PawBookings: Elaborazione - Iterazione 1

Giuseppe Leocata, Alberto Provenzano, Daniele Lucifora

Introduzione

Conclusa la fase di ideazione, si passa alla fase di elaborazione.
Scopo delle iterazioni seguenti sarà quello di, oltre che affinare gli elaborati prodotti durante la fase di ideazione, applicare l'analisi e la progettazione orientata agli oggetti.
Per l'iterazione 1, sono stati scelti i seguenti requisiti:

- Implementazione dello scenario principale di successo del caso d'uso *UC1: Iscrizione a un corso.*
- Implementazione dello scenario principale di successo del caso d'uso *UC2: Gestisci prenotazione turno lezione.*
- Implementazione del caso d'uso d'avviamento necessario per inizializzare questa iterazione.
- dati solo in memoria principale

Aggiornamenti elaborati della fase di Ideazione

Per quanto concerne i due casi d'uso presi in esame per questa iterazione, sono stati individuati alcuni passi poco chiari se non errati. Sono state, pertanto, apportate alcune modifiche a questi.
Inoltre, è stata fatta l'assunzione che la prenotazione di un turno di una lezione venga interpretata dal sistema come la partecipazione effettiva del cane alla lezione. Di conseguenza, sono stati modificati gli elaborati di Visione. Non si è ritenuto invece necessario aggiornare il Glossario.

1. Analisi Orientata agli Oggetti

1.1. Modello di Dominio

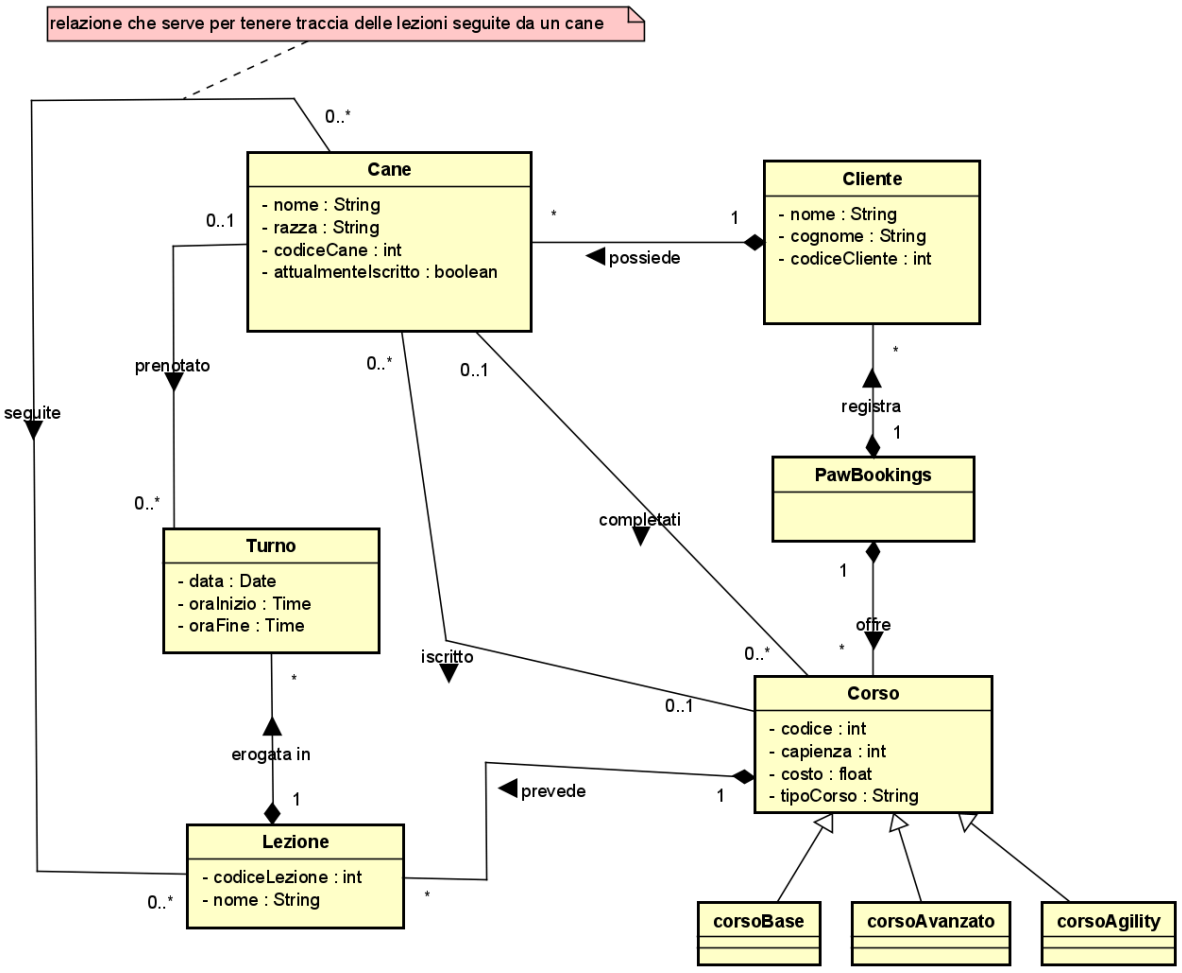
Essendo questo l'elaborato grafico in cui vengono identificate le classi, gli attributi e le associazioni di quei concetti significativi all'interno del dominio del problema, si elencano di seguito le classi concettuali individuate dopo un'attenta analisi dello scenario principale di successo relativo al caso d'uso *UC1: Iscrizione a un corso*:

- **Cliente**
- **Cane**
- **Sistema**
- **Corso**
- **Lezione**

In accordo con la natura incrementale del processo software Unified Process, si procede con l'analisi del flusso base del caso d'uso *UC2: Gestisci prenotazione turno lezione*.
Nello specifico, si riscontra l'esigenza di dover aggiungere la seguente classe concettuale:

- **Turno**

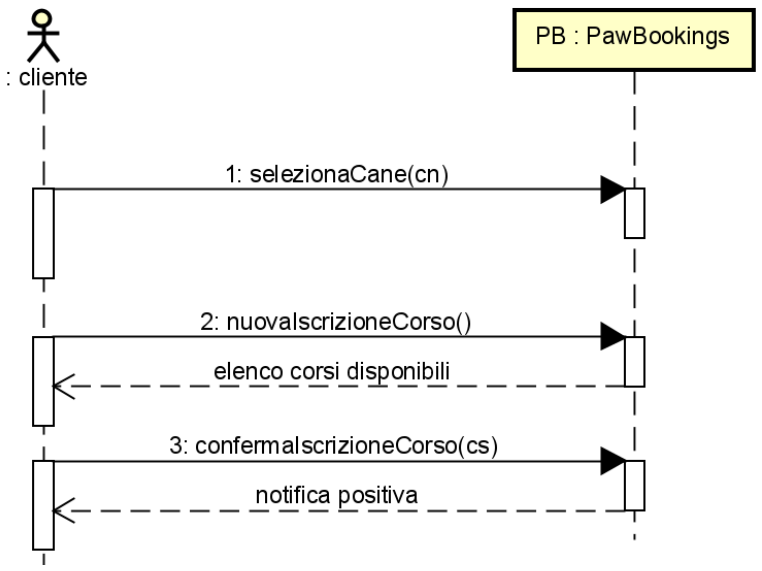
Dall'analisi dei due casi d'uso presi in esame, tenendo anche in considerazione i relativi attributi e associazioni, è stato ricavato il seguente Modello di Dominio:



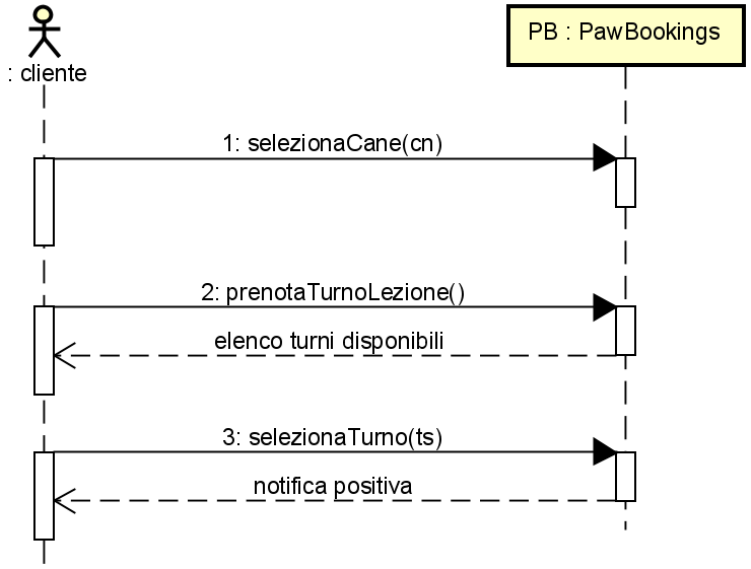
1.2. Diagrammi di Sequenza di Sistema (SSD)

Procedendo con l'OOA, il passo successivo consiste nella realizzazione dei Diagrammi di Sequenza di Sistema (SSD) relativi ai due casi d'uso prescelti.

1.2.1. SSD UC1



1.2.2. SSD UC2



1.3. Contratti delle Operazioni

Vengono ora descritte attraverso i Contratti le principali operazioni di sistema.

1.3.1. C01: confermaIscrizioneCorso

operazione:	confermaIscrizioneCorso(cs: Corso)
riferimenti:	Caso d'uso UC1: Iscrizione ad un corso
pre-condizioni:	- è stata recuperata l'istanza *cn* della classe Cane selezionata dall'utente
post-condizioni:	- l'istanza *cn* è stata associata all'istanza *cs* tramite l'associazione "iscritto" - è stata aggiornata *cs.capienza* - *cn.attualmenteIscritto* è diventato *true*

1.3.2. C02: selezionaTurno

operazione:	selezionaTurno(ts: Turno)
riferimenti:	Caso d'uso UC2: Gestisci prenotazione
pre-condizioni:	- è stata recuperata l'istanza *cn* della classe Cane selezionata dall'utente
post-condizioni:	- *cn* è stato associato a *ts* tramite l'associazione "prenotato" - *cn* è stata associata a Lezione tramite l'associazione "seguite"

2. Progettazione Orientata agli Oggetti

Si passa, in questa fase, alla definizione degli oggetti software a partire dagli oggetti concettuali precedentemente individuati mantenendo un salto rappresentazionale basso. Si vogliono, inoltre, definire anche le loro responsabilità e le loro interazioni al fine di soddisfare i requisiti.

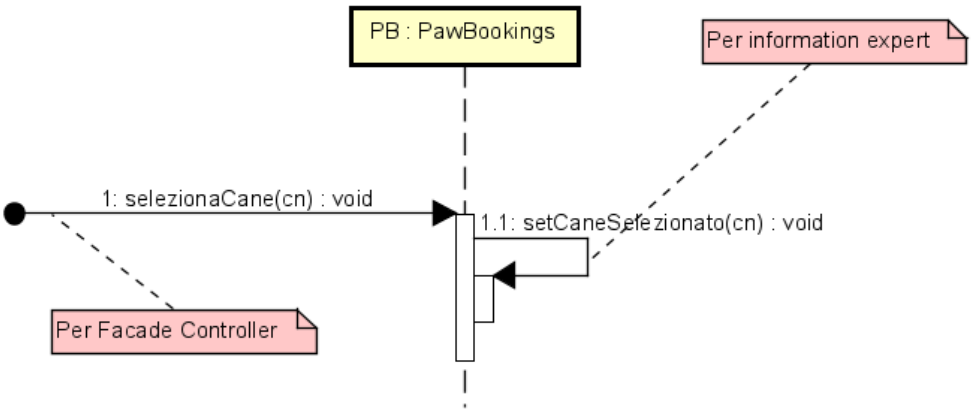
2.1. Pattern applicati

Si è applicato il pattern GoF Singleton per la Classe software PawBookings al fine di avere un'unica istanza di quest'ultima all'interno del sistema.
Sono stati inoltre applicati alcuni dei principali pattern GRASP (Information Expert, Controller, Low Coupling e High Cohesion) al fine di seguire una progettazione guidata dalle responsabilità.

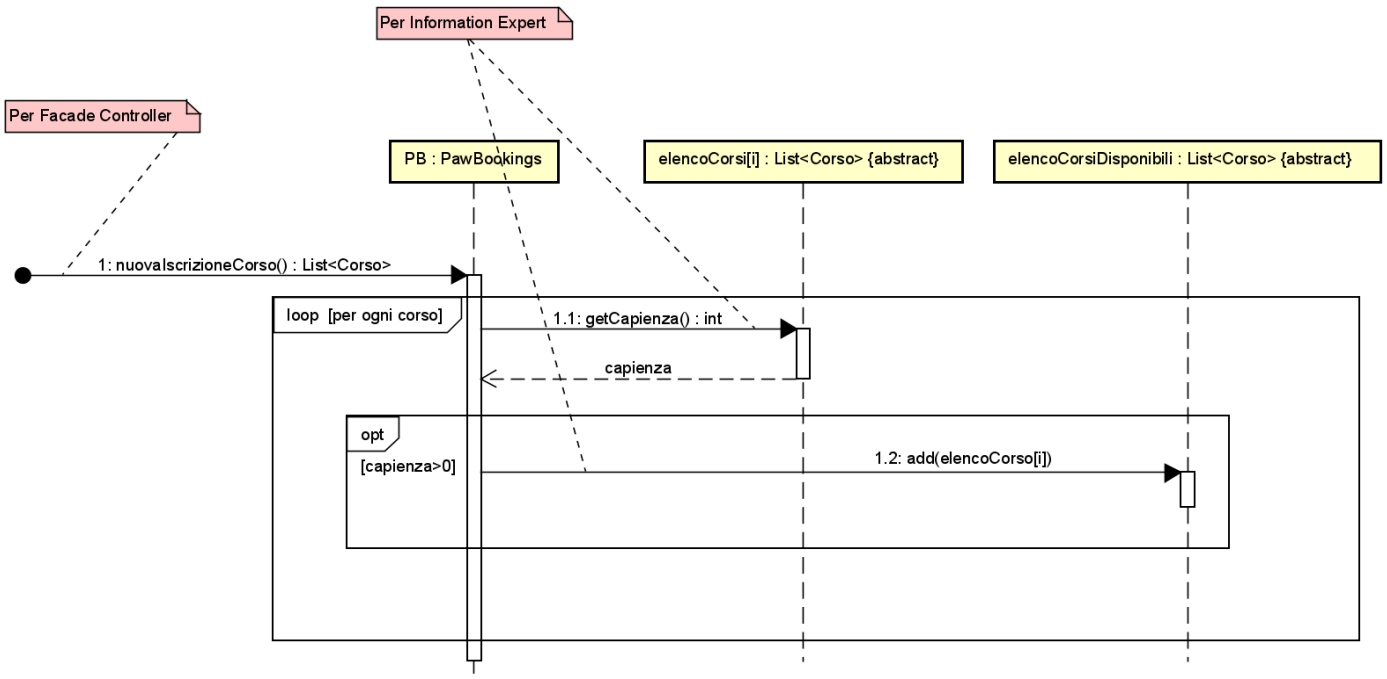
2.2. Diagrammi di Sequenza (SD)

La prima scelta di progetto da fare è sul Controller per i vari casi d'uso: si è scelto di usare PB (PawBookings, ovvero la classe concettuale che astrae il sistema) come facade controller.

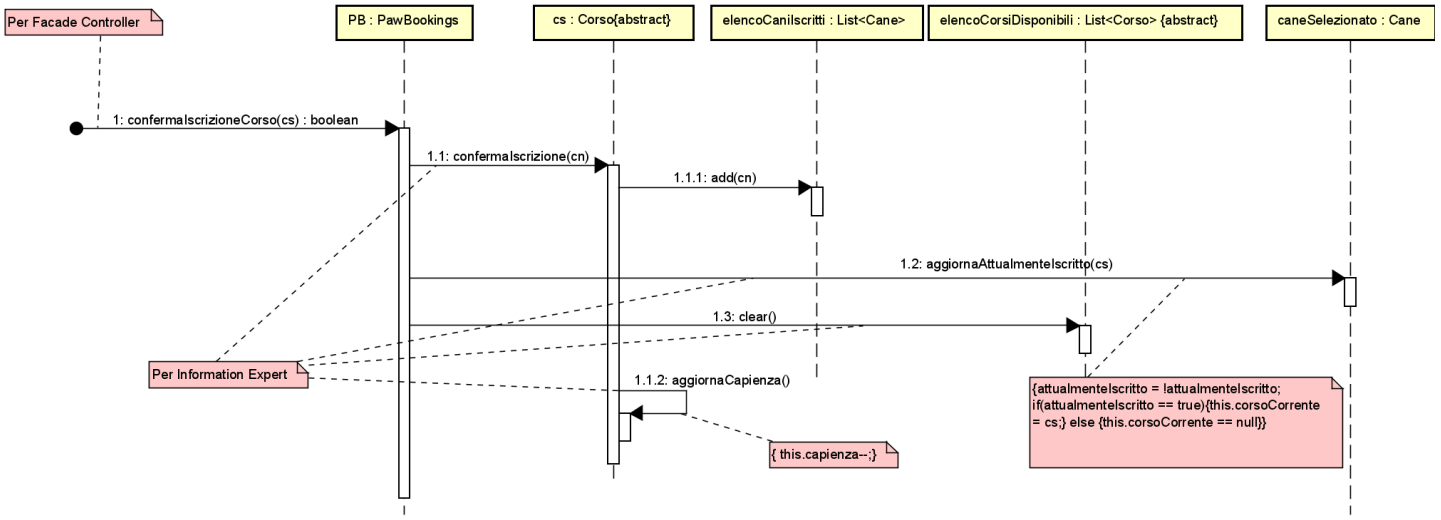
2.2.1. UC1/UC2 SD0:



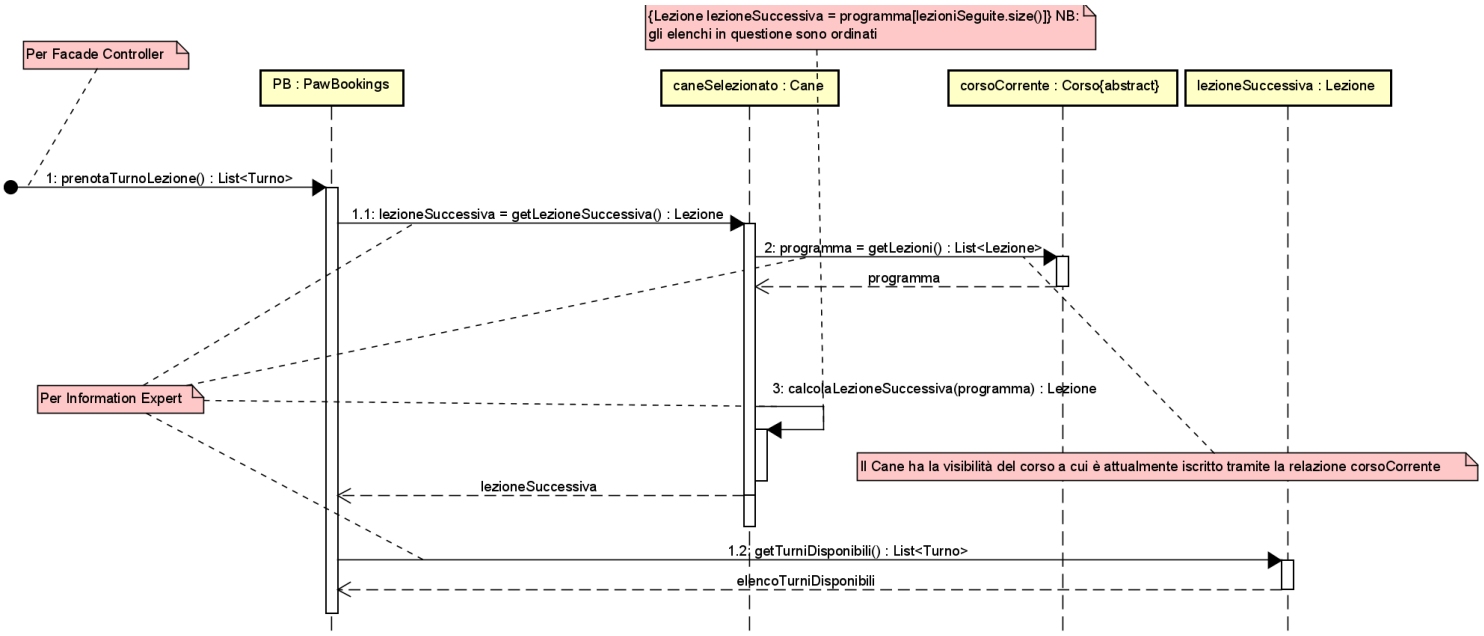
2.2.2. UC1 SD1



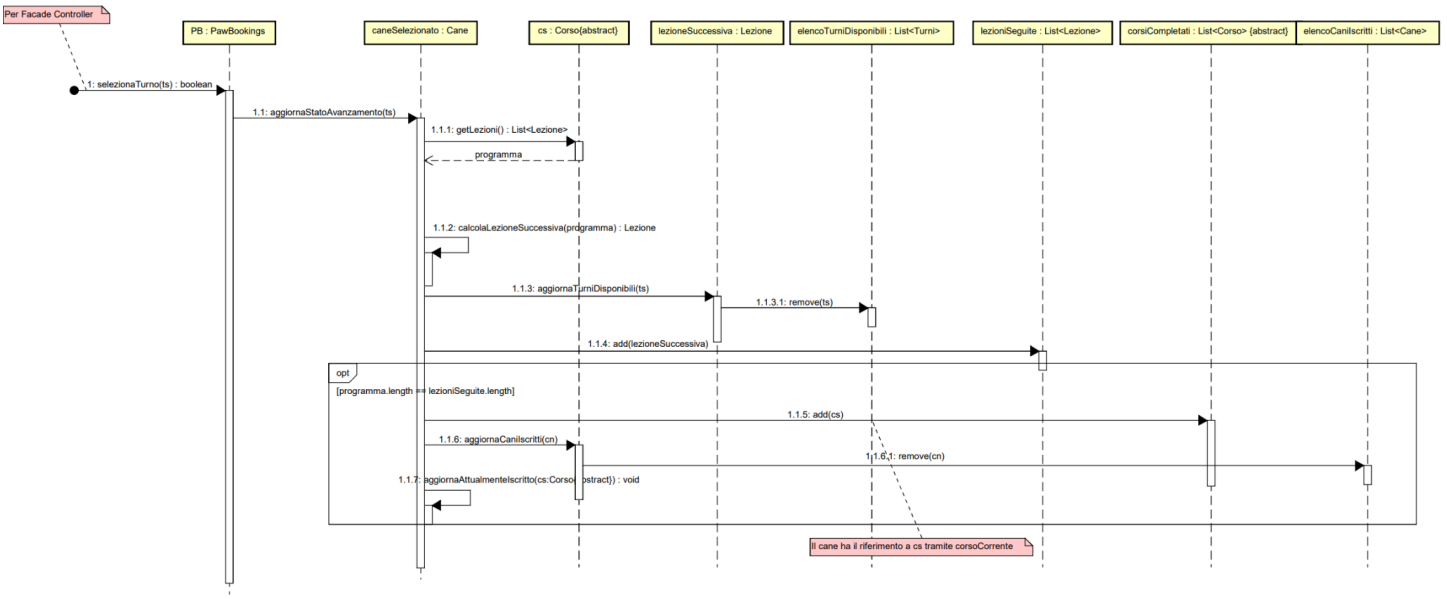
2.2.3. UC1 SD2



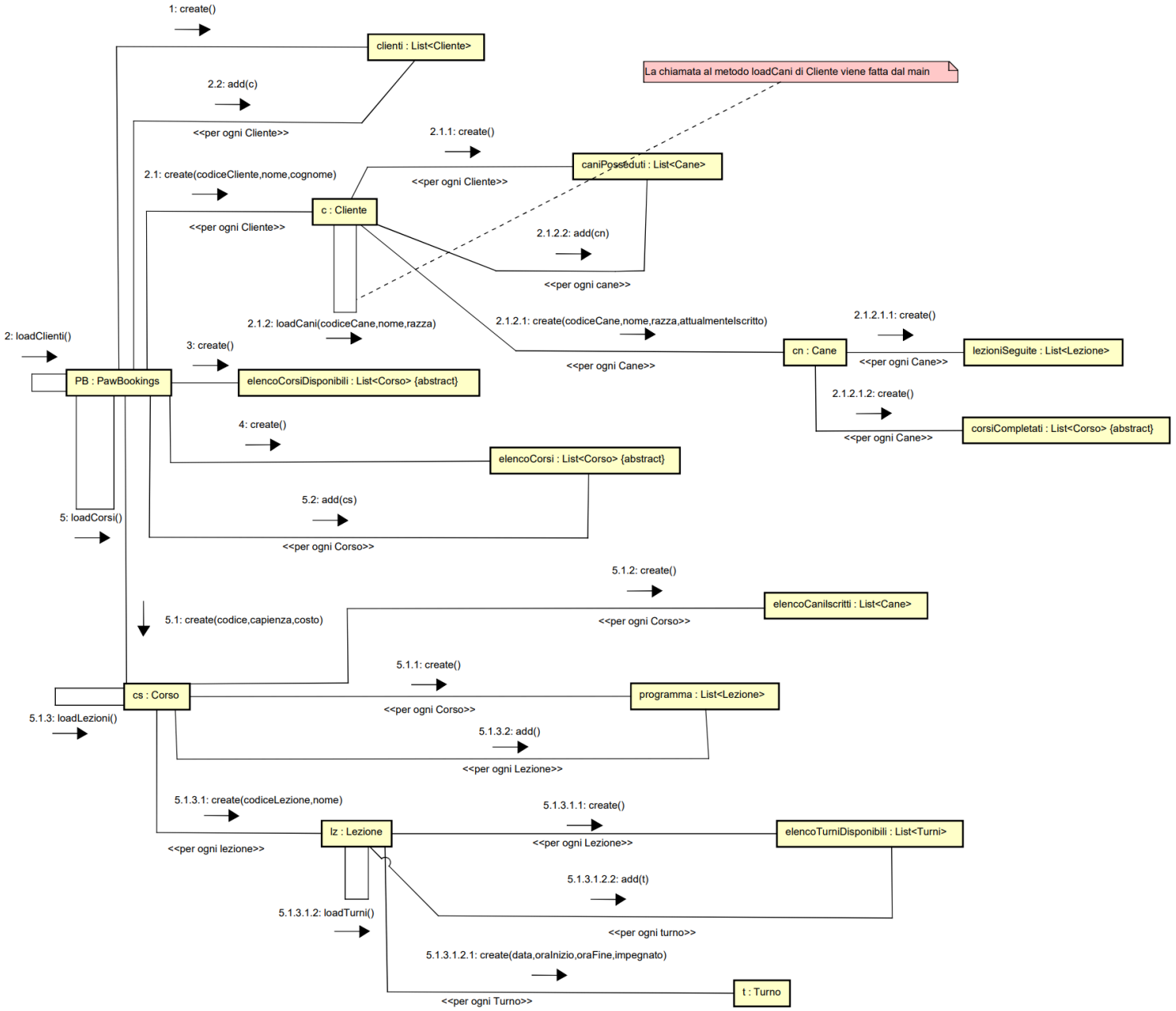
2.2.4. UC2 SD1



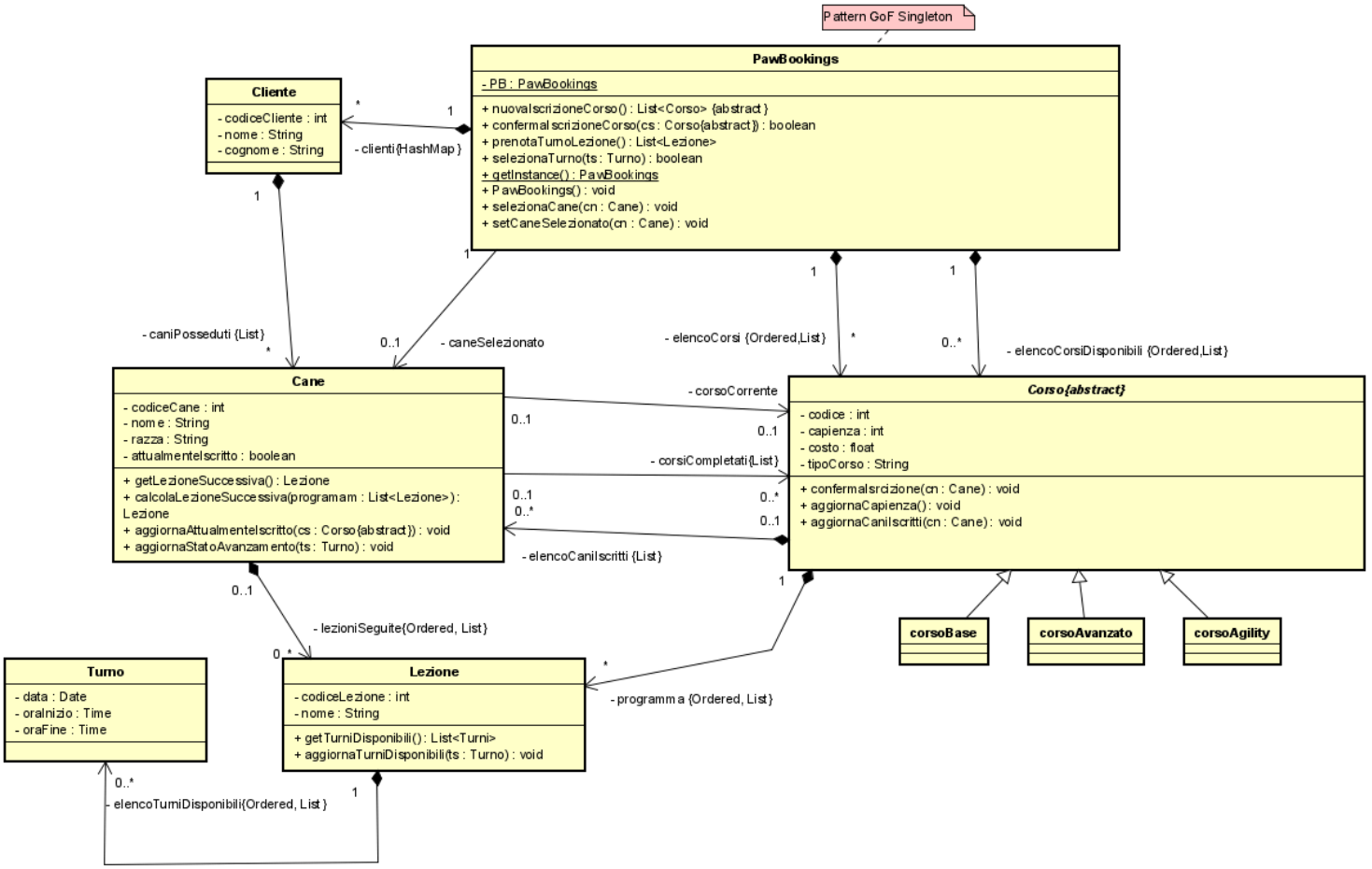
2.2.5. UC2 SD2


Per Facade Controller
PB : PawBookings
caneSelezionato : Cane
cs : Corso{abstract}
lezioneSuccessiva : Lezione
elencoTurniDisponibili : List<Turni>
lezioniSeguite : List<Lezione>
corsiCompletati : List<Corso> {abstract}
elencoCanilscritti : List<Cane>
1: selezionaTurno(ts) : boolean
1.1: aggiornaStatoAvanzamento(ts)
1.1.1: getLezioni() : List<Lezione>
programma
1.1.2: calcolaLezioneSuccessiva(programma) : Lezione
1.1.3: aggiornaTurniDisponibili(ts)
1.1.3.1: remove(ts)
1.1.4: add(lezioneSuccessiva)
opt
[programma.length == lezioniSeguite.length]
1.1.5: add(cs)
1.1.6: aggiornaCanilscritti(cn)
1.1.6.1: remove(cn)
1.1.7: aggiornaAttualmenteIscritto(cs:Corso{abstract}) : void
Il cane ha il riferimento a cs tramite corsoCorrente

2.2.6. SD Caso d'uso d'avviamento



2.3. Modello di Progetto (Diagramma delle Classi di Progetto)



3. Testing

Il testing è un processo fondamentale nello sviluppo del software che consiste nell'esaminare un sistema per valutarne le funzionalità, l'affidabilità e le prestazioni.
Per eseguire il testing dell'applicazione PawBookings sono stati utilizzati i test unitari (Unit Test), i quali permettono di effettuare dei test su singole unità di codice per poterne verificare il corretto funzionamento.
È stato seguito un approccio Top-Down, testando prima le unità più grandi, utilizzando un criterio di scelta dei metodi da testare che, per questa prima iterazione, tenga conto del flusso degli scenari principali di successo UC1 e UC2.
In particolar modo sono stati individuati i seguenti metodi delle relative classi:

- PawBookings
    - nuovaIscrizioneCorso
        - viene verificato che la lista dei corsi disponibili contenga i corsi la cui capienza sia maggiore di 0
    - selezionaTurno
        - viene verificato che il valore booleano restituito, relativamente al turno passato come parametro, sia false se viene passato null come parametro
        - viene verificato che il valore booleano restituito, relativamente al turno passato come parametro, sia true
    - selezionaCane
        - viene verificato che una volta chiamato il metodo selezionaCane passando un oggetto Cane questo sia stato assegnato al riferimento caneSelezionato di PawBookings
        - viene verificato che il nome del Cane passato al metodo selezionaCane coincida con quello del riferimento caneSelezionato di PawBookings
- Cane
    - aggiorna Attualmente Iscritto
        - viene verificato che una volta chiamato questo metodo l'attributo attualmenteIscritto di Cane diventi true (se prima era false)
        - viene verificato che una volta chiamato questo metodo il codice del corsoCorrente di Cane sia lo stesso del corso passato come parametro al metodo sotto test
    - getLezioneSuccessiva
        - viene verificato che la lezione successiva sia quella seguente all'ultima lezione seguita, relativamente al programma del corso in cui il cane è iscritto
    - aggiornaAvanzamentoCorso
        - viene verificato che la dimensione dell'elenco delle lezioni seguite dal Cane aumenti di 1 a seguito della chiamata del metodo sotto test passando come parametro un Turno relativo alla lezione successiva
        - viene verificato che l'elenco delle lezioni seguite dal Cane contenga la lezione relativa al Turno passato come parametro al metodo sotto test
    - completamentoCorso (test legato al metodo aggiornaAvanzamentoCorso)
        - viene verificato che il cane selezionato non sia più iscritto ad un corso
        - viene verificato che l'attributo corsoCorrente di Cane sia diventato null
        - viene verificato che la lista delle lezioni seguite dal Cane corrisponda al numero di lezioni del programma del corso seguito
        - viene verificato che il corso completato sia presente nell'elenco dei corsi completati dal Cane
        - viene verificato che il cane selezionato non sia presente nell'elenco dei cani iscritti al corso seguito

- Corso
    - confermaIscrizione
        - viene verificato che l’elenco dei cani iscritti del Corso sia aumentato di 1 a seguito della chiamata del metodo sotto test passando un’istanza di Cane
    - aggiornaCapienza
        - viene verificato che, in seguito alla chiamata del metodo sotto test, la capienza del corso sia stata decrementata

A seguito dell'esecuzione dei testi unitari, eseguiti utilizzando Junit, è stato possibile correggere le porzioni di codice i cui test sono risultati fallimentari.