2 – Elaborazione – Iterazione 1

2.1 Introduzione

Conclusa la fase di ideazione, si passa alla fase di elaborazione. Lo scopo di questa fase è di implementare il nucleo dell'architettura software di DigiGoose!. In questa prima iterazione ci concentreremo sull'implementare lo scenario principale di successo del caso d'uso UC1: Avviare una nuova partita; Infine, implementeremo un caso d'uso di start up necessario per gestire l'inizializzazione per questa iterazione.

2.2 Aggiornamento caso d'uso UC1

Durante la lettura della parte sulla fase di Ideazione, sono emerse delle parti mancanti al caso d'uso UC1, al fine di poter rispettare i requisiti dell'applicazione. Verrà quindi riproposto il caso d'uso a seguire, con le modifiche aggiunte.

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 sulla casella iniziale e il sistema pronto a ricevere l'interazione del primo giocatore.
Scenario principale di successo	1. Il giocatore seleziona l'opzione "Nuova Partita". 2. Il sistema visualizza le opzioni per impostare la partita, ossia: a. Numero di giocatori (2 – 6); b. Scelta tra umano e computer per ogni giocatore c. Inserimento del nome di ciascun giocatore umano d. Scelta del colore della pedina; 3. Il giocatore seleziona il numero di giocatori desiderato e configura se è un umano o computer per ciascuno. 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. 5. Il giocatore seleziona un colore univoco per la propria pedina. Il sistema assicura che ogni giocatore abbia un colore diverso. 6. Il giocatore conferma le impostazioni della partita. 7. Il sistema inizializza la partita con le impostazioni scelte.

	8. Il sistema determina casualmente quale giocatore inizia la partita. 9. Il sistema visualizza il tabellone di gioco, le pedine posizionate sulla freccia che indica la casella numero 1, 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: a) Il giocatore riavvia l'applicazione. 2) Il giocatore seleziona opzioni di gioco non valide: a) Il sistema visualizza un messaggio di errore. b) Il giocatore modifica le opzioni e riprova. 3) Il giocatore annulla la creazione della partita: a) Il sistema torna alla schermata principale.
Requisiti speciali	1) Interfaccia utente intuitiva per la selezione delle opzioni di gioco. 2) Il sistema deve assicurare l'univocità del colore delle pedine. 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) Metodi: selezione del primo giocatore casuale, la posizione delle pedine deve essere alla freccia prima della casella 1.
Frequenza delle ripetizioni	Ogni volta che un giocatore vuole iniziare una nuova partita.
Varie	

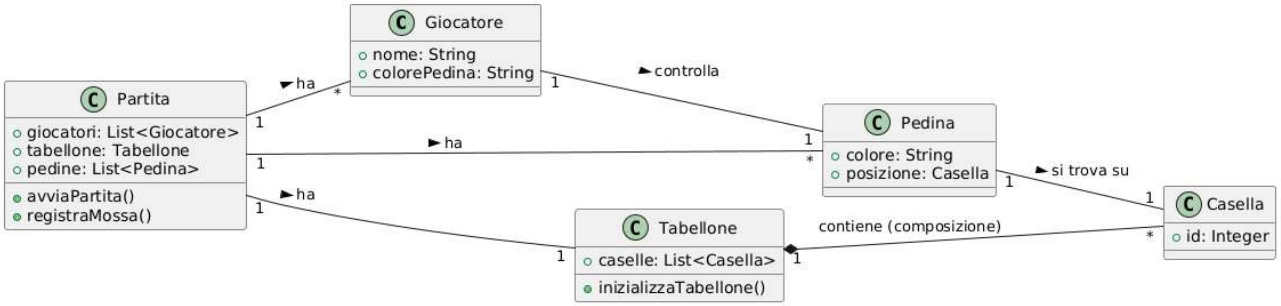
2.3 Analisi orientata agli oggetti

Verrà adesso eseguita l'analisi OO tramite i seguenti strumenti: modello di dominio, diagramma di sequenza SSD del sistema e contratti delle operazioni.

2.2.1 Modello di dominio

Dopo l'analisi del caso d'uso, sono stati scelti le seguenti classi concettuali per ottenere lo scenario principale di successo:

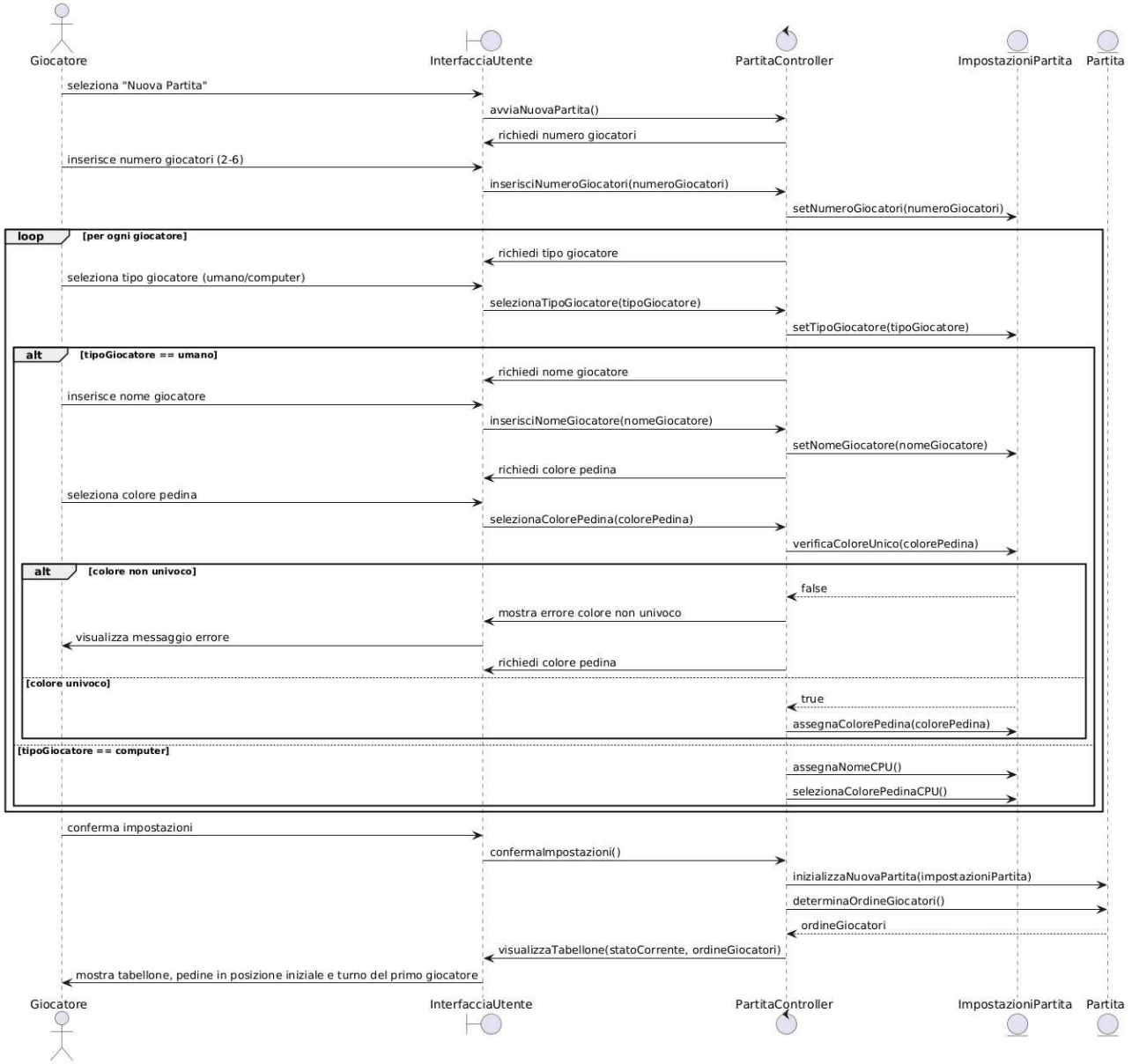
- **Giocatore**: rappresenta un partecipante al gioco, rappresentato da una pedina nel tabellone;
- **Partita**: Rappresenta la partita in corso, dove sono presenti tutti gli elementi del gioco;
- **Pedina**: Rappresenta un giocatore sul tabellone.
- **Tabellone**: Rappresenta il tabellone di gioco, composto da caselle;
- **Casella**: Rappresenta una singola casella sul tabellone;



2.2.2 Diagramma di sequenza di sistema

Il passo successivo, adesso, è la creazione del diagramma di sequenza SSD del sistema, con lo scopo di illustrare il corso degli eventi di input e output per lo scenario principale di successo. Il diagramma conterrà le seguenti interazioni, spiegate in breve:

1. L'utente inizia il processo di avviare una nuova partita all'interno dell'applicazione.
2. Il sistema risponde, chiedendo il numero di giocatori che parteciperanno alla partita.
3. L'utente inserisce il numero di giocatori desiderato e lo comunica al sistema.
4. Per ogni giocatore:
    a. Il sistema chiede il tipo di giocatore (umano o computer).
    b. L'utente seleziona il tipo desiderato per quel giocatore.
    c. Se il giocatore è un umano:
        i. Il sistema richiede il nome del giocatore.
        ii. Il giocatore inserisce il proprio nome.
        iii. Il sistema richiede il colore della pedina desiderato dal giocatore.
        iv. Il giocatore seleziona il colore della pedina.
    d. Se il giocatore è controllato dal computer:
        i. Il sistema assegna automaticamente un nome predefinito (CPU 1, CPU 2,...) assicurandosi sia univoco.
        ii. Il sistema seleziona automaticamente un colore per la pedina casuale tra quelli ancora disponibili.
5. Il sistema esegue internamente la funzione di inizializzazione con le impostazioni scelte.
6. Il sistema determina internamente l'ordine dei giocatori in maniera casuale.
7. Il sistema comunica l'ordine di gioco, visualizza il tabellone di gioco con le pedine nella posizione iniziale e indica il primo giocatore.


Giocatore
InterfacciaUtente
PartitaController
ImpostazioniPartita
Partita
seleziona "Nuova Partita"
avviaNuovaPartita()
richiedi numero giocatori
inserisce numero giocatori (2-6)
inserisciNumeroGiocatori(numeroGiocatori)
setNumeroGiocatori(numeroGiocatori)
loop
[per ogni giocatore]
richiedi tipo giocatore
seleziona tipo giocatore (umano/computer)
selezionaTipoGiocatore(tipoGiocatore)
setTipoGiocatore(tipoGiocatore)
alt
[tipoGiocatore == umano]
richiedi nome giocatore
inserisce nome giocatore
inserisciNomeGiocatore(nomeGiocatore)
setNomeGiocatore(nomeGiocatore)
richiedi colore pedina
seleziona colore pedina
selezionaColorePedina(colorePedina)
verificaColoreUnico(colorePedina)
alt
[colore non univoco]
false
mostra errore colore non univoco
visualizza messaggio errore
richiedi colore pedina
[colore univoco]
true
assegnaColorePedina(colorePedina)
[tipoGiocatore == computer]
assegnaNomeCPU()
selezionaColorePedinaCPU()
conferma impostazioni
confermaImpostazioni()
inizializzaNuovaPartita(impostazioniPartita)
determinaOrdineGiocatori()
ordineGiocatori
visualizzaTabellone(statoCorrente, ordineGiocatori)
mostra tabellone, pedine in posizione iniziale e turno del primo giocatore
Giocatore
InterfacciaUtente
PartitaController
ImpostazioniPartita
Partita

2.2.3 Contratti delle operazioni

Adesso vengono definiti i contratti, per definire le principali operazioni di sistema al fine di gestire quanto visto nell'SSD:

Contratto CO1: avviaNuovaPartita

Operazione:	avviaNuovaPartita()
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	L'applicazione DigiGoose! è in esecuzione e l'Attore Primario (Giocatore) ha selezionato l'opzione "Avvia nuova partita".
Post-condizioni:	Il sistema è nello stato di attesa dell'inserimento del numero di giocatori.

Contratto CO2: inserisciNumeroGiocatori

Operazione:	inserisciNumeroGiocatori(numeroGiocatori: integer)
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	Il sistema ha richiesto il numero di giocatori.
Post-condizioni:	• Il sistema è nello stato di attesa della selezione del tipo di giocatore per il primo giocatore (umano o computer). • L'attributo 'numeroGiocatori' della sessione di nuova partita è impostato al valore fornito. • Si avvia un ciclo per raccogliere le informazioni di ogni giocatore.

Contratto CO3: selezionaTipoGiocatore

Operazione:	selezionaTipoGiocatore(tipoGiocatore: enum {umano, computer})
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	Il sistema ha richiesto il tipo di giocatore per il giocatore corrente
Post-condizioni:	• Se tipoGiocatore è 'umano', il sistema è nello stato di attesa dell'inserimento del nome del giocatore. • Se tipoGiocatore è 'computer', il sistema procede automaticamente all'assegnazione del nome CPU e alla selezione del colore della pedina. • L'attributo 'tipo' del giocatore corrente nella sessione di nuova partita è impostato al valore fornito.

Contratto CO4: inserisciNomeGiocatore

Operazione:	inserisciNomeGiocatore(nomeGiocatore: string)
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	Il sistema ha richiesto il nome del giocatore (giocatore umano)
Post-condizioni:	• Il sistema è nello stato di attesa della selezione del colore della pedina per il giocatore corrente. • L'attributo 'nome' del giocatore umano corrente nella sessione di nuova partita è impostato al valore fornito.

Contratto CO5: selezionaColorePedina

Operazione:	selezionaColorePedina(colorePedina: enum {elenco dei colori disponibili})
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	l sistema ha richiesto il colore della pedina per il giocatore corrente (giocatore umano).
Post-condizioni:	• L'attributo 'colorePedina' del giocatore umano corrente viene verificato per unicità tramite verificaColoreUnico. • Se il colore è univoco, l'attributo 'colorePedina' del giocatore viene impostato al valore fornito. • Se il colore non è univoco, il sistema mostra un messaggio di errore e richiede nuovamente la selezione del colore. • Se ci sono altri giocatori da configurare, il sistema richiede il tipo di giocatore successivo, altrimenti procede con la conferma delle impostazioni.

Contratto CO6: verificaColoreUnico

Operazione:	verificaColoreUnico(colorePedina: enum {elenco dei colori disponibili})
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	È stato selezionato un colore per il giocatore corrente.
Post-condizioni:	• Il sistema ha verificato se il colore selezionato è già stato assegnato ad un altro giocatore. • Ritorna true se il colore è disponibile (univoco), false altrimenti.

Contratto CO7: assegnaNomeCPU

Operazione:	assegnaNomeCPU()
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	Il tipo di giocatore corrente è 'computer'.
Post-condizioni:	• Il sistema ha generato un nome univoco per il giocatore computer (ad esempio, "CPU 1", "CPU 2", ...). • L'attributo 'nome' del giocatore computer corrente nella sessione di nuova partita è impostato sul nome generato. • Il sistema procede alla selezione automatica del colore della pedina per il giocatore computer.

Contratto CO8: selezionaColorePedinaCPU

Operazione:	selezionaColorePedinaCPU()
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	Il tipo di giocatore corrente è 'computer' e il nome è stato assegnato.
Post-condizioni:	• Il sistema ha selezionato un colore della pedina casuale tra quelli disponibili (non scelti dai giocatori umani e da altri giocatori computer). • L'attributo 'colorePedina' del giocatore computer corrente nella sessione di nuova partita è impostato sul colore selezionato. • Se ci sono altri giocatori da configurare, il sistema richiede il tipo di giocatore successivo. Altrimenti, procede all'inizializzazione della partita.

Contratto CO9: confermaImpostazioni

Operazione:	confermaImpostazioni()
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	Sono state raccolte tutte le informazioni per tutti i giocatori (tipo, nome, colore pedina).
Post-condizioni:	Il sistema procede all'inizializzazione della nuova partita con le impostazioni specificate.

Contratto CO10: inizializzaNuovaPartita

Operazione:	inizializzaNuovaPartita(impostazioniPartita: collezione di informazioni sui giocatori)
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	Le impostazioni della partita sono state confermate.
Post-condizioni:	• È stata creata una nuova istanza di 'Partita'. • Per ogni giocatore specificato, è stata creata una nuova istanza di 'Giocatore' (umano o computer) con i rispettivi attributi (nome, colorePedina) e associata alla 'Partita'. • È stata gestita l'associazione univoca dei colori delle pedine ai giocatori. • La 'Partita' è nello stato iniziale.

Contratto CO11: determinaOrdineGiocatori

Operazione:	determinaOrdineGiocatori()
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	La nuova partita è stata inizializzata con i giocatori.
Post-condizioni:	• È stato selezionato casualmente l'ordine dei giocatori della partita. • L'attributo 'giocatoreCorrente' della 'Partita' è impostato sul primo giocatore seguendo l'ordine. • L'ordine dei giocatori viene restituito al controller.

Contratto CO12: visuaizzaTabellone

Operazione:	visualizzaTabellone(statoCorrente: stato del gioco, ordineGiocatori: [Giocatori])
Riferimenti:	caso d'uso: Avviare una nuova partita
Pre-condizioni:	È stato determinato l'ordine dei giocatori.
Post-condizioni:	• L'interfaccia utente visualizza il tabellone di gioco con le pedine nella posizione iniziale (sulla freccia che indica la casella numero 1). • L'interfaccia utente visualizza i nomi di tutti i giocatori partecipanti. • L'interfaccia utente indica chiaramente il nome del giocatore a cui tocca il turno, seguendo l'ordine determinato. • Il sistema aggiorna la sessione corrente, sovrascrivendo eventuali partite salvate precedentemente.

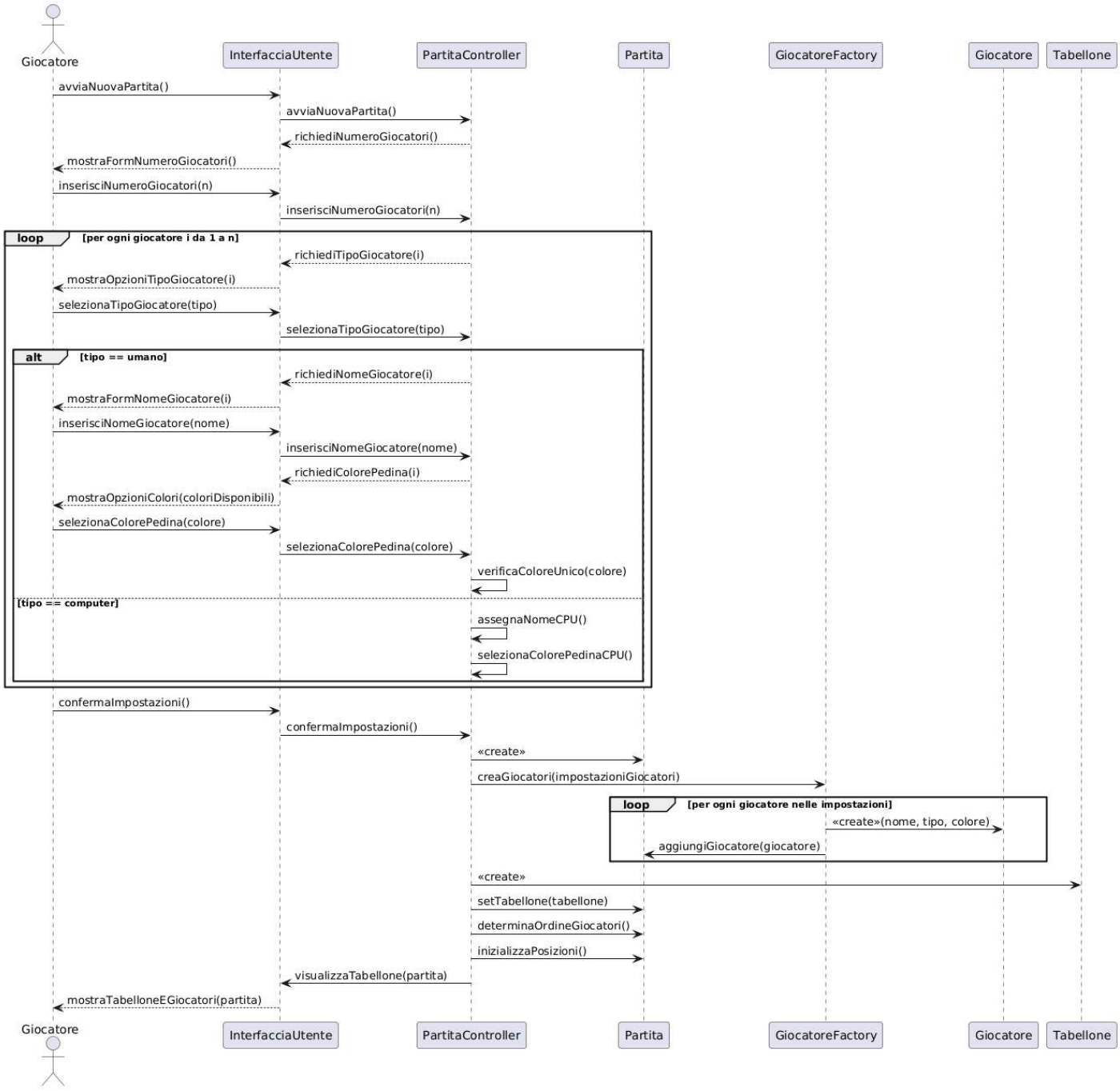
2.4 Progettazione

Adesso concentriamoci su gli oggetti software necessari per la progettazione. L'obiettivo di questa fase è il modello di progetto, che illustreremo attraverso i diagrammi di sequenza e delle classi.

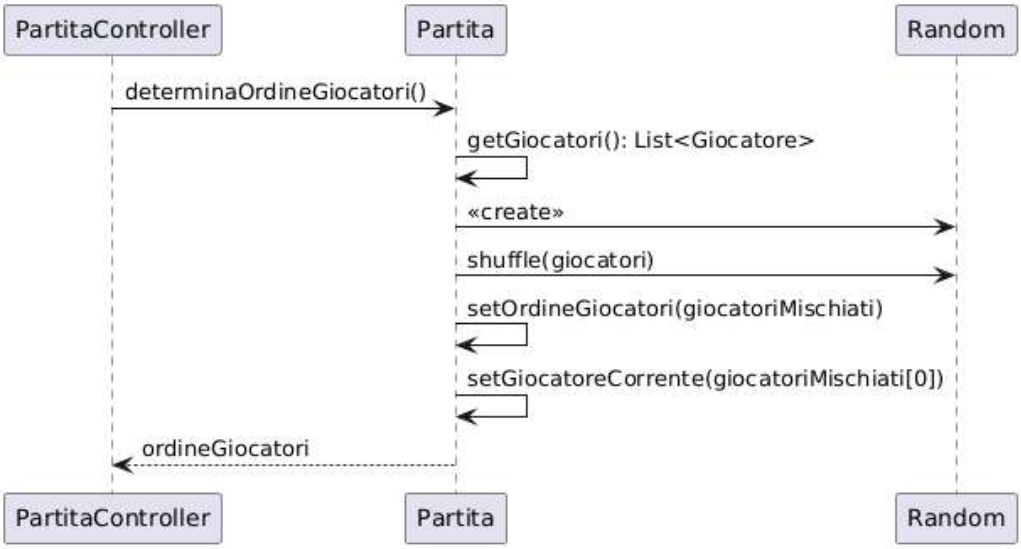
2.4.1 Diagrammi di sequenza

Il diagramma di sequenza ha lo scopo di illustrare nel dettaglio il flusso degli eventi che si verificano tra il giocatore e il sistema.

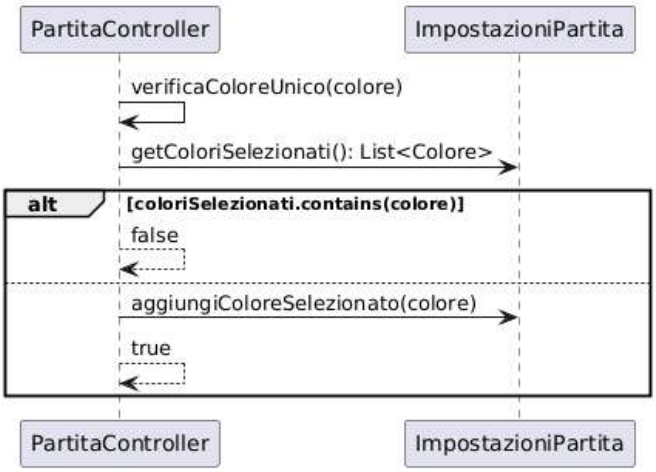
- Avviare una nuova partita



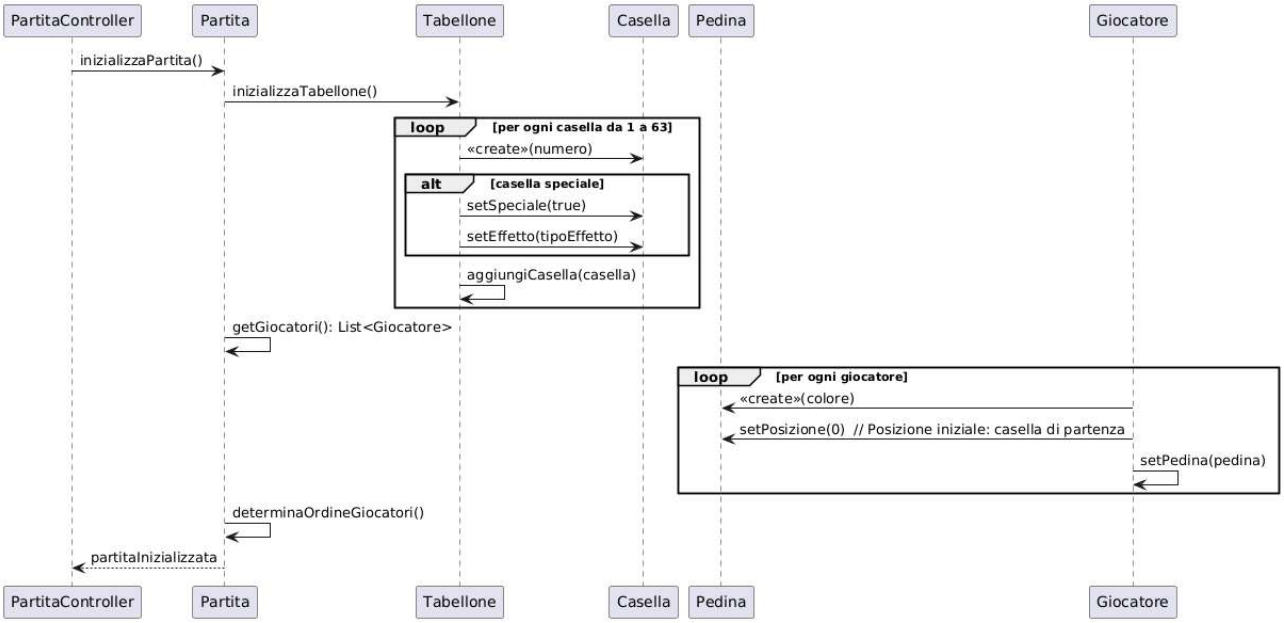
- Determinare l’ordine dei giocatori



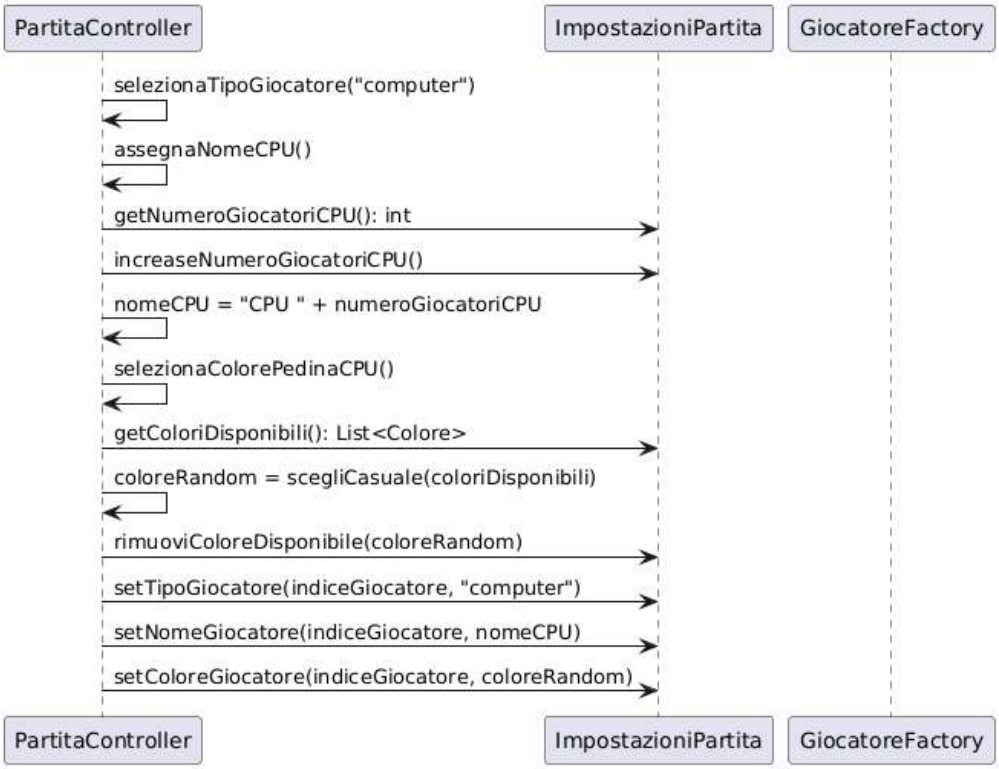
- Verifica dell'unicità del colore della pedina



- Inizializzazione della partita



- Gestione dei giocatori controllati dal computer



2.4.2 Diagramma delle classi

Infine, facciamo il diagramma delle classi, strutturandolo secondo l'architettura MVC. Di seguito viene riportato il diagramma per completezza, ma per una maggior fruibilità nella documentazione viene fornito l'UML e il PNG del diagramma delle classi (e di tutti gli altri diagrammi).