Elaborazione-Iterazione 4

1.1 Introduzione

Durante la quarta iterazione i requisiti principali su cui mi sono concentrato sono:

- Implementazione dello scenario principale di successo e delle relative estensioni del caso d'uso *UC3: Pagamento prenotazione campo.*
- Implementazione degli scenari secondari, e quindi delle estensioni, per quanto riguarda i casi d'uso *UC1: Inserimento nuova torneo*, *UC2: Gestione prenotazione* e *UC6: Registrazione cliente*.
- Implementazione delle regole di dominio;
- Aggiornamento classi Prenotazione e Partita che vengono racchiusi in un'unica entità e con un'aggiunta in SportPlanner.
- Applicazione del pattern *Strategy* per la politica di gestione del prezzo di una Prenotazione.

1.2 Caso d'uso UC3: Pagamento prenotazione campo

Di seguito viene riportato nel dettaglio il caso d'uso *UC3: Effettua pagamento*, in cui è stata aggiunta l'estensione 3a e 3b.

Nome del caso d'uso	UC3: Pagamento prenotazione campo
Portata	Applicazione SportPlanner
Livello	Obiettivo utente
Attore primario	Addetto
Parti interessate e interessi	**Addetto** → vuole assicurarsi del corretto pagamento della prenotazione. **Cliente** → vuole pagare la quota totale corretta
Pre-condizioni	Il Cliente deve aver prenotato correttamente la partita tramite l'interfaccia.

Garanzia di successo	L'Addetto riscuote la corretta somma rispetto alla prenotazione del cliente. Il pagamento verrà registrato nel sistema.
Scenario principale di successo	1. Il Cliente si reca dall'Addetto per pagare la quota corretta; 2. Il Cliente mostra all'Addetto il codice di prenotazione; 3. L'Addetto sceglie l'attività "Paga prenotazione" passando il codice; 4. Il Sistema ritorna la quota da pagare; 5. Il Cliente paga la somma; L'Addetto registra il pagamento nel sistema;
Estensioni	***a**. In un qualsiasi momento il Sistema fallisce e ha un arresto improvviso: 1. L'Addetto riavvia il software e ripristina lo stato precedente del Sistema; 2. Il Sistema ripristina lo stato; **3a**. L'Addetto passa al sistema il codice di una prenotazione non esistente: 1. Il Sistema genera un messaggio di errore; 2. L'Addetto ripete il passaggio 3 inserendo nuovamente il codice; 3. Se viene nuovamente generato un errore, l'Addetto richiede il codice al Cliente, ripetendo il passaggio 2; **3b**. L'Addetto passa al sistema il codice di una prenotazione già pagata: 1. Il Sistema genera un messaggio di errore; 2. L'Addetto ripete il passaggio 3 inserendo nuovamente il codice; 3. Se viene nuovamente generato un errore, l'Addetto richiede il codice al Cliente, ripetendo il passaggio 2;
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	

Frequenza di ripetizioni	Legata all'affluenza di Clienti.
Varie	

1.3 Analisi Orientata agli Oggetti

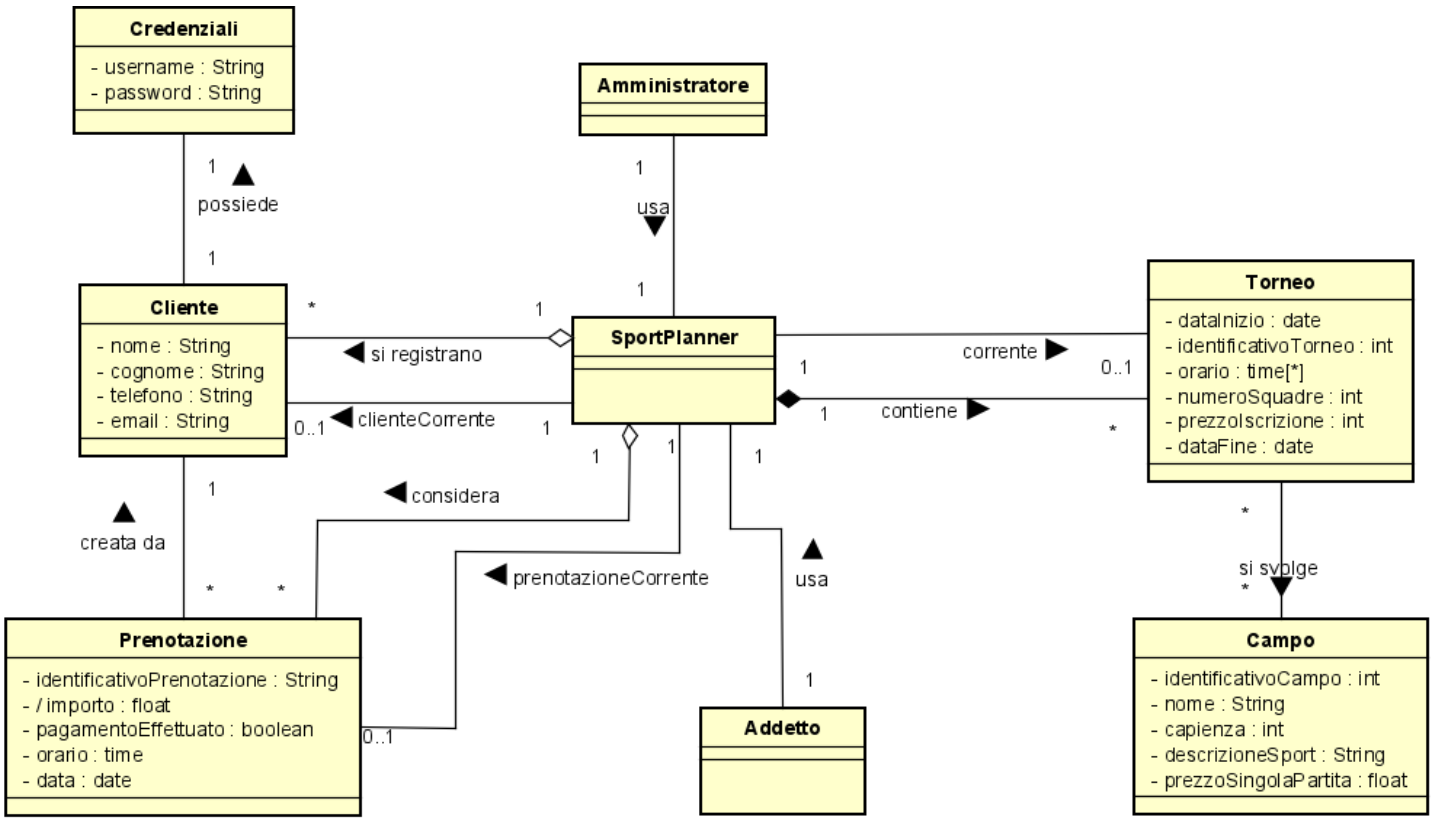
Per l'analisi della quarta iterazione verranno quindi utilizzati gli stessi strumenti: Modello di Dominio, SSD (Sequence System Diagram) e Contratti delle operazioni.

1.3.1 Modello di Dominio

La prima cosa da fare è la definizione del Modello di Dominio, un elaborato grafico in cui vengono identificati concetti, attributi e associazioni più importanti. Dall'analisi del caso d'uso UC3, valutando lo scenario principale di success, è stata possibile identificare la seguente nuova classe concettuale:

- **Addetto:** Persona fisica che si occupa della attività di gestione dei pagamenti, consentendoli ai Clienti che hanno effettuato una prenotazione di un campo o si sono iscritti ad un Torneo.

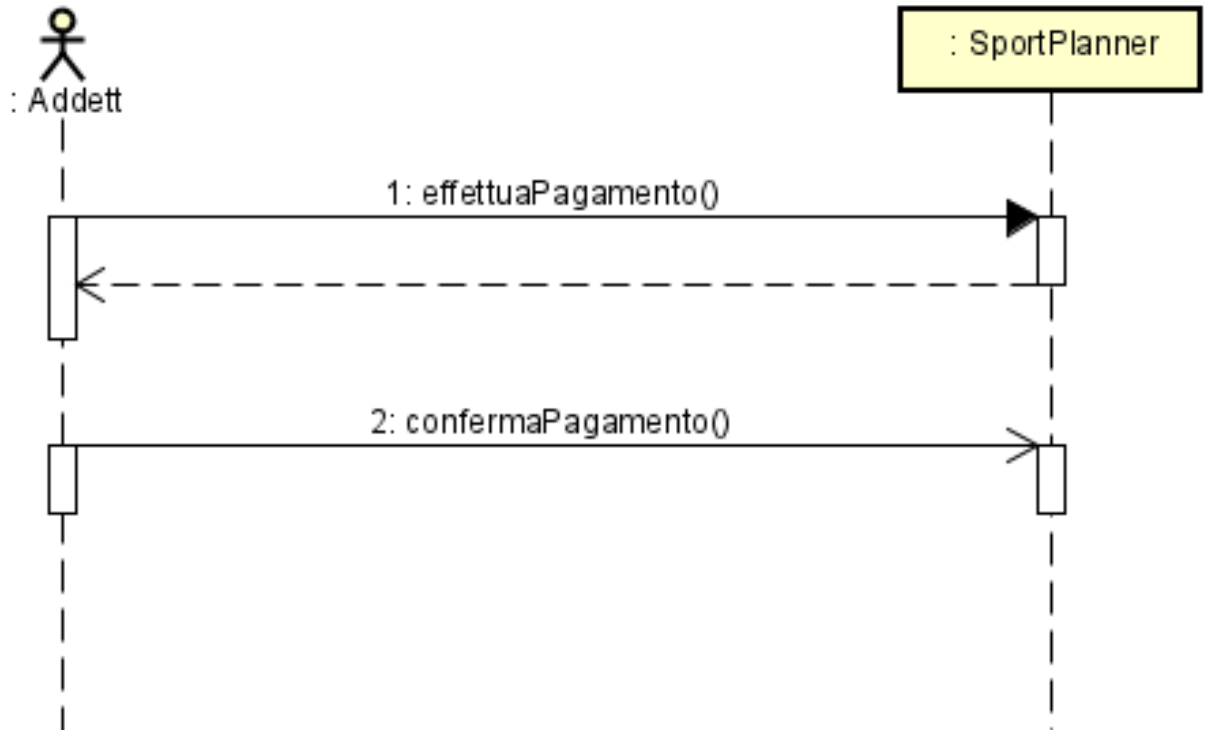
Inoltre in questa iterazione è stato ritenuto opportuno riunire in un'unica classe concettuale l'entità della Partita e della Prenotazione, in quanto la prenotazione non è un evento che viene aggiunto dall'Amministratore, come il Torneo, ma viene creata solo in corso di una Prenotazione.
In questo modo all'interno del software si potrà tenere una mappa contenente l'elenco delle Prenotazioni che permetta anche di tenere conto di data e orario. Tenendo conto di associazioni e attributi, si costruisce il seguente Modello di Dominio.



1.3.2 Diagramma di Sequenza di Sistema

Procedendo con l'Analisi Orientata agli Oggetti, il secondo passo è la creazione del Diagramma di Sequenza di Sistema (SSD) per poter illustrare il corso degli eventi di I/O per lo scenario principale di successo del caso d'uso scelto (*UC3*).

Il diagramma sarà il seguente:



1.3.1 Contratti delle operazioni

Utilizzando i Contratti delle operazioni, verrà fatta una descrizione delle principali operazioni di sistema che si occupano della gestione degli eventi di sistema trovati e analizzati.

CONTRATTO CO1	**effettuaPagamento**
Operazione	effettuaPagamento (identificativoPrenotazione)
Riferimenti	Caso d'uso UC3: Pagamento prenotazione partita
Pre-condizioni	–
Post-condizioni	–

CONTRATTO CO2	confermaPagamento
Operazione	confermaPagamento ()
Riferimenti	Caso d'uso UC3: Pagamento prenotazione partia
Pre-condizioni	– *identificativoPrenotazione* è stato accettato dal Sistema
Post-condizioni	– gli attributi di Prenotazione sono stati aggiornati

1.4 Progettazione Orientata agli Oggetti

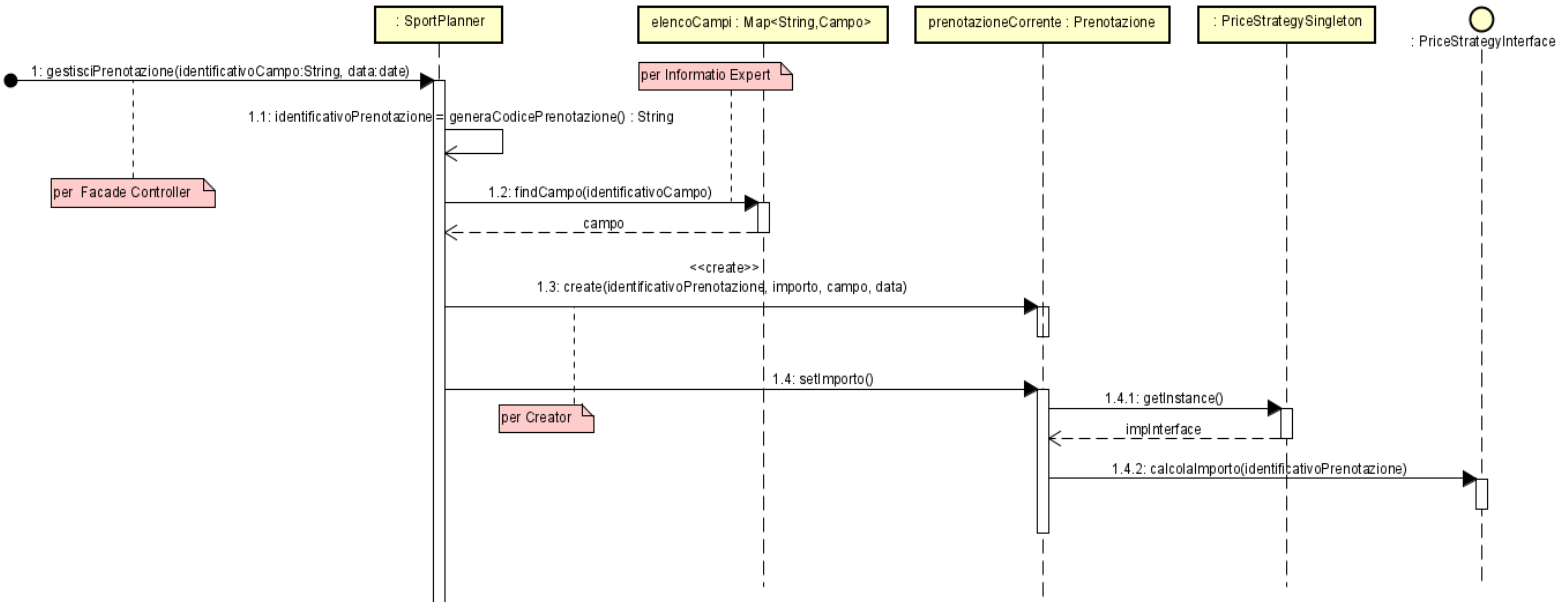
Di seguito troviamo i diagrammi più significativi relativi al caso d'uso UC3.

Inoltre in questa iterazione è stato aggiornato il diagramma di sequenza relativo al caso d'uso UC2, in precisione nell'operazione "gestionePrenotazione" è stata presa in considerazione la presenza del pattern *Strategy* per il calcolo del prezzo più opportuno, permettendo anche di implementare le varie Regole di Dominio oltre che altre possibili strategie di sconto.

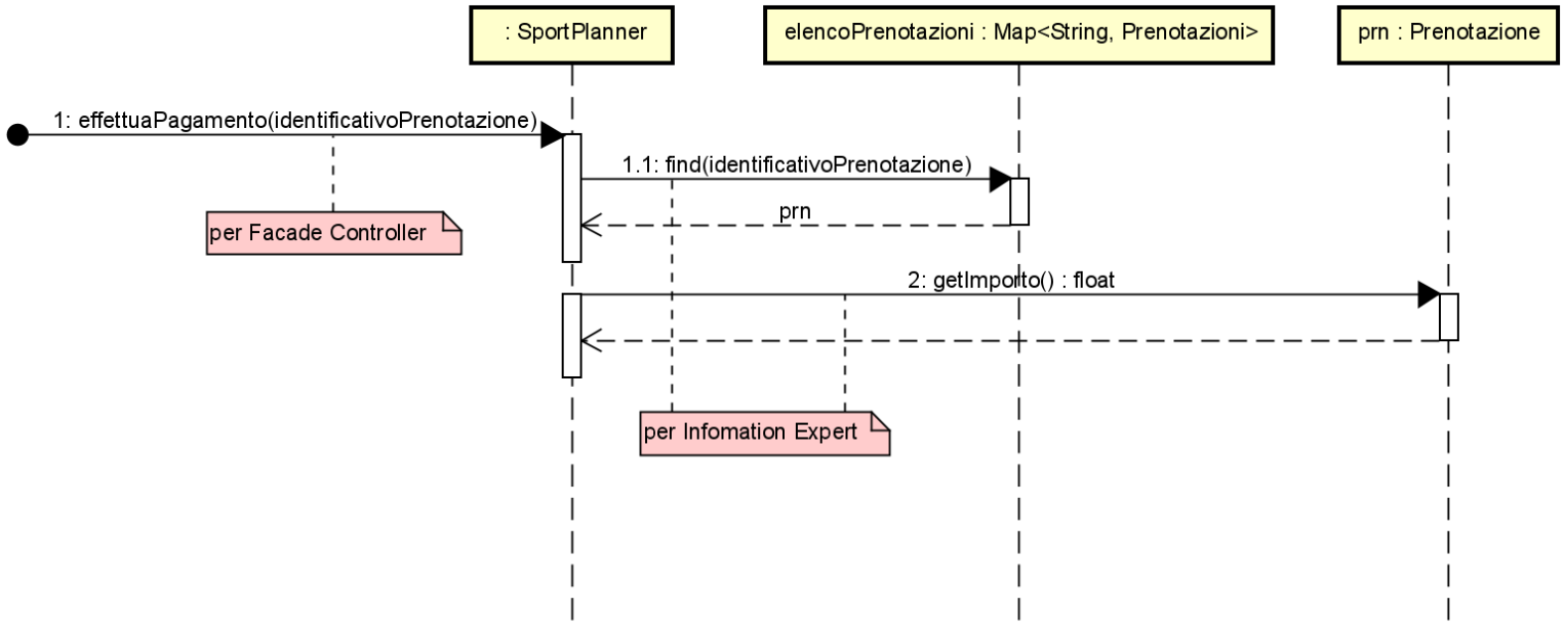
1.4.1 Diagramma di Sequenza

La classe Prenotazione richiede alla classe *PriceStrategySingleton* un oggetto che implementa l'interfaccia Price*StrategyInterface* utilizzando il design pattern *Singleton*. La classe *PriceStrategySingleton* sceglierà quale tipo di oggetto da ritornare, anche se in questo caso verrà ritornata la strategia Standard o la strategia Promo, implementate dalla classe *PriceStandardStrategy, PricePromoStrategy, PriceGiftStrategy e PriceIncreaseStrategy* ma sarà possibile aggiungere altre strategie semplicemente implementando l'interfaccia *PriceStrategyInterface* e ridefinendo il metodo *calcolaImporto*.

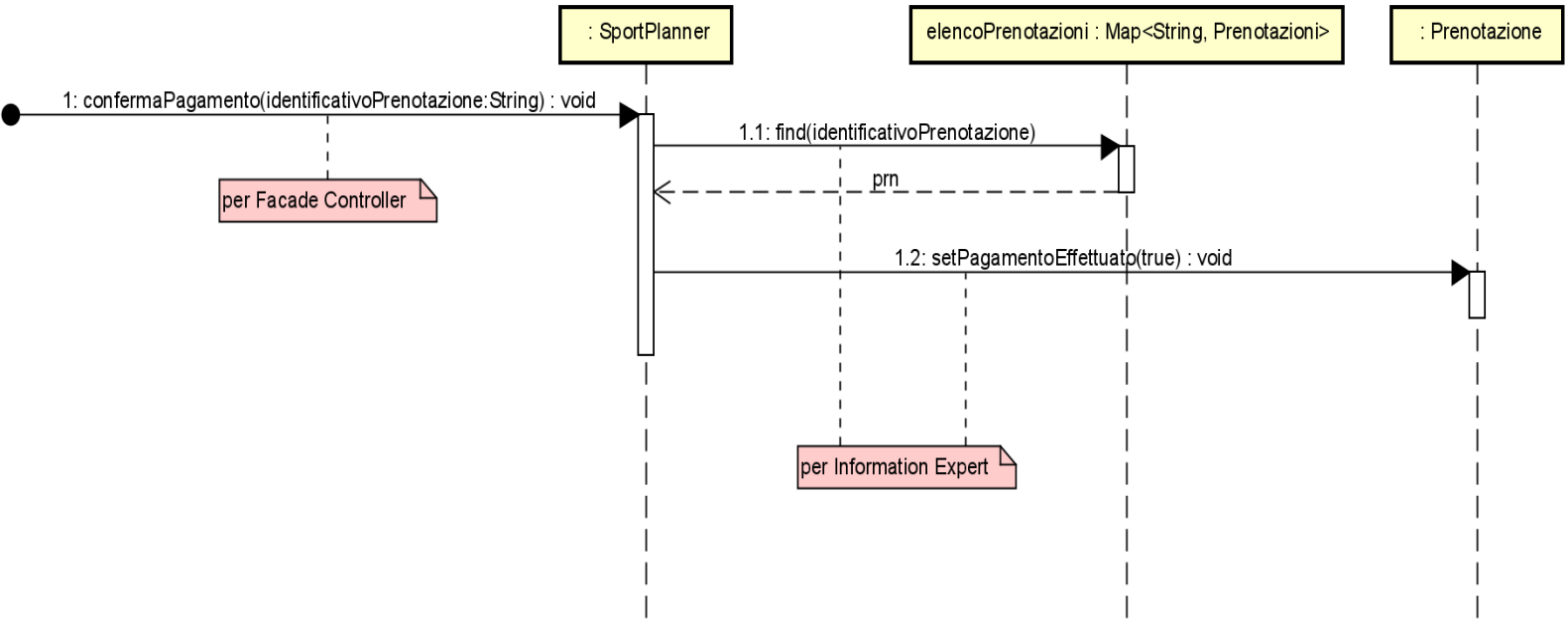
- **Gestione Prenotazione (UC2):**



- **Effettua pagamento:**



- **Conferma pagamento:**



1.4.2 Diagramma delle classi

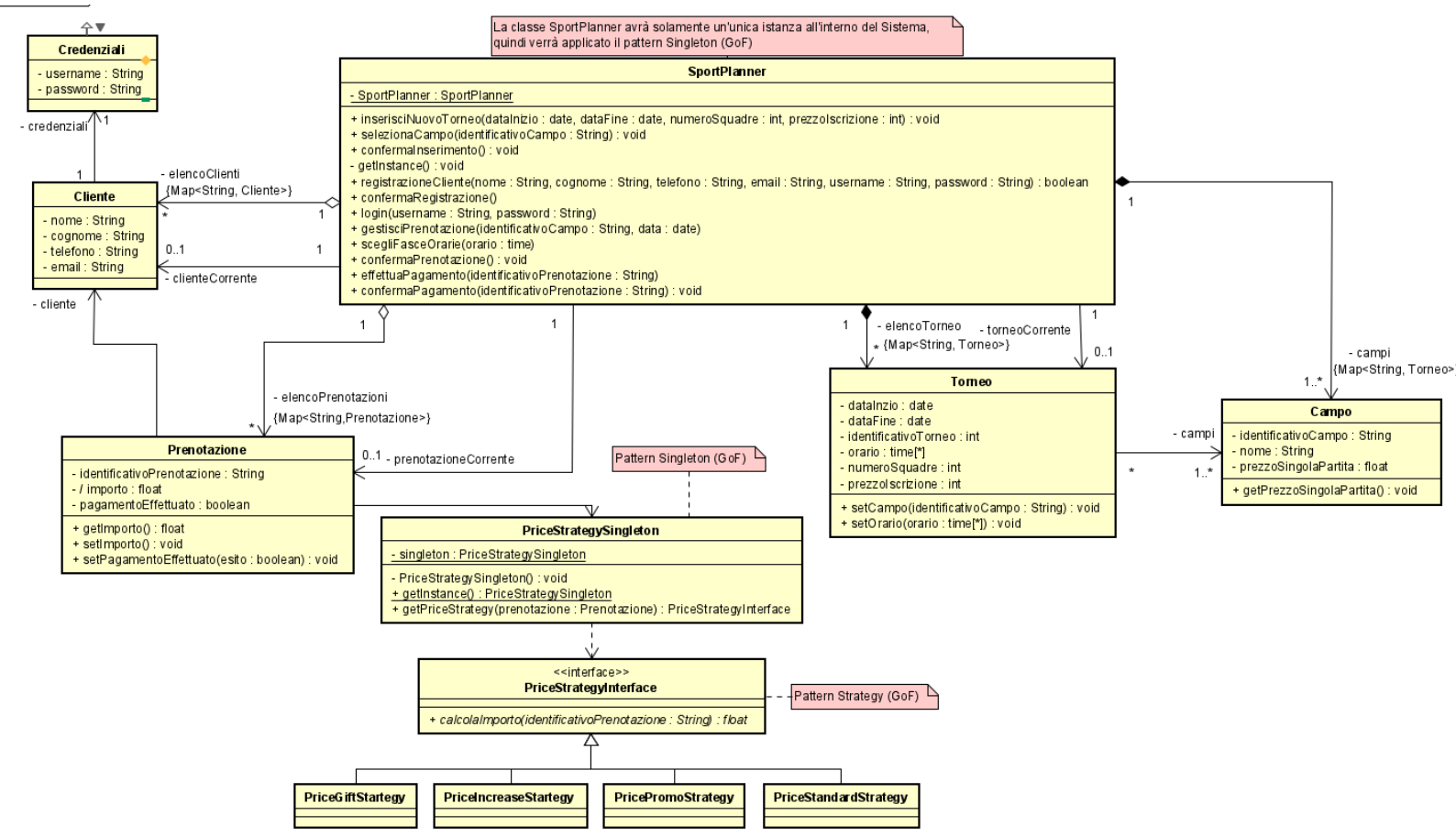
In questa iterazione è stato deciso che gli attributi della classe Partita potevano essere memorizzate direttamente dalla classe Prenotazione, il che ha portato all'eliminazione della classe Partita dal Diagramma delle classi, in quanto l'esistenza di una partita dipende dall'esistenza univoca della sua prenotazione e non è aggiunta da un amministratore in attesa di una Prenotazione come avverrà per il Torneo invece.
Inoltre come detto in precedenza per implementare le regole di dominio è stato usato il Design Pattern *Strategy* e il Design Pattern *Singleton.*

L'obiettivo di questa architettura è isolare un algoritmo all'interno di un oggetto, in maniera tale da risultare utile in diverse situazioni, dove sia necessario modificare dinamicamente gli algoritmi utilizzati da un'applicazione.

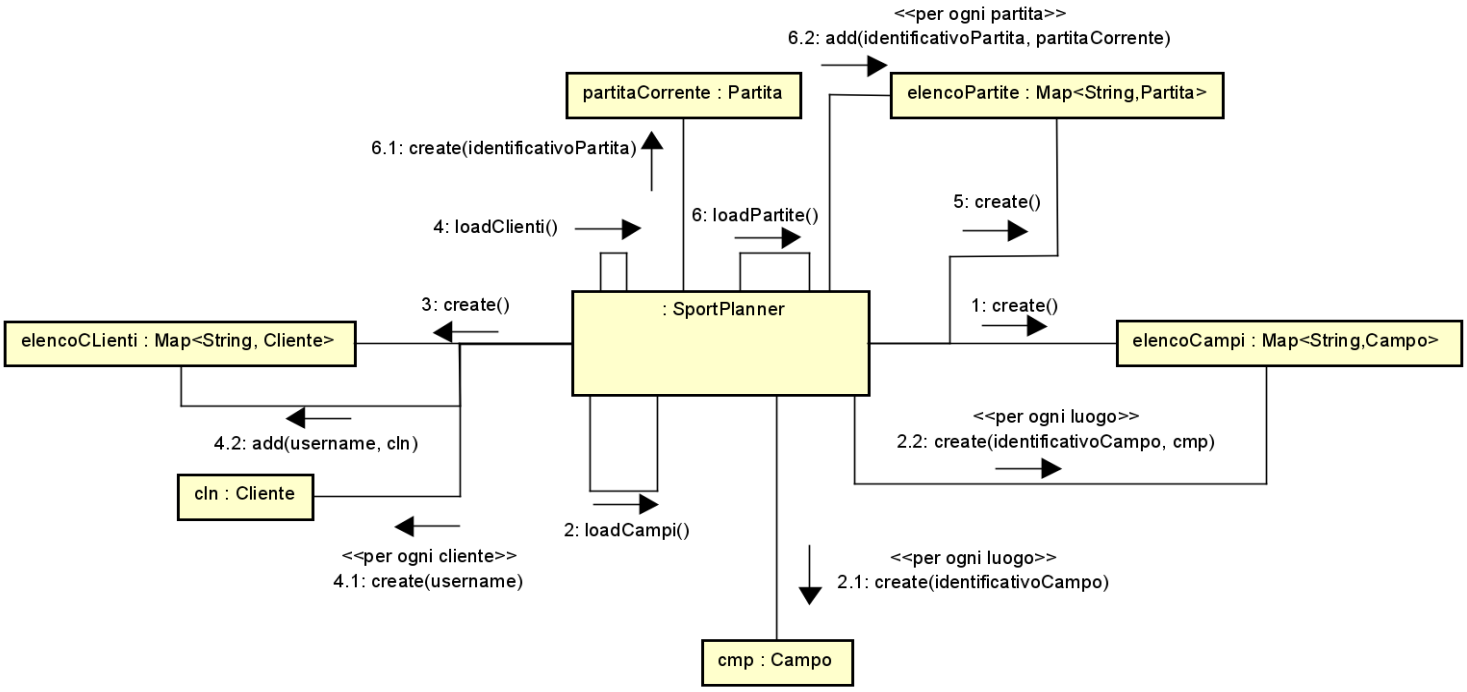
Per fare ciò sono state aggiunte:

- Classe PriceStrategySingleton;
- Interfaccia PriceStrategyInterface
- Classe PriceStandardStrategy;
- Classe PricePromoStrategy;
- Classe PriceIncreaseStrategy;
- Classe PriceGiftStrategy;

Potrà essere aggiunta una qualunque classe che implementi PriceStrategyInterface e che ridefinisca un proprio modo di calcolare l'importo.



1.4.3 Caso d'uso d'avviamento



1.5 Implementazione

Per questa iterazione sono state fatte le seguenti scelte progettuali:

- Si è scelto di non utilizzare un database per memorizzare i dati in maniera persistente, ma piuttosto di mantenerli in memoria principale. Sarà quindi necessario in fase di avviamento caricare i dati in memoria.
- È stata implementata una semplice interfaccia grafica tramite Java Swing.

1.6 Testing

I test sono stati eseguiti sulla classe SportPlanner.

- *login*
  - Il Cliente inserisce le proprie credenziali e diventa *clienteCorrente*.
- *gestisciPrenotazione*
  - Viene creata l'istanza di Prenotazione e di Partita
- *confermaInserimento*
  - Un torneo corrente per SportPlanner viene correttamente aggiunto all'elenco di tornei di SportPlanner
- scegliFasciaOraria
  - viene settato l'attributo *orario* nell'istanza di Partita