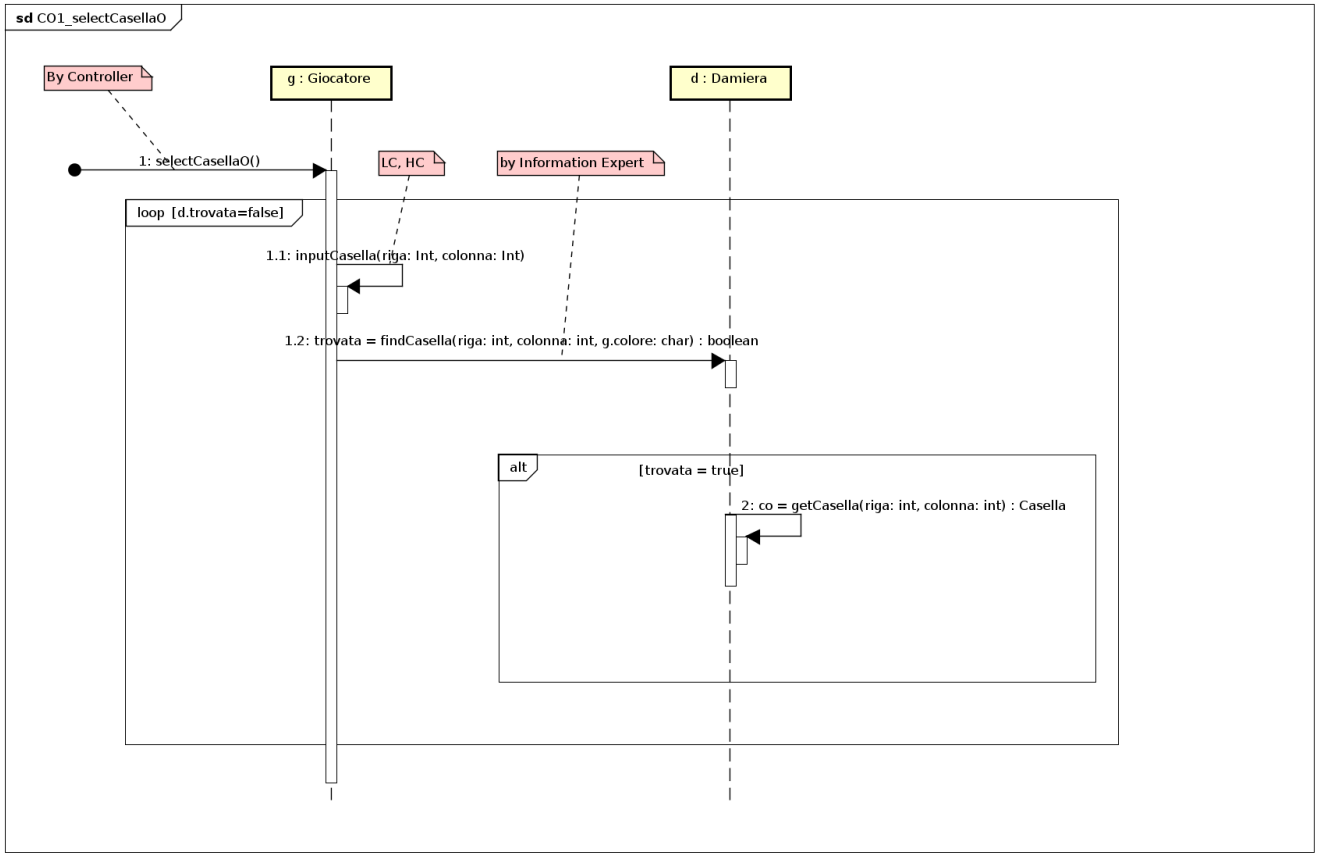
2.6.1 Diagramma di Sequenza SSD



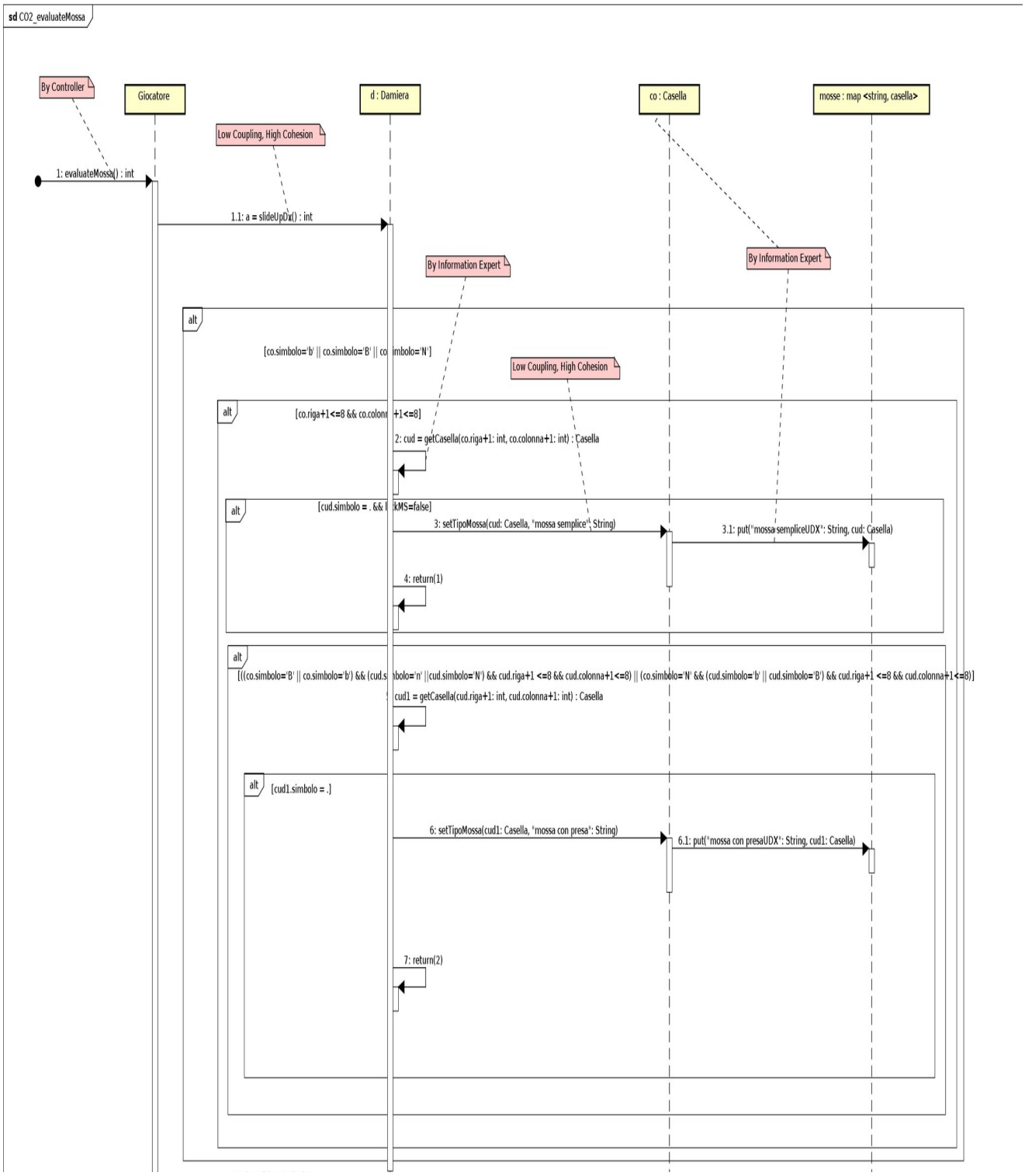
2.7 UC3: Progettazione orientata agli oggetti

2.7.1 Diagrammi di Interazione:

- **selectCasellaO**

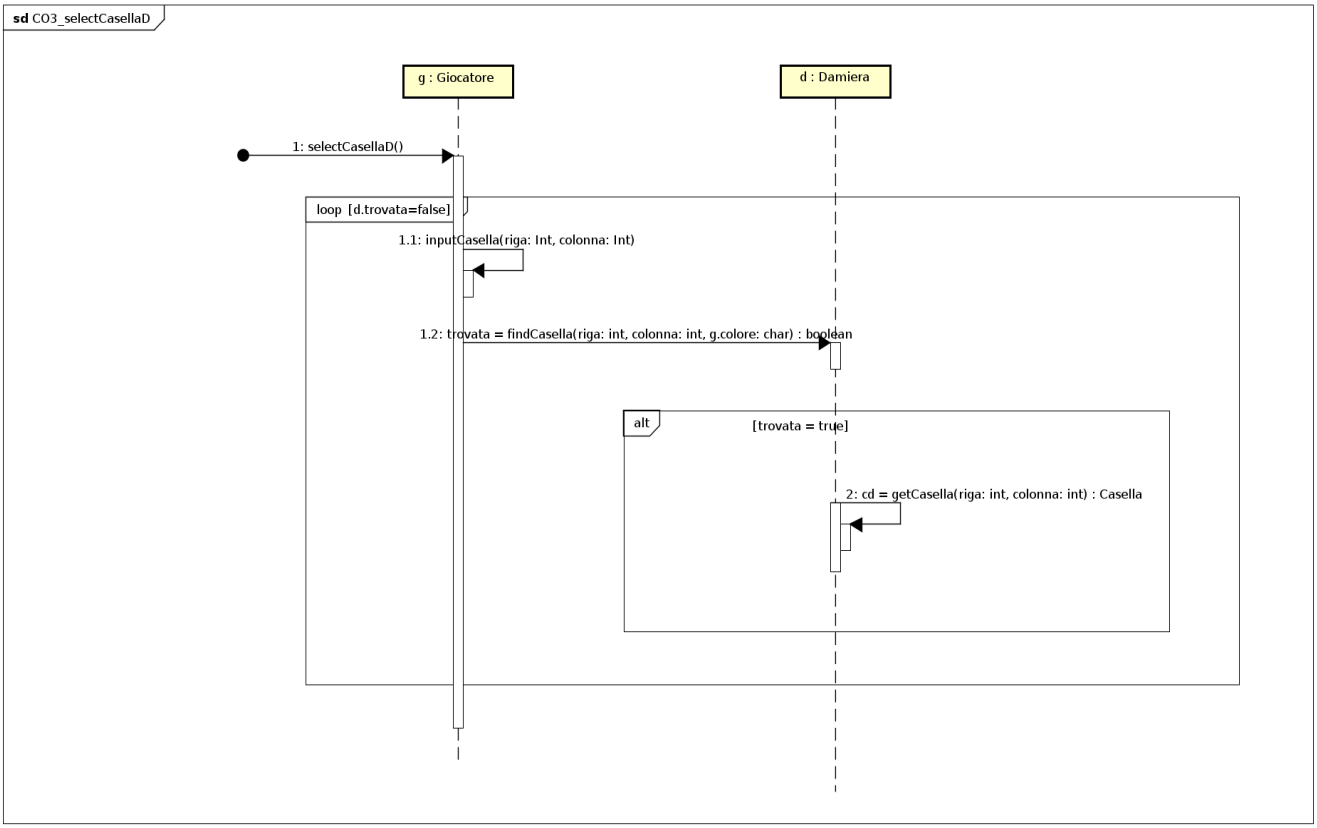


- **evaluateMossa**

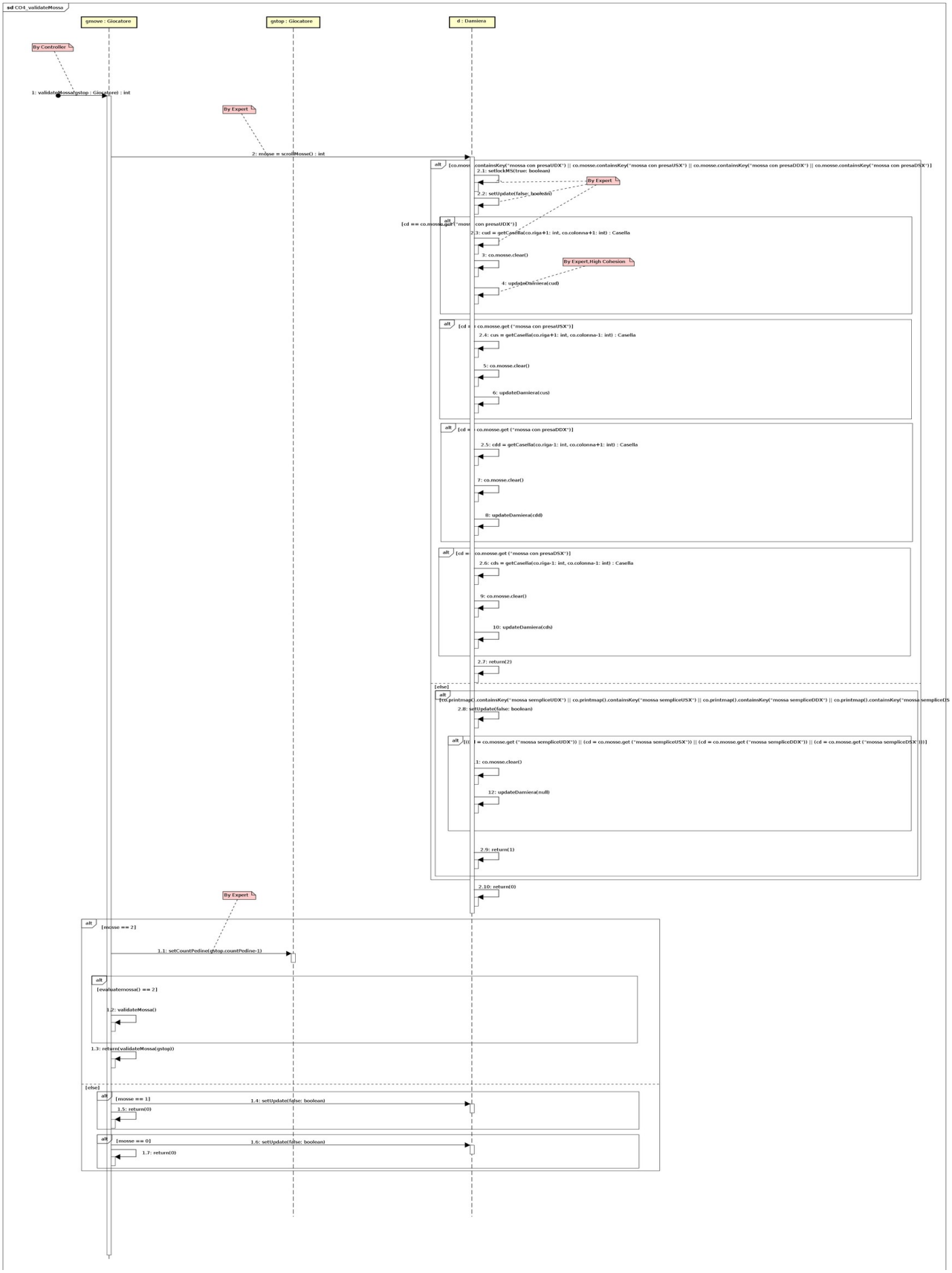


(Per completezza consultare il file asta in Dama_DOC/Iterazione1/UC3/UC3_OO.asta)

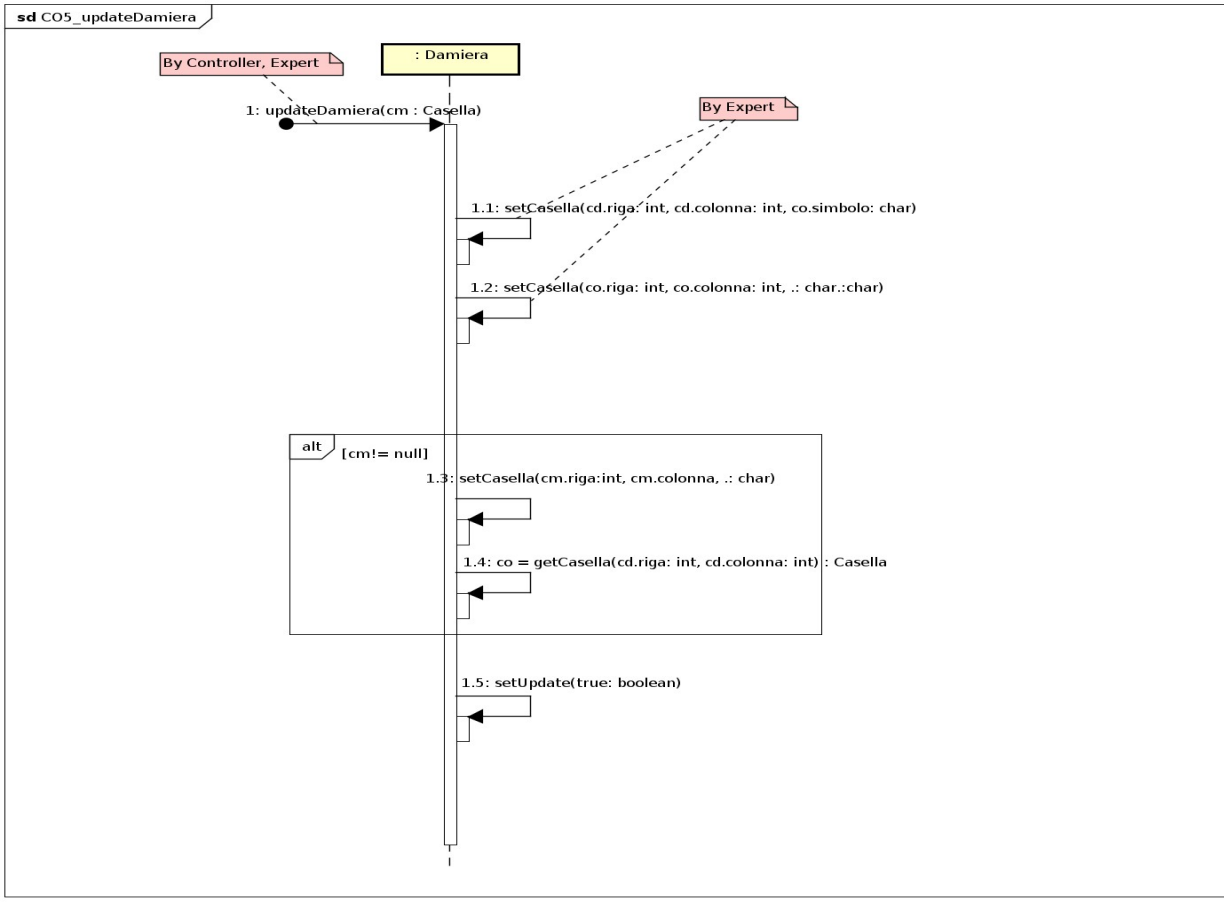
- **selectCasellaD**



- **validateMossa**



- **updateDamiera**



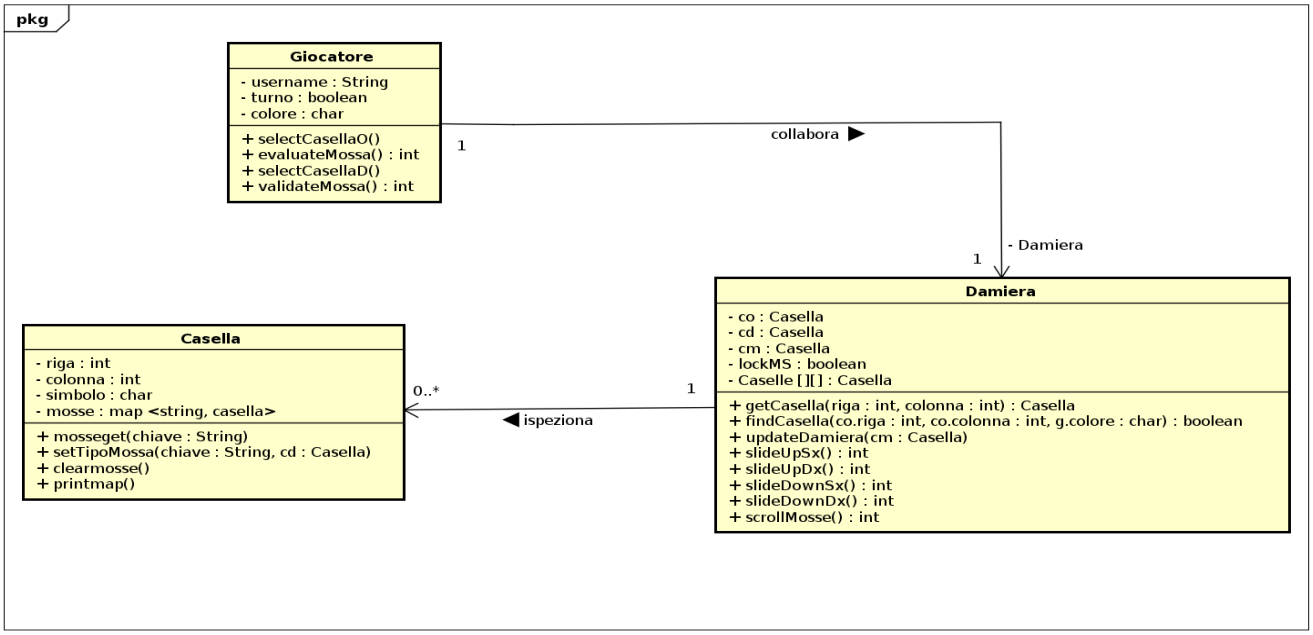
2.7.2 DCD:

Sono state identificate le seguenti classi progettuali:

Giocatore: Controller del caso d'uso Turno, poichè è il responsabile di effettuare un turno;

Damiera;

Casella;



2.8 Casi d’uso UC1, UC2, UC3 Diagramma delle classi

Il seguente diagramma delle classi, è stato sviluppato tenendo in considerazione i tre casi d’uso presi in esame durante quest’iterazione. Viene scelto di usare il design pattern **Singleton** (GoF) per la classe Partita e per la classe Damiera.