# 3 – Elaborazione – Iterazione 2

## 3.1 Introduzione

In questa iterazione della fase di elaborazione sono stati implementati tutti gli scenari di successo di tutti i casi d'uso rimanenti, da UC2 a UC6 e le estensioni di tutti i casi d'uso.

## 3.2 Casi d'uso in dettaglio

Di seguito verranno presentati nuovamente tutti i casi d'uso, in dettaglio.

### 3.2.1 UC1: Avviare una nuova partita

| **Nome del caso d'uso** | **UC1: Avviare una nuova partita** |
|---|---|
| **Portata** | Applicazione DigiGoose! |
| **Livello** | Obiettivo utente |
| **Attore primario** | Giocatore |
| **Parti interessate e interessi** | Giocatore: vuole iniziare e gestire una partita del gioco dell'oca con un numero variabile di partecipanti, scegliendo tra giocatori umani e controllati dal computer, colori delle pedine e i loro nomi. |
| **Precondizioni** | L'applicazione è avviata. |
| **Garanzia di successo** | La partita viene avviata con successo, con il numero di giocatori desiderato (umani e computer), l'ordine del gioco determinato, le pedine col colore scelto per giocatore alla posizione iniziale (0) e il sistema pronto a ricevere l'interazione del primo giocatore. |
| **Scenario principale di successo** | 1. Il giocatore seleziona l'opzione "Nuova Partita".<br>2. Il sistema visualizza le opzioni per impostare la partita, ossia:<br>a. Numero di giocatori (2 – 6);<br>b. Scelta tra umano e computer per ogni giocatore<br>c. Inserimento del nome di ciascun giocatore umano<br>d. Scelta del colore della pedina;<br>3. Il giocatore seleziona il numero di giocatori desiderato e configura se è un umano o computer per ciascuno.<br>4. Per i giocatori umani, il giocatore inserisce i nomi nei campi appositi. Il sistema valida che il nome non sia vuoto o contiene caratteri non validi.<br>5. Il giocatore seleziona un colore univoco per la propria pedina. Il sistema assicura che ogni giocatore abbia un colore diverso.<br>6. Il giocatore conferma le impostazioni della partita.<br>7. Il sistema inizializza la partita con le impostazioni scelte.<br>8. Il sistema determina casualmente quale giocatore inizia la partita.<br>9. Il sistema visualizza il tabellone di gioco, le pedine posizionate alla casella 0, e i nomi dei giocatori, indicando il giocatore a cui tocca il suo turno. Il sistema aggiorna sessione corrente (non verrà |

| | |
|---|---|
| | applicata in questa elaborazione, ma la creazione di una nuova partita deve sovrascrivere la sessione salvata per “riprendi partita”) |
| **Estensioni** | 1) In qualsiasi momento, l'applicazione si blocca:<br>a) Il giocatore riavvia l'applicazione.<br>2) Il giocatore seleziona opzioni di gioco non valide:<br>a) Il sistema visualizza un messaggio di errore.<br>b) Il giocatore modifica le opzioni e riprova.<br>3) Il giocatore annulla la creazione della partita:<br>a) Il sistema torna alla schermata principale. |
| **Requisiti speciali** | 1) Interfaccia utente intuitiva per la selezione delle opzioni di gioco.<br>2) Il sistema deve assicurare l’univocità del colore delle pedine.<br>3) Il sistema deve fornire un feedback chiaro in caso di input non validi. |
| **Elenco delle varianti tecnologiche e dei dati** | Dati: impostazioni della partita (numero di giocatori, nomi dei giocatori, tipo di giocatori, colori delle pedine)<br>Metodi: selezione del primo giocatore casuale, la posizione delle pedine deve essere sulla casella 0. |
| **Frequenza delle ripetizioni** | Ogni volta che un giocatore vuole iniziare una nuova partita. |
| **Varie** | L'implementazione corrente include codice per il salvataggio/caricamento (commentato in GestoreSalvataggio), ma questa funzionalità non è esposta nell'interfaccia utente e una nuova partita sovrascrive effettivamente qualsiasi stato precedente nella logica attuale della UI. |

### 3.2.2 UC2: Tirare i dadi

| **Nome del caso d’uso** | **UC2: Tirare i dadi** |
|---|---|
| **Portata** | Partita in corso |
| **Livello** | Sotto-funzione (parte del turno di un giocatore) |
| **Attore primario** | Giocatore corrente |
| **Parti interessate e interessi** | Giocatore corrente: vuole ottenere il risultato del lancio per determinare il movimento.<br>Altri giocatori: il risultato del lancio potrebbe sbloccare condizioni che li tengono fermi.<br>Sistema: ha bisogno del risultato per calcolare il movimento e applicare effetti. |
| **Precondizioni** | La partita è in stato IN_CORSO.<br>È il turno del giocatore corrente.<br>Il giocatore corrente non è bloccato per un numero fisso di turni (getTurniSaltati() > 0). Può essere bloccato da una condizione (getTurniSaltati() < 0). |

| | |
|---|---|
| **Garanzia di successo** | I dadi vengono simulati e lanciati. Il risultato è determinato e visualizzato. Eventuali condizioni di blocco di giocatori (incluso il giocatore corrente se bloccato condizionalmente) che dipendono dal risultato del lancio vengono verificate e rimosse se soddisfatte. |
| **Scenario principale di successo** | 1. È il turno del giocatore corrente.<br>2. Il giocatore:<br>a. Se è umano, il sistema gli chiede di lanciare i dadi.<br>b. Se è CPU, il sistema procede autonomamente con il lancio, chiedendo conferma all'utente per il passaggio al turno successivo.<br>3. Il sistema simula il lancio di 2 dadi a 6 facce.<br>4. Il sistema comunica il risultato di ciascun dado e la loro somma al giocatore/utente.<br>5. Il sistema verifica se il risultato del lancio soddisfa le condizioni di sblocco per qualsiasi giocatore bloccato (getTurniSaltati() < 0, es. attesa di un 6, attesa di 3/6, attesa di 4/5, prigione).<br>6. Per ogni giocatore le cui condizioni di blocco sono soddisfatte dal lancio, il sistema rimuove lo stato di blocco (setTurniSaltati(0)) e informa l'utente (se umano).<br>7. Il giocatore corrente:<br>a. Se era bloccato condizionalmente e *non* è stato sbloccato da questo lancio, il suo turno finisce qui senza muoversi. La gestione del passaggio al giocatore successivo avverrà in UC5.<br>b. Se era libero o è stato sbloccato da questo lancio, il sistema procederà al movimento - UC3). |
| **Estensioni** | 1. Il giocatore umano sceglie un'opzione diversa dal lanciare i dadi (es. Esci):<br>a. L'UC termina e il flusso passa alla gestione dell'opzione scelta (es. ritorno al menu principale).<br>2. Si verifica un errore tecnico durante la simulazione del lancio dei dadi:<br>a. Il sistema si blocca o gestisce l'errore (rif. UC1 Estensione 1). |
| **Requisiti speciali** | Il risultato dei dadi deve essere generato in modo casuale.<br>Il sistema deve visualizzare chiaramente il risultato del lancio.<br>Il sistema deve verificare e gestire le condizioni di sblocco per i giocatori bloccati (turniSaltati < 0) basate sul risultato del lancio. |
| **Elenco delle varianti tecnologiche e dei dati** | Dati: valore dei due dadi.<br>Metodi: generazione casuale dei dadi (Dadi.lancia(), Random), verifica condizioni di sblocco (PartitaController.verificaELiberaGiocatoriBloccati()). |
| **Frequenza delle ripetizioni** | Una volta per ogni turno di un giocatore, a meno che non debba saltare il turno per un numero fisso (turniSaltati > 0). |
| **Varie** | |

### 3.2.3 UC3: Muovere la pedina

| Nome del caso d'uso | UC3: Muovere la pedina |
|---|---|
| **Portata** | Partita in corso |
| **Livello** | Sotto-funzione (parte del turno di un giocatore) |
| **Attore primario** | Sistema (ma agisce per conto di Giocatore) |
| **Parti interessate e interessi** | Giocatore corrente: vede la sua pedina avanzare sul tabellone.<br>Tabellone: ospita la pedina nella sua nuova posizione. |
| **Precondizioni** | UC2 (Tirare i dadi) è stato completato con successo.<br>Il giocatore corrente non è bloccato da una condizione (getTurniSaltati() >= 0 dopo la verifica in UC2).<br>Il risultato totale del lancio dei dadi è disponibile. |
| **Garanzia di successo** | La pedina del giocatore corrente viene spostata sulla casella calcolata in base al risultato dei dadi. La nuova posizione rispetta le regole di gioco, inclusa la gestione del superamento della casella finale (63) tramite "rimbalzo". Il sistema identifica la casella di destinazione. |
| **Scenario principale di successo** | 1. 1. Il sistema ottiene la somma dei valori dei dadi lanciati in UC2.<br>2. Il sistema ottiene la posizione corrente della pedina del giocatore corrente.<br>3. Il sistema calcola la potenziale nuova posizione sommando la posizione corrente al totale dei dadi.<br>4. Il sistema verifica se la potenziale nuova posizione supera la casella finale (63):<br>a. Se la potenziale nuova posizione *non* supera 63, la nuova posizione è quella calcolata al passo 3.<br>b. Se la potenziale nuova posizione *supera* 63, il sistema calcola la nuova posizione "rimbalzando" indietro da 63 del numero di caselle in eccesso rispetto a 63.<br>5. Il sistema aggiorna la posizione della pedina del giocatore alla nuova posizione finale calcolata.<br>6. Il sistema identifica la casella corrispondente alla nuova posizione sul tabellone.<br>7. Il sistema comunica la nuova posizione del giocatore all'utente.<br>8. La gestione degli effetti speciali di questa casella avviene in UC4. |
| **Estensioni** | 1. La somma dei dadi non è disponibile o valida.<br>a. UC3 non può essere eseguito. Si verifica un errore di sistema ancora da gestire. |
| **Requisiti speciali** | La logica di movimento deve gestire correttamente il "rimbalzo" oltre la casella 63. |

| | |
|---|---|
| **Elenco delle varianti tecnologiche e dei dati** | Dati: posizione corrente della pedina, somma dei dadi.<br>Metodi: calcolo nuova posizione, aggiornamento posizione pedina. |
| **Frequenza delle ripetizioni** | Una volta per ogni turno giocato (non saltato per turniSaltati > 0 e sbloccato se turniSaltati < 0). |
| **Varie** | |

### 3.2.4 UC4: Gestire le caselle speciali

| **Nome del caso d'uso** | **UC4: Gestire le caselle speciali** |
|---|---|
| **Portata** | Partita in corso |
| **Livello** | Sotto-funzione (parte del turno di un giocatore) |
| **Attore primario** | Sistema (ma agisce per conto di Giocatore) |
| **Parti interessate e interessi** | Giocatore corrente: subisce o beneficia dell'effetto della casella.<br>Tabellone: contiene le definizioni delle caselle speciali.<br>Altri giocatori: potrebbero essere influenzati indirettamente (es. effetto che fa saltare un turno). |
| **Precondizioni** | UC3 (Muovere la pedina) è stato completato con successo.<br>La pedina del giocatore corrente è arrivata su una casella.<br>Il risultato del lancio dei dadi del turno corrente è disponibile. |
| **Garanzia di successo** | Se la casella su cui è arrivato il giocatore è una casella speciale (diversa dalla casella 63), l'effetto associato viene applicato al giocatore (e potenzialmente ad altri) secondo le regole del gioco.<br>Lo stato o la posizione del giocatore possono essere modificati dall'effetto.<br>Il sistema comunica l'effetto all'utente. |
| **Scenario principale di successo** | 1. Il sistema identifica la casella di destinazione sulla quale è arrivata la pedina del giocatore corrente.<br>2. Il sistema verifica se la casella è la casella finale (63):<br>a. Se sì, viene avviato UC6.<br>b. Se la casella *non* è la casella 63, il sistema verifica se la casella è speciale.<br>3. Se la casella:<br>a. *non* è speciale, UC4 termina.<br>b. Se è speciale, il sistema determina il tipo di effetto.<br>4. Il sistema applica l'effetto base associato al tipo di casella speciale. Questo potrebbe modificare la posizione o lo stato del giocatore.<br>5. Il sistema verifica se l'effetto richiede un'ulteriore gestione basata sul risultato dei dadi (es. Raddoppia Movimento).<br>6. Se l'effetto richiede l'interazione con i dadi, il sistema applica la logica specifica utilizzando il risultato dei dadi del turno |

| | |
|---|---|
| | corrente. Questo potrebbe comportare un ulteriore spostamento della pedina.<br>7. Il sistema comunica il tipo di effetto applicato e la potenziale nuova posizione del giocatore all'utente. |
| **Estensioni** | 1. La casella di destinazione non è valida.<br>a. UC4 non può essere eseguito correttamente. Si verifica un errore di sistema da gestire.<br>2. Si verifica un errore durante l'applicazione della logica dell'effetto speciale.<br>a. Il sistema gestisce l'errore (rif. UC1 Estensione 1). |
| **Requisiti speciali** | Deve essere implementata correttamente la logica per ogni tipo di casella speciale.<br>La posizione o lo stato del giocatore devono essere aggiornati secondo l'effetto.<br>Gli effetti che dipendono dal lancio dei dadi devono utilizzare il risultato corretto. |
| **Elenco delle varianti tecnologiche e dei dati** | Dati: tipo di effetto casella, posizione giocatore, stato giocatore, risultato dadi.<br>Metodi: applicazione effetti, accesso casella. |
| **Frequenza delle ripetizioni** | Una volta per ogni turno giocato (non saltato per turniSaltati > 0 e sbloccato se turniSaltati < 0), se la casella di destinazione è speciale (diversa dalla 63). |
| **Varie** | |

### 3.2.5 UC5: Gestire i turni

| **Nome del caso d'uso** | **UC5: Gestire i turni** |
|---|---|
| **Portata** | Partita in corso |
| **Livello** | Sistema (coordinamento del flusso di gioco) |
| **Attore primario** | Sistema |
| **Parti interessate e interessi** | Tutti i giocatori: attendono il proprio turno e osservano lo stato del gioco. |
| **Precondizioni** | La partita è in stato IN_CORSO.<br>Il turno del giocatore precedente è terminato (ha completato il movimento e gli effetti, o ha saltato il turno per turniSaltati > 0). |
| **Garanzia di successo** | Il sistema identifica correttamente il giocatore successivo nell'ordine stabilito. Se il giocatore successivo deve saltare il turno per un numero fisso (turniSaltati > 0), questo turno viene gestito come "saltato" e il sistema passa immediatamente al giocatore ancora successivo, aggiornando il conteggio dei turni da saltare per quel giocatore. |

| | |
|---|---|
| | Il giro e il turno vengono incrementati correttamente all'inizio di ogni nuovo giro e per ogni giocatore effettivo.<br>Il sistema presenta l'interfaccia/logica per il giocatore corrente che può agire. |
| **Scenario principale di successo** | 1. Il turno del giocatore corrente (precedente) si conclude.<br>2. Il sistema verifica se esso è rimasto bloccato condizionalmente in UC2 o UC4.<br>3. Se *non* è bloccato condizionalmente, il sistema identifica il giocatore successivo nell'ordine di gioco predefinito.<br>4. Il sistema verifica se questo prossimo giocatore deve saltare il turno per un numero fisso.<br>5. Se il prossimo giocatore deve saltare il turno:<br>a. Il sistema informa l'utente che il giocatore salta il turno.<br>b. Il sistema decrementa il contatore dei turni da saltare per quel giocatore.<br>c. l sistema ripete i passi 3 e 4 per trovare il giocatore successivo nell'ordine (questo ciclo continua finché non si trova un giocatore che *non* deve saltare il turno).<br>6. Se il prossimo giocatore *non* deve saltare il turno per un numero fisso, questo giocatore diventa il giocatore corrente per il nuovo turno.<br>7. Il sistema incrementa il contatore del turno. Se questo giocatore è il primo nell'ordine del giro, il sistema incrementa il contatore del giro.<br>8. Il sistema presenta l'interfaccia utente/logica per consentire al giocatore corrente di eseguire il suo turno (avvia UC2).<br>9. Se il giocatore precedente *era* bloccato condizionalmente, rimane il giocatore corrente per la prossima iterazione del ciclo di gioco principale, e il flusso riprende da UC2 per lui, senza passare al giocatore successivo nell'ordine finché non viene sbloccato. |
| **Estensioni** | 1. L'ordine dei giocatori non è disponibile o è vuoto.<br>a. UC5 non può essere eseguito. Si verifica un errore di sistema da gestire. |
| **Requisiti speciali** | La gestione dei turni deve seguire l'ordine predeterminato dei giocatori.<br>La logica deve distinguere correttamente tra giocatori che saltano turni fissi e giocatori bloccati da condizioni.<br>I contatori di turno e giro devono essere aggiornati correttamente. |
| **Elenco delle varianti tecnologiche e dei dati** | Dati: lista dell'ordine dei giocatori.<br>Metodi: passaggio turno, verifica e decremento dei turni saltati, logica di controllo del ciclo principale. |
| **Frequenza delle ripetizioni** | Una volta alla fine di ogni turno effettivo (non saltato) o una volta per ogni giocatore che deve saltare un turno fisso. |
| **Varie** | |

### 3.2.6 UC6: Determinare il vincitore

| Nome del caso d'uso | UC6: Determinare il vincitore |
|---|---|
| **Portata** | Partita in corso |
| **Livello** | Sistema |
| **Attore primario** | Sistema |
| **Parti interessate e interessi** | Tutti i giocatori: vogliono sapere chi ha vinto la partita e che il gioco è finito. |
| **Precondizioni** | La partita è in stato IN_CORSO.<br>Un giocatore ha completato il suo turno (ha mosso e gli effetti, se presenti, sono stati applicati). |
| **Garanzia di successo** | Se un giocatore raggiunge esattamente la casella 63 alla fine del suo turno, viene dichiarato vincitore e lo stato della partita cambia a TERMINATA. |
| **Scenario principale di successo** | 1. Il turno del giocatore corrente si conclude (dopo UC4).<br>2. Il sistema verifica la posizione finale del giocatore corrente.<br>3. Il sistema verifica se la posizione del giocatore corrente è esattamente 63.<br>4. Se la posizione è esattamente 63:<br>a. Il sistema identifica il giocatore corrente come il vincitore.<br>b. Il sistema cambia lo stato della partita a TERMINATA.<br>c. Il sistema informa l'utente che un vincitore è stato determinato e quale giocatore ha vinto.<br>d. Il ciclo di gioco principale termina.<br>e. L'interfaccia utente ritorna al menu principale.<br>5. Se la posizione *non* è 63 e lo stato della partita non è già TERMINATA, la partita continua. |
| **Estensioni** | 1. Più giocatori raggiungono la casella 63 contemporaneamente:<br>a. la logica corrente verifica il vincitore dopo il turno *di ciascun* giocatore, garantendo che il primo a raggiungere 63 venga identificato.<br>2. La partita termina per un altro motivo.<br>a. Il sistema gestisce l'errore; non viene dichiarato un vincitore secondo le regole del gioco. |
| **Requisiti speciali** | La condizione di vittoria deve essere l'arrivo *esattamente* sulla casella 63.<br>L'identificazione del vincitore deve avvenire subito dopo che un giocatore completa il suo movimento e l'applicazione degli effetti. |
| **Elenco delle varianti tecnologiche e dei dati** | Dati: posizione del giocatore, stato della partita.<br>Metodi: verifica vincitore, impostazione stato partita. |

| | |
|---|---|
| **Frequenza delle ripetizioni** | Almeno una volta alla fine di ogni turno giocato (non saltato), fino a che lo stato della partita non diventa TERMINATA |
| **Varie** | |

## 3.3 Changelog completo dell'elaborazione

Questa elaborazione introduce significativi miglioramenti alle meccaniche di gioco e alla stabilità dell'interfaccia utente, allineandosi più fedelmente alle regole classiche del gioco dell'oca. La funzionalità di salvataggio è stata temporaneamente rimossa per riprogettazione nella prossima iterazione.

### 3.3.1 Modifiche principali

#### *3.3.1.1 Configurazione partita*

- Miglioramenti all'UI console:
  - Gestione avanzata degli errori di input (es. NumberFormatException, IllegalArgumentException).
  - Utilizzo di nextLine() combinato con Integer.parseInt() per una lettura più affidabile.
  - Messaggi di errore più chiari con ripristino dello stato in caso di fallimento.
- Aggiunti controlli in ImpostazioniPartita per:
  - Nomi vuoti (IllegalArgumentException).
  - Valori fuori intervallo (IndexOutOfBoundsException).
  - Assegnazione automatica di colori e tipi di giocatore (CPU/umano).

#### *3.3.1.2 Gestione giri e turni*

- Introduzione del concetto di giro:
  - Tracciamento del giroCorrente nella classe Partita.
  - Incremento automatico quando il turno ritorna al primo giocatore.
- Differenziazione tra:
  - Blocchi temporali (turniSaltati > 0, es. Pozzo = 3 turni).
  - Blocchi condizionali (turniSaltati < 0, es. Prigione = attesa di un "6" sul dado).

#### *3.3.1.3 Meccanica di sblocco giocatore*

- Nuova Funzionalità:
  - Metodo verificaELiberaGiocatoriBloccati(int[] valoreDadi) per verificare automaticamente se un lancio di dadi sblocca giocatori fermi in caselle speciali (es. un "6" libera dalla Prigione).

#### *3.3.1.4 Movimento ed effetti caselle*

- Redesign del tabellone:
  - Assegnazione precisa di effetti a caselle specifiche (Ponte → casella 12, Labirinto → casella 35, ecc.).
  - Nuovi tipi di TipoEffettoCasella per una gestione più dettagliata delle regole.
- Separazione della logica degli effetti:
  - Casella.applicaEffetto(): gestisce effetti fissi (spostamenti, blocchi semplici).
  - PartitaController.gestisciEffettiSpeciali(): gestisce effetti dipendenti dai dadi (es. Raddoppio Movimento).

#### *3.3.1.5 Interfaccia utente*

- Visualizzazione migliorata:
  - Aggiunta di indicatori per "Giro X" e "Turno Y".
  - Dettagli sullo stato dei giocatori bloccati (es. "Giocatore: fermo per 2 turni (Pozzo)").
- Ottimizzazione del flusso di gioco:
  - Salti automatici dei turni bloccati con notifica.
  - Messaggi espliciti per i blocchi condizionali (es. "Lancia un 6 per liberarti!").

## 3.4 Contratti delle operazioni

Di seguito verranno introdotti i nuovi contratti, da aggiungere ai contratti esposti nell'iterazione 1 della fase di elaborazione:

**Contratto CO13: confermaImpostazioni**

| | |
|---|---|
| Operazione: | GiocoController.confermaImpostazioni() |
| Riferimenti: | Fase finale di configurazione partita, inizio effettivo del gioco. |
| Pre-condizioni: | • Tutti i giocatori sono stati configurati (nome, tipo, colore).<br>• L'interfaccia utente ha mostrato un riepilogo delle impostazioni e l'utente ha confermato ('S'). |
| Post-condizioni: | • Un nuovo oggetto Partita è stato creato nel GiocoController, popolato con i giocatori configurati.<br>• Il tabellone è stato inizializzato. Le posizioni iniziali dei giocatori sono state impostate a 0.<br>• L'ordine di turno dei giocatori è stato determinato casualmente. Lo stato della partita è impostato su IN_CORSO.<br>• L'interfaccia utente inizia a mostrare il tabellone e i giocatori, entrando nel ciclo di gioco principale. |

**Contratto CO14: mostraTabelloneEGiocatori**

| | |
|---|---|
| Operazione: | InterfacciaUtente.mostraTabelloneEGiocatori(Partita partita) |
| Riferimenti: | Ciclo principale del gioco. |
| Pre-condizioni: | Una partita è iniziata e il suo stato è IN_CORSO. |

| | |
|---|---|
| Post-condizioni: | • Il ciclo di gioco continua finché lo stato della partita non è TERMINATA.<br>• Per ogni iterazione del ciclo:<br>  ○ viene mostrato lo stato attuale del gioco (giocatori, posizioni, turno/giro);<br>  ○ viene gestito il turno del giocatoreCorrente (lancio dadi, movimento, effetti, ecc.);<br>  ○ (se la partita non è terminata e il giocatore non è bloccato da condizione) il turno passa al giocatore successivo.<br>• Se la partita termina (vittoria o uscita), viene mostrato il vincitore (se presente) e l'interfaccia utente torna al menu principale dopo un input dell'utente. |

**Contratto CO15: passaAlProssimoGiocatore**

| | |
|---|---|
| Operazione: | Partita.passaAlProssimoGiocatore() |
| Riferimenti: | Flusso del turno di gioco. |
| Pre-condizioni: | Un giocatore ha completato il suo turno e non è bloccato da una condizione speciale (il suo turniSaltati è >= 0 dopo la verifica degli effetti o sblocchi). L'ordine dei giocatori nella partita non è vuoto. |
| Post-condizioni: | • Il giocatoreCorrente nella Partita viene aggiornato al giocatore successivo nell'ordine.<br>• Il contatore turnoCorrente della Partita viene incrementato.<br>• Se il passaggio del turno riporta all'inizio dell'ordine dei giocatori, il giroCorrente viene incrementato e turnoCorrente resettato a 1.<br>• Se il nuovo giocatoreCorrente ha turniSaltati > 0, il suo contatore viene decrementato e il metodo continua a passare al giocatore successivo finché non ne trova uno con turniSaltati <= 0.<br>• I giocatori con turniSaltati < 0 fermano questa catena di passaggi automatici. |

**Contratto CO16: gestisciTurnoUmano**

| | |
|---|---|
| Operazione: | InterfacciaUtente.gestisciTurnoUmano(Partita partita, Giocatore giocatore) |
| Riferimenti: | Sottociclo del turno per giocatore umano. |
| Pre-condizioni: | • È il turno del giocatoreCorrente che è di tipo Umano.<br>• Il giocatore non deve saltare il turno a causa di un blocco fisso (turniSaltati > 0). |

| | |
|---|---|
| Post-condizioni: | • In base all'input dell'utente, il giocatore Umano può:<br>  ○ Lanciare i dadi (attivando il flusso CO18, CO19, CO20)<br>  ○ Tentare di uscire dalla partita (se conferma, lo stato della partita viene impostato su TERMINATA).<br>• Se l'utente non lancia i dadi o annulla l'uscita, il menu del turno viene ripresentato. |

**Contratto CO17: gestisciTurnoComputer**

| | |
|---|---|
| Operazione: | InterfacciaUtente.gestisciTurnoComputer(Partita partita, Giocatore giocatore) |
| Riferimenti: | Sottociclo del turno per giocatore Computer. |
| Pre-condizioni: | • È il turno del giocatoreCorrente che è di tipo Computer.<br>• Il giocatore non deve saltare il turno a causa di un blocco fisso (turniSaltati > 0). |
| Post-condizioni: | • Dopo l'input dell'utente per procedere, il giocatore Computer lancia automaticamente i dadi (attivando il flusso CO18, CO19, CO20).<br>• Il suo turno si completa senza ulteriori interazioni da parte dell'utente fino al passaggio al giocatore successivo (a meno che non si blocchi condizionalmente). |

**Contratto CO18: tiraDadi**

| | |
|---|---|
| Operazione: | PartitaController.tiraDadi() |
| Riferimenti: | Azione di base nel turno di gioco. |
| Pre-condizioni: | È il turno di un giocatore (Umano o Computer) che non è bloccato per un numero fisso di turni (turniSaltati > 0). |
| Post-condizioni: | • Vengono generati due valori casuali tra 1 e 6, rappresentanti il risultato del lancio.<br>• Il risultato viene mostrato all'utente (anche per la CPU).<br>• Il sistema verifica la lista di tutti i giocatori per vedere se questo lancio sblocca qualcuno (inclusa la Prigione o Attesa Dado Specifico per il giocatore corrente), aggiornando il loro turniSaltati a 0 se le condizioni sono soddisfatte.<br>• Se il giocatore corrente non è bloccato condizionalmente (turniSaltati >= 0) dopo questa verifica, il flusso prosegue con il movimento della pedina (CO19).<br>• Se il giocatore corrente è ancora bloccato condizionalmente (turniSaltati < 0), il suo turno termina qui. |

**Contratto CO19: muoviPedina**

| | |
|---|---|
| Operazione: | PartitaController.muoviPedina(Giocatore giocatore, int passi) |
| Riferimenti: | Conseguenza del lancio di dadi. |
| Pre-condizioni: | • Un giocatore (Umano o Computer) ha lanciato i dadi (somma = passi) e non è bloccato da una condizione speciale (turniSaltati >= 0) dopo il lancio.<br>• La posizione precedente del giocatore è nota. |
| Post-condizioni: | • La posizione del giocatore viene aggiornata aggiungendo passi.<br>• Viene gestito il "rimbalzo" se la nuova posizione supera la casella 63. La casella di destinazione viene identificata.<br>• La posizione aggiornata viene mostrata all'utente.<br>• Il flusso prosegue con l'applicazione dell'effetto della casella di destinazione (CO20). |

**Contratto CO20: applicaEffettoCasella**

| | |
|---|---|
| Operazione: | PartitaController.applicaEffettoCasellaEsteso(Casella casella, Giocatore giocatore, int[] valoreDadi) |
| Riferimenti: | Conseguenza del movimento sulla casella di destinazione. |
| Pre-condizioni: | • Un giocatore è atterrato sulla casella di destinazione.<br>• Lo stato della partita è IN_CORSO.<br>• I valori specifici del lancio di dadi sono disponibili (valoreDadi). |
| Post-condizioni: | • Se la casella è la numero 63, lo stato della Partita è impostato su TERMINATA. Altrimenti, se la casella è speciale, il suo effetto base viene applicato (chiamando casella.applicaEffetto(giocatore)) modificando potenzialmente la posizione del giocatore, i turni da saltare o lo stato di rilancio.<br>• Se l'effetto base della casella richiede un'ulteriore elaborazione basata sul risultato dei dadi (es. Raddoppia Movimento, Rilancia e Torna Indietro), la logica aggiuntiva viene eseguita, modificando ulteriormente lo stato del giocatore.<br>• L'effetto applicato viene mostrato all'utente. |

**Contratto CO21: mostraVincitore**

| | |
|---|---|
| Operazione: | InterfacciaUtente.mostraVincitore(Giocatore giocatore) |
| Riferimenti: | Fine partita. |
| Pre-condizioni: | Lo stato della Partita è TERMINATA. |

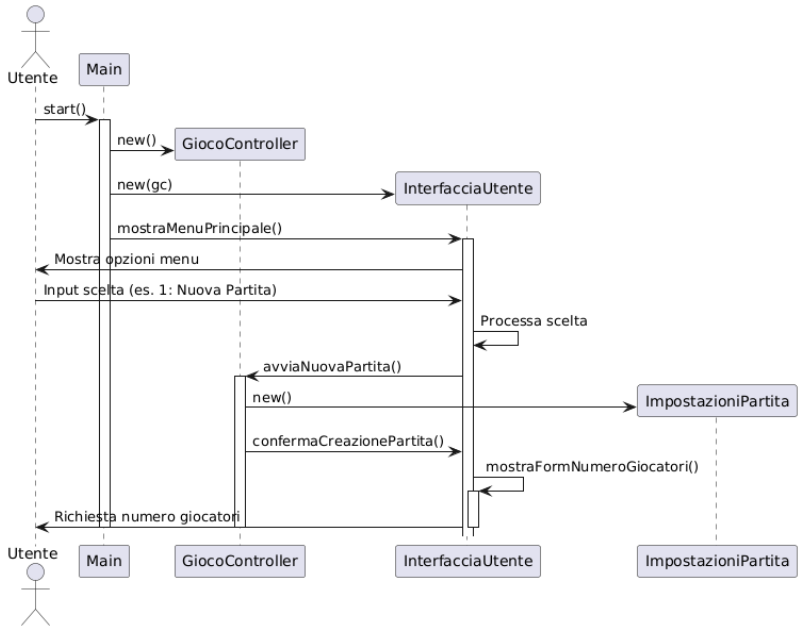
| | |
|---|---|
| Post-condizioni: | • Viene mostrato un messaggio che annuncia il giocatore come vincitore.<br>• L'interfaccia utente attende un input dell'utente (Invio) prima di tornare al menu principale. |

## 3.5 Progettazione aggiornata

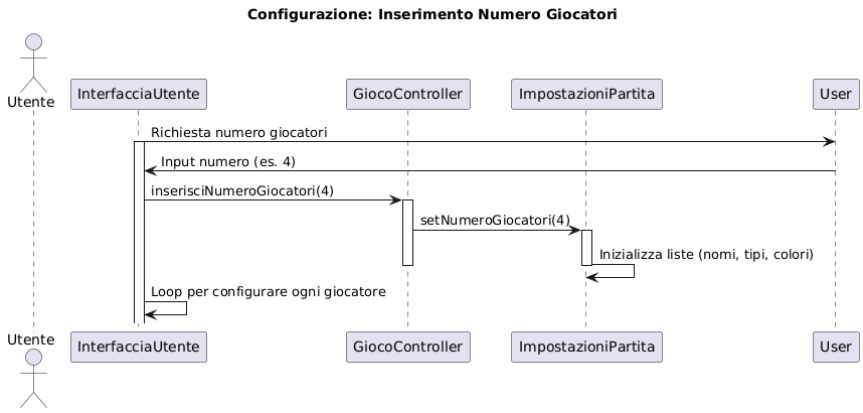
Conseguentemente al changelog, di seguito verranno mostrate tutte le versioni aggiornate e nuove dei diagrammi di sequenza e delle classi.

### 3.5.1 Diagramma di sequenza

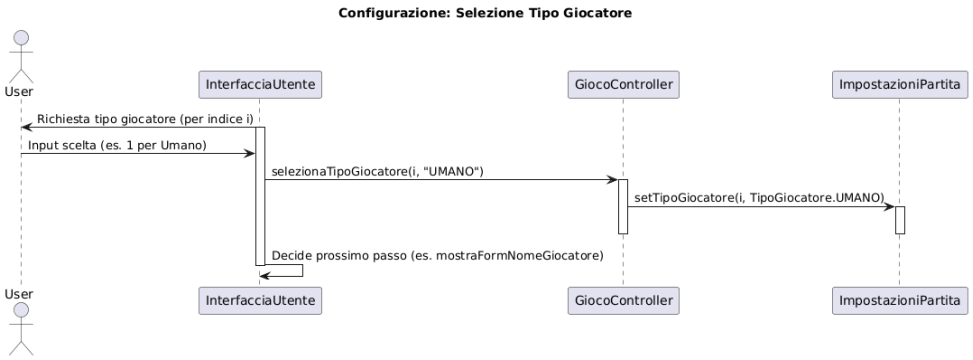
- Avvio partita e caricamento menù



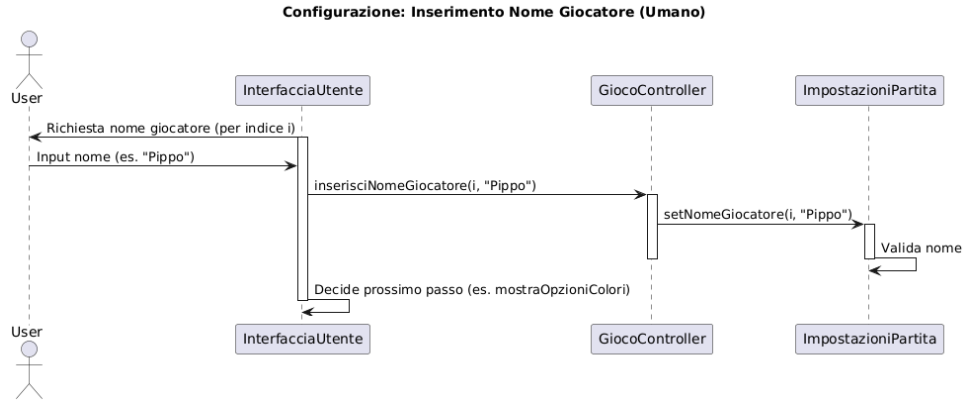

- Configurazione: inserimento giocatori



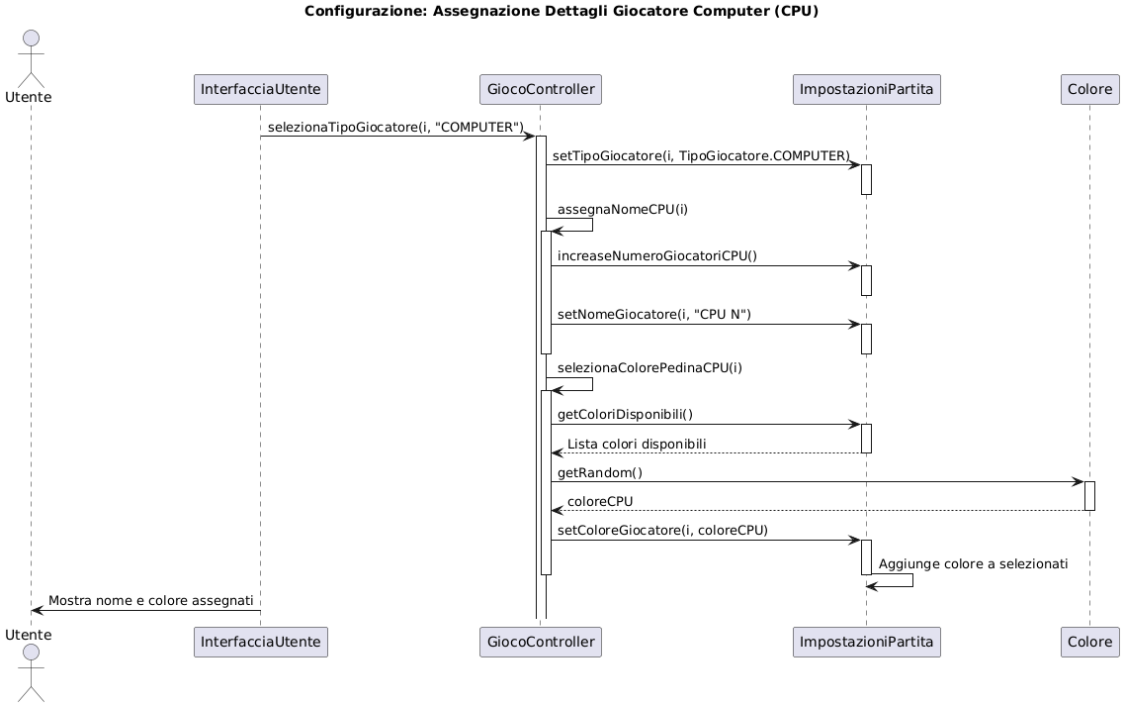
- Configurazione: selezione giocatore Umano/CPU



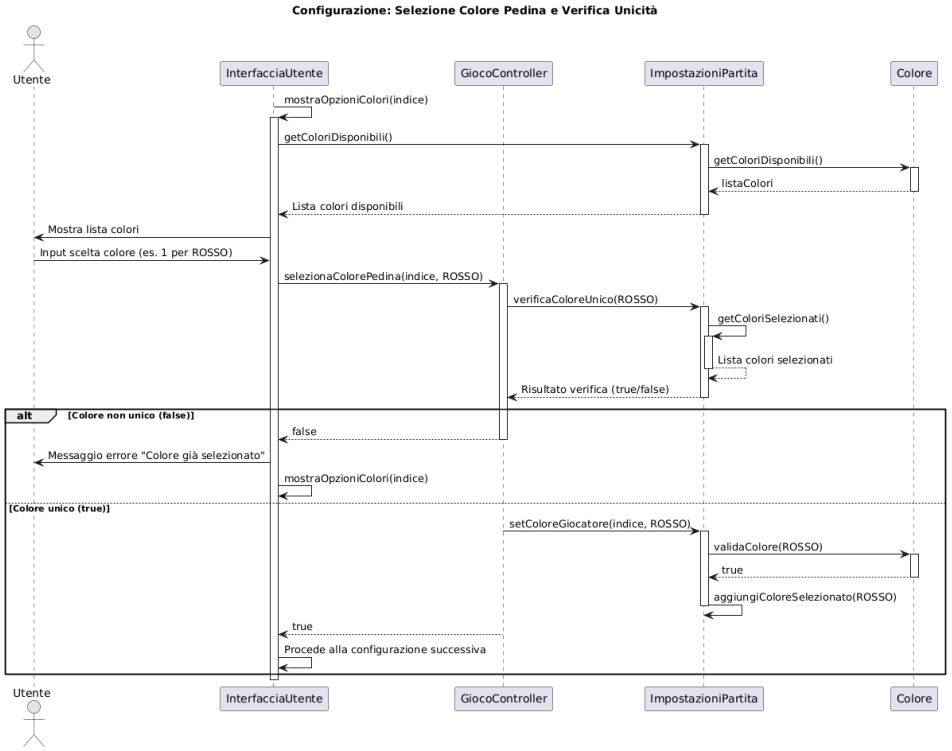

- Configurazione: inserimento nome giocatore Umano



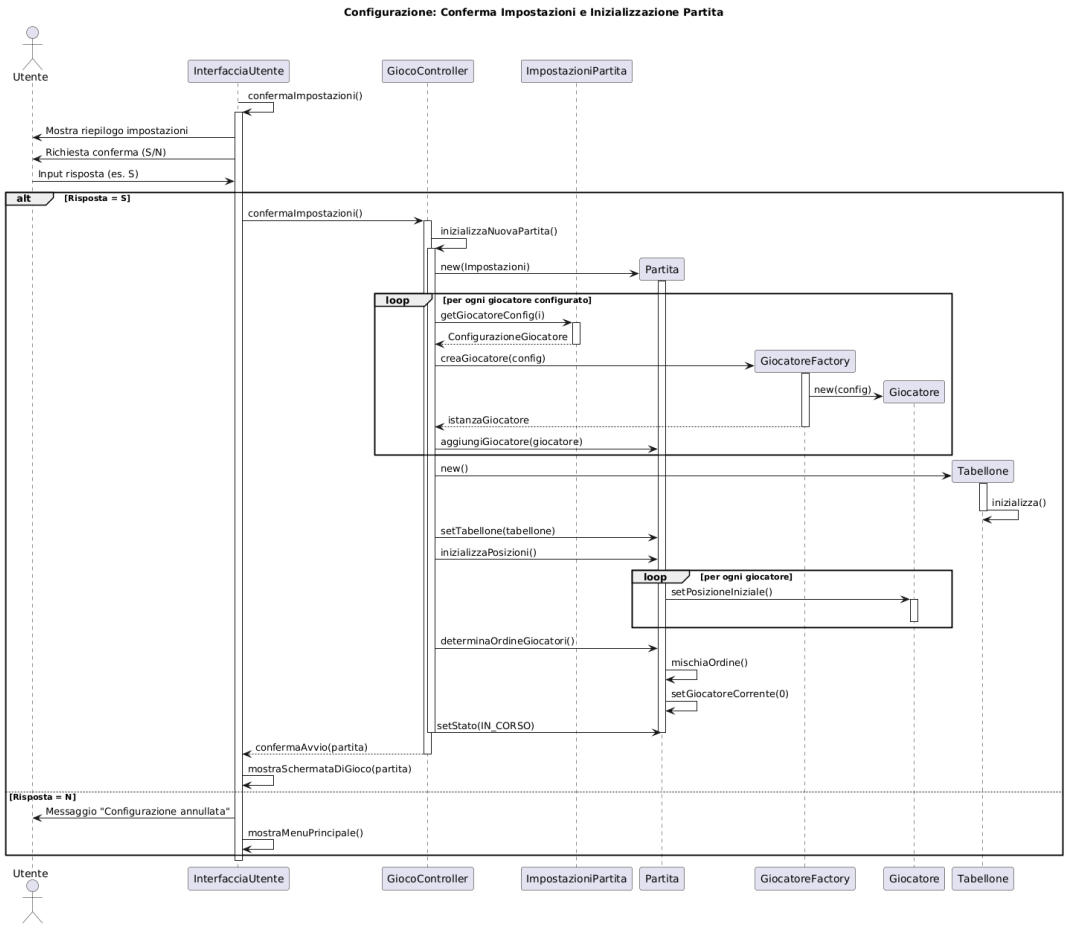

- Configurazione: assegnazione dettagli giocatore CPU



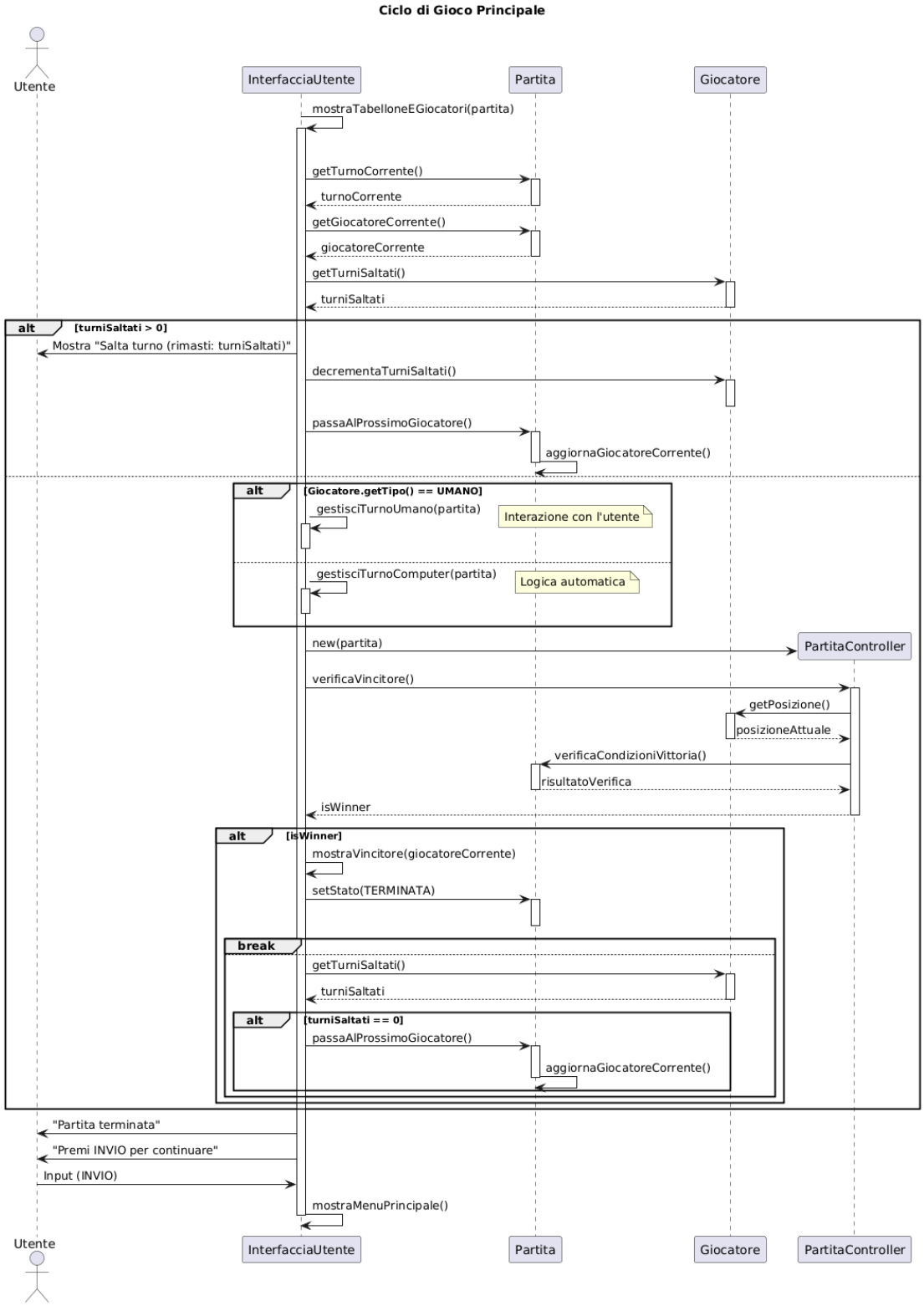
- Configurazione: selezione colore della pedina



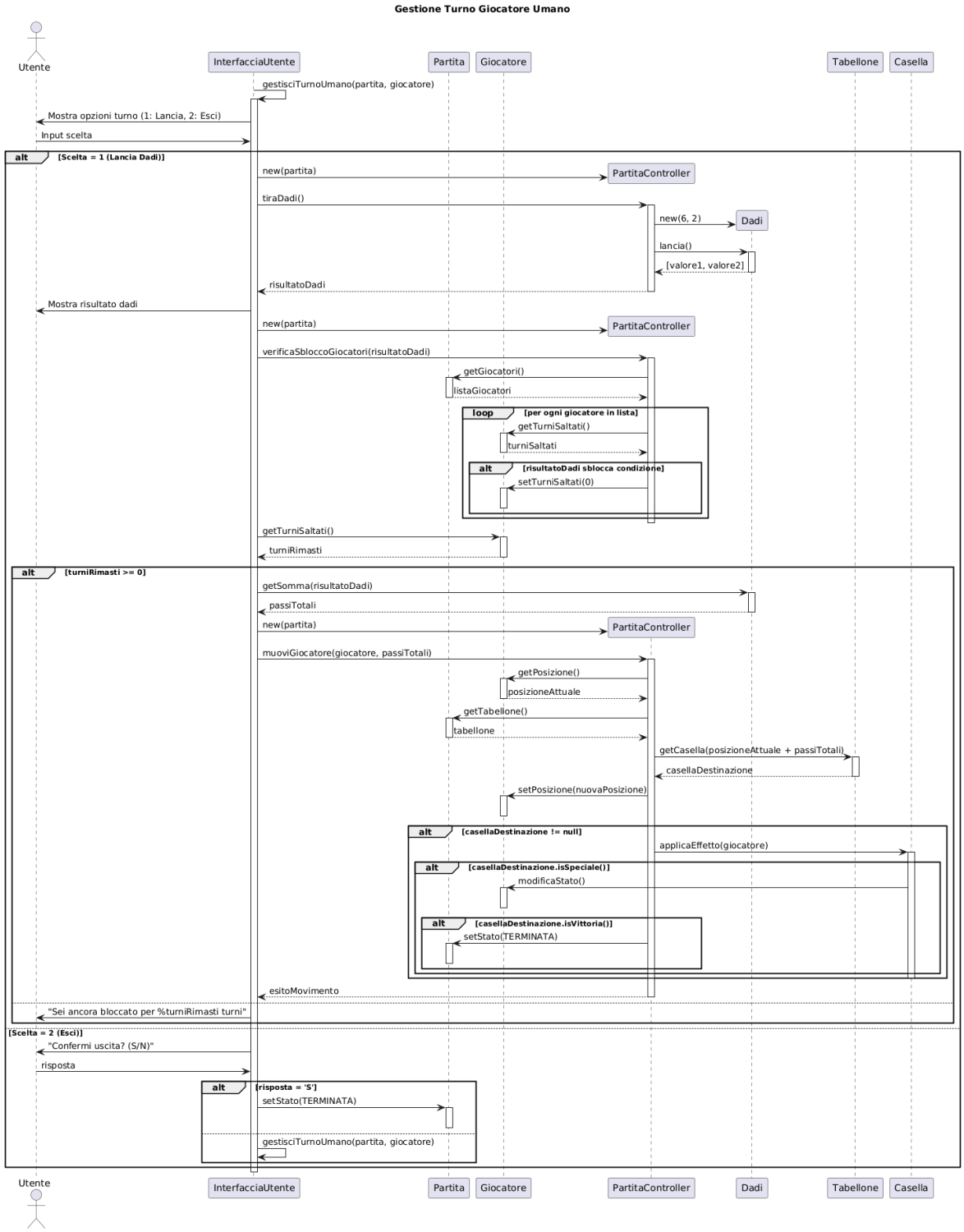

- Configurazione: inizializzazione partita



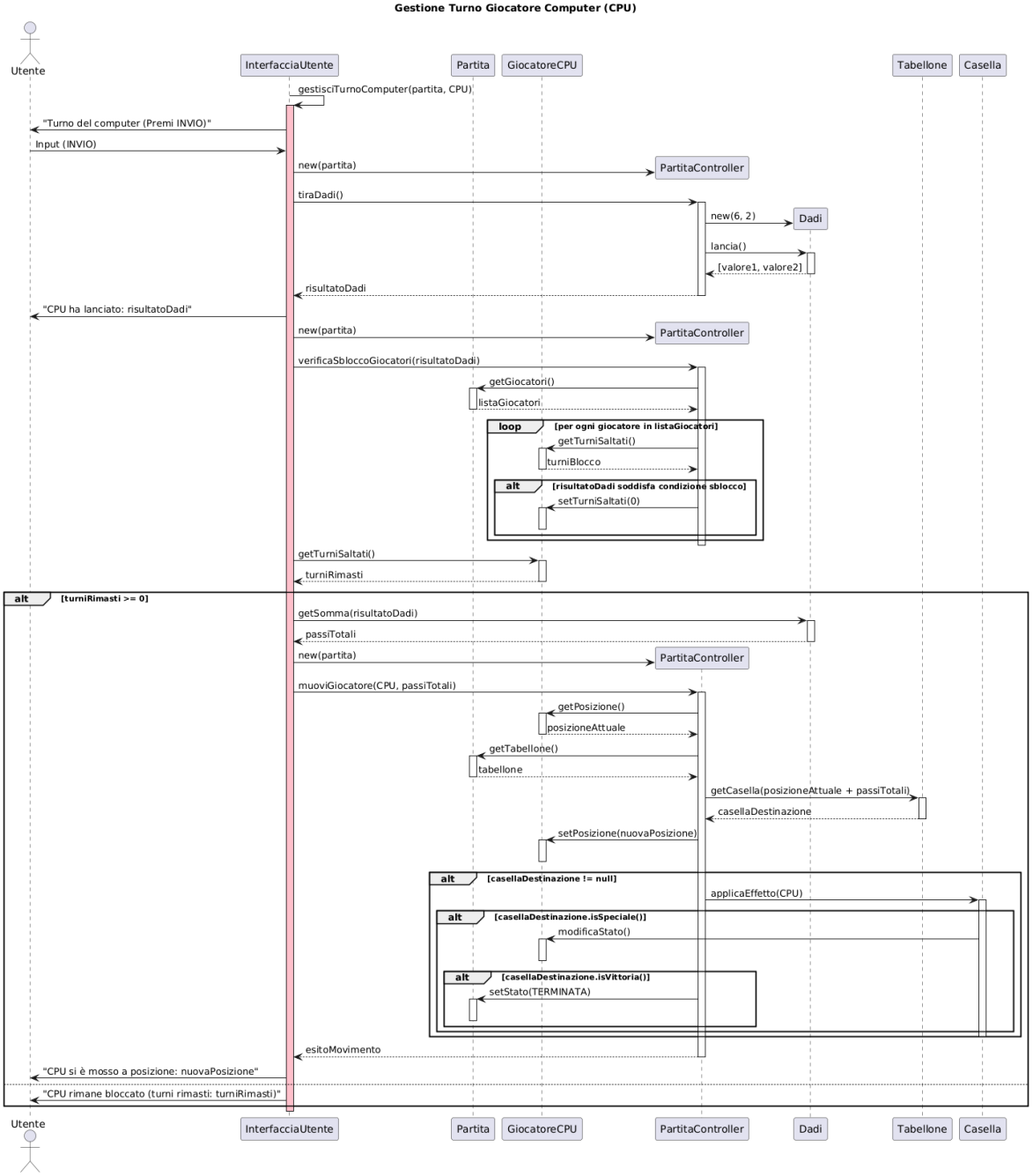
- Ciclo principale del gioco



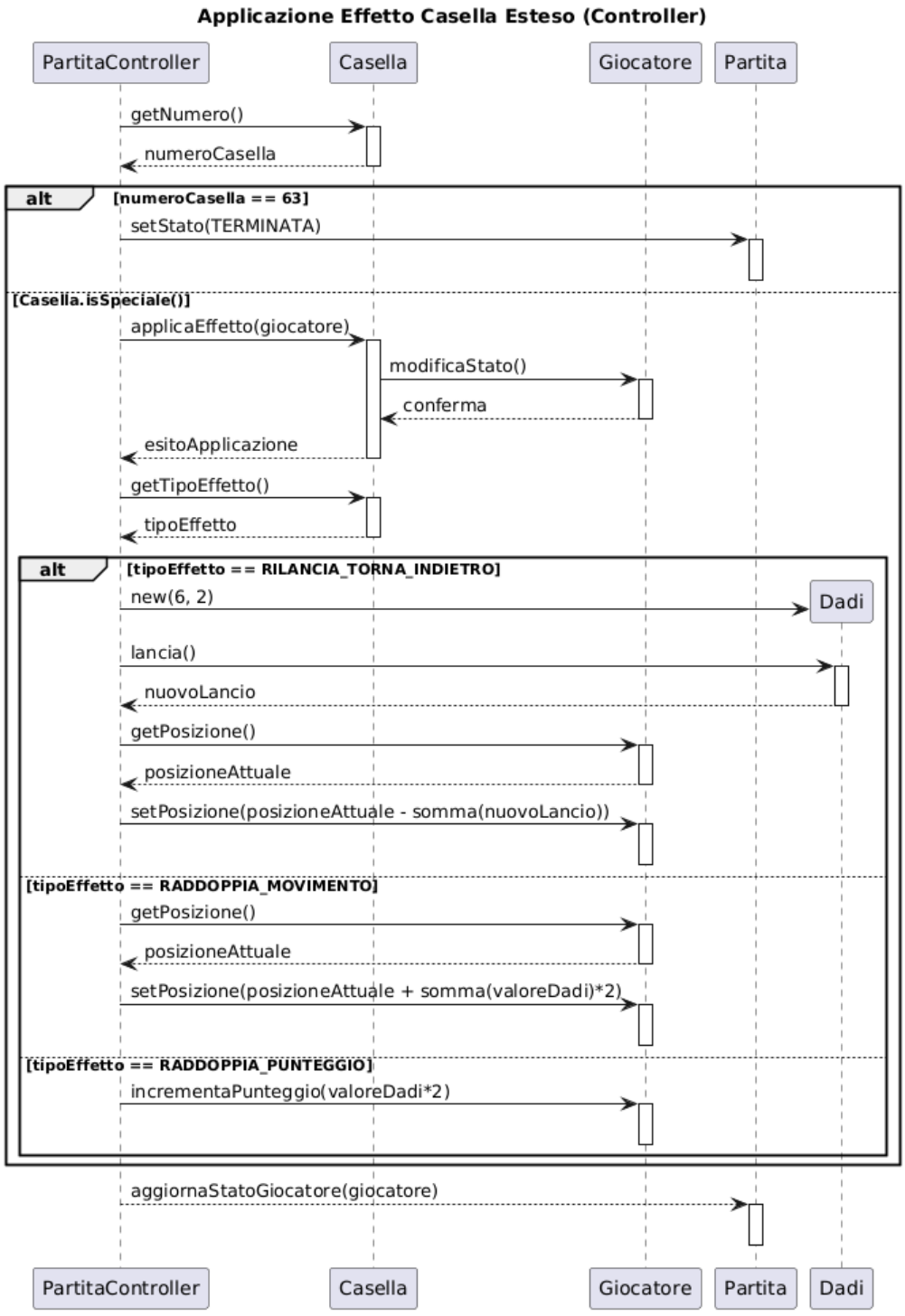
- Gestione turno del giocatore Umano



- Gestione turno del giocatore CPU



- Applicazione dell’effetto di una casella



- Gestione passaggio del turno