PawBookings: Elaborazione - Iterazione 4

Giuseppe Leocata, Alberto Provenzano, Daniele Lucifora

Introduzione

Per l'iterazione 4, sono stati scelti i seguenti requisiti:

- Implementazione degli scenari alternativi dei casi d'uso *UC1, UC2, UC3, UC4 e UC11.*
  - relativamente all'UC4, si è tralasciata l'estensione *1a* in quanto coincide con una regola di dominio; pertanto, è stata rimossa.
- Implementazione di un nuovo caso d'uso *UC14: Timbra Turno.*
- Implementazione del caso d'uso d'avviamento necessario per inizializzare questa iterazione.
- Dati solo in memoria principale.

Aggiornamenti elaborati della fase di Ideazione

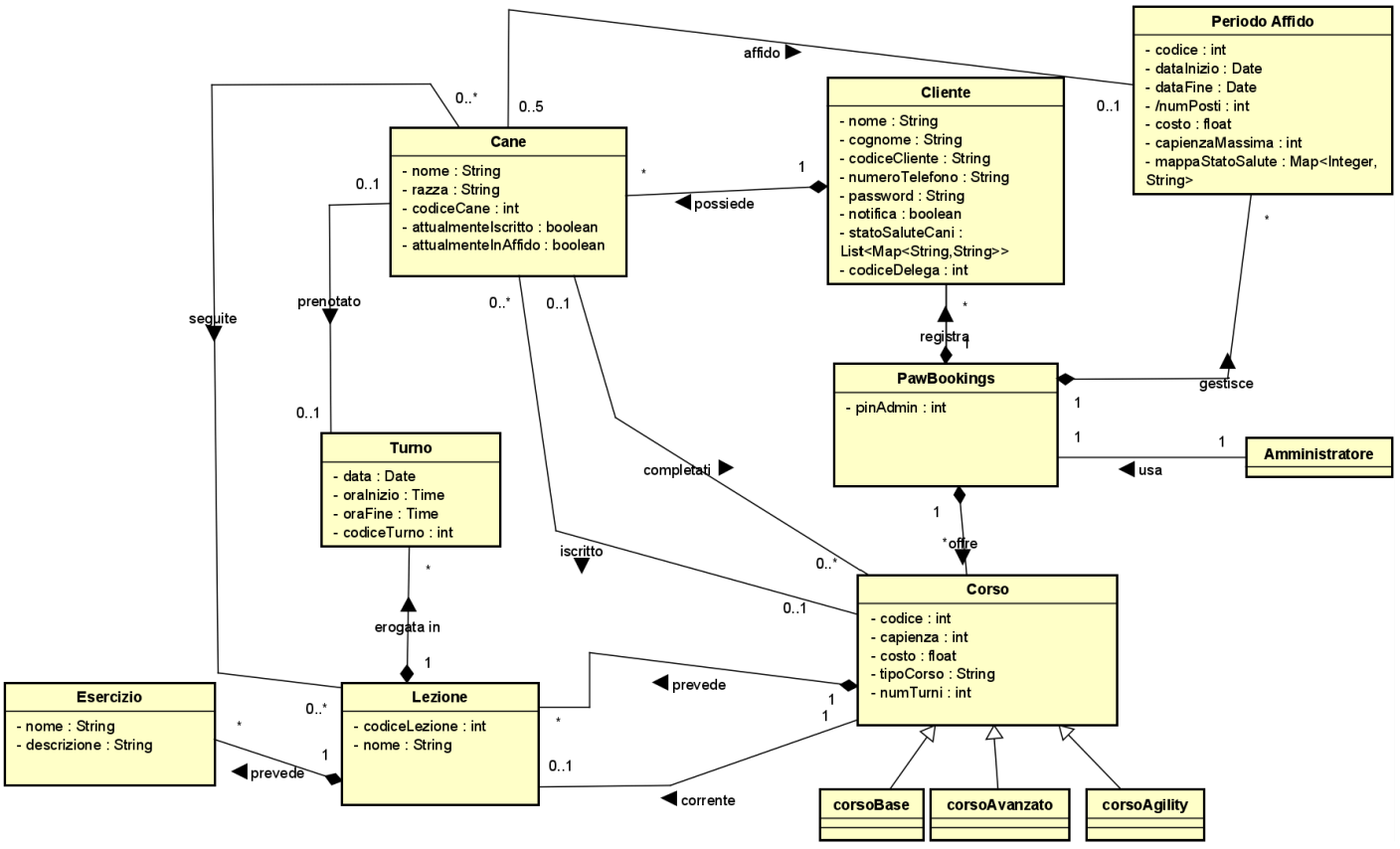
E' stato aggiornato il Modello dei casi d'uso in termini di aggiunta di scenari alternativi ed interi casi d'uso (*UC14*) ed il Glossario, dove è stato definito il concetto del *timbro di un turno*. Per lo stesso motivo, inoltre, è stata aggiornata la Visione: il timbro di un turno può avvenire anche nel caso di turni scaduti in quanto non farlo significherebbe costringere il Cliente a recuperare una lezione (se la perde).

1. Analisi Orientata agli Oggetti

Proseguendo con lo stesso approccio utilizzato nelle iterazioni precedenti, verranno riproposti: Modello di Dominio, SSD e Contratti delle Operazioni.

1.1. Modello di Dominio

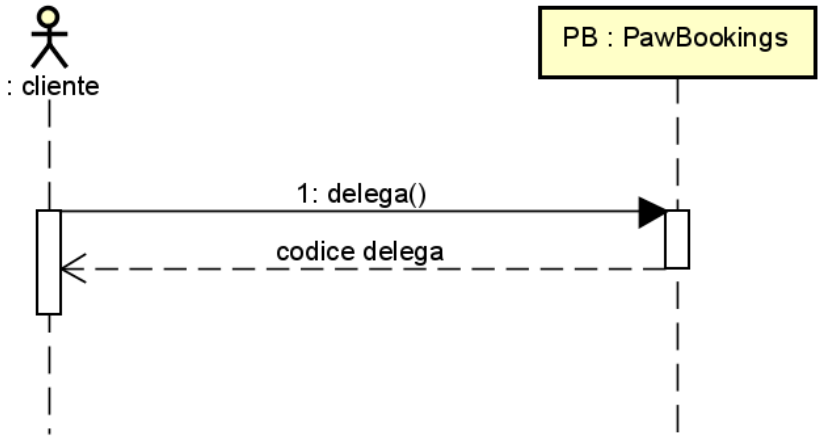
Rimasto pressoché invariato. Le uniche differenze stanno nell'aggiunta dell'attributo *codiceDelega* in **Cliente** e nella modifica della molteplicità dell'associazione "prenotato" tra **Cane** e **Turno**.



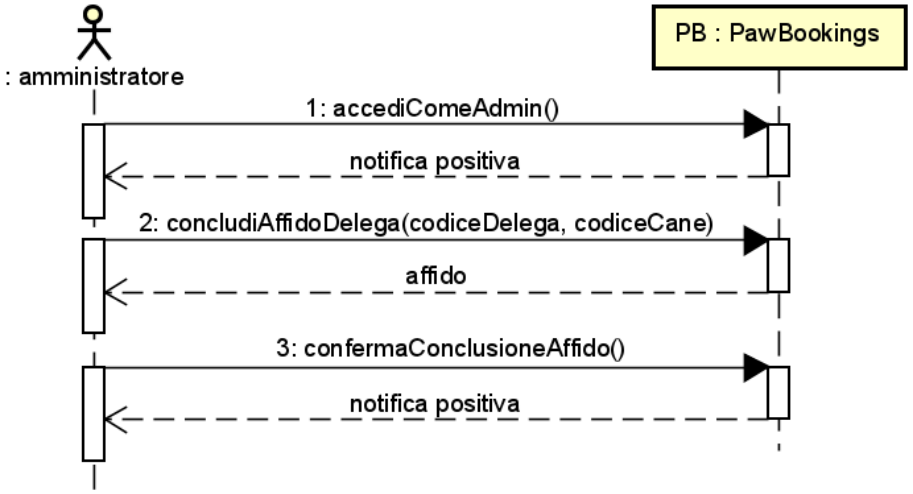
1.2. Diagrammi di Sequenza di Sistema (SSD)

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

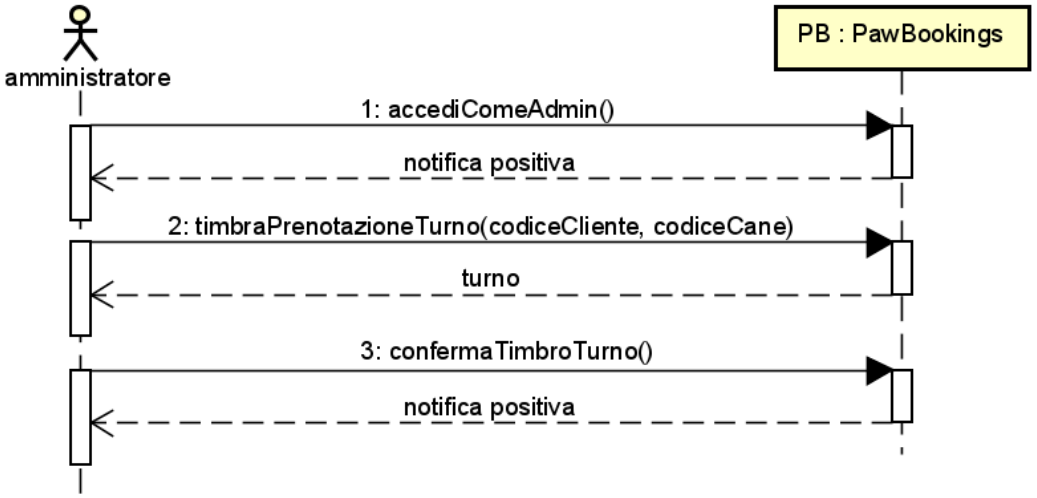
1.2.1. SSD UC4 - Estensione 1



1.2.2. SSD UC4 - Estensione 2



1.2.3. SSD UC14



1.3. Contratti delle Operazioni

1.3.1. C01: confermaTimbroTurno

operazione:	confermaTimbroTurno()
riferimenti:	Caso d'uso UC14: Timbra turno
pre-condizioni:	- è stata recuperata l'istanza cn di Cane - è stata recuperato il Turno t associato a cn tramite l'associazione "prenotato"
post-condizioni:	- cn è dissociato da t

2. Progettazione Orientata agli Oggetti

Si prosegue adesso con lo sviluppo degli SD e del DCD.

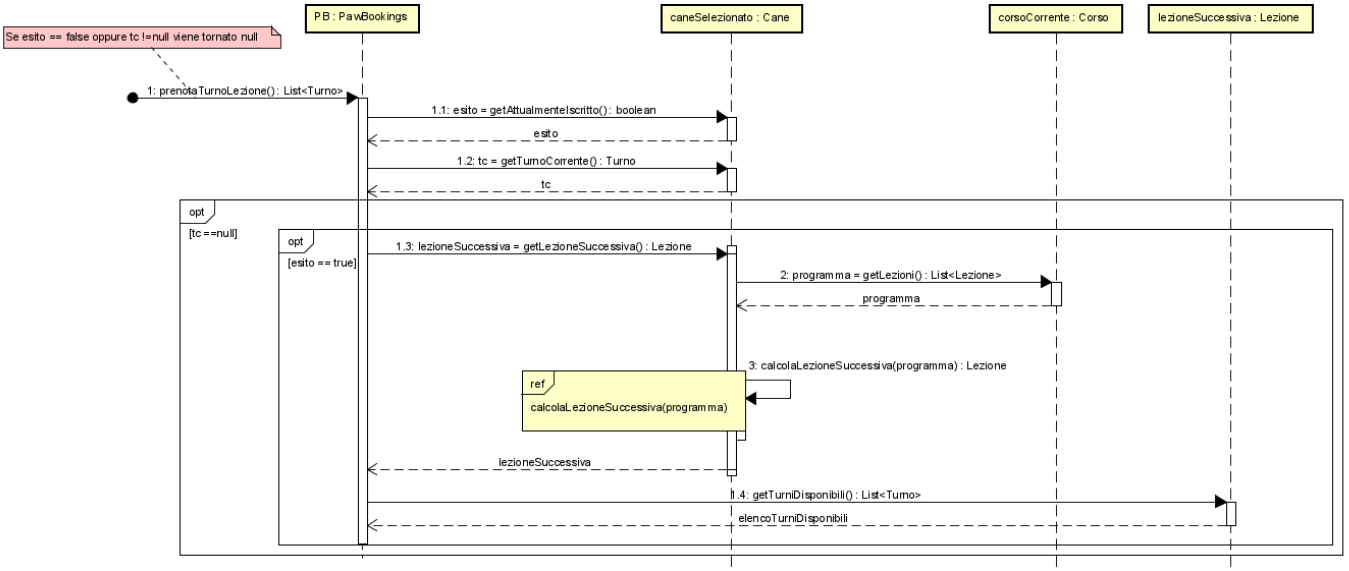
2.1. Pattern applicati

Sono stati applicati i pattern GRASP (Creator, Information Expert, Controller, Low Coupling e High Cohesion).

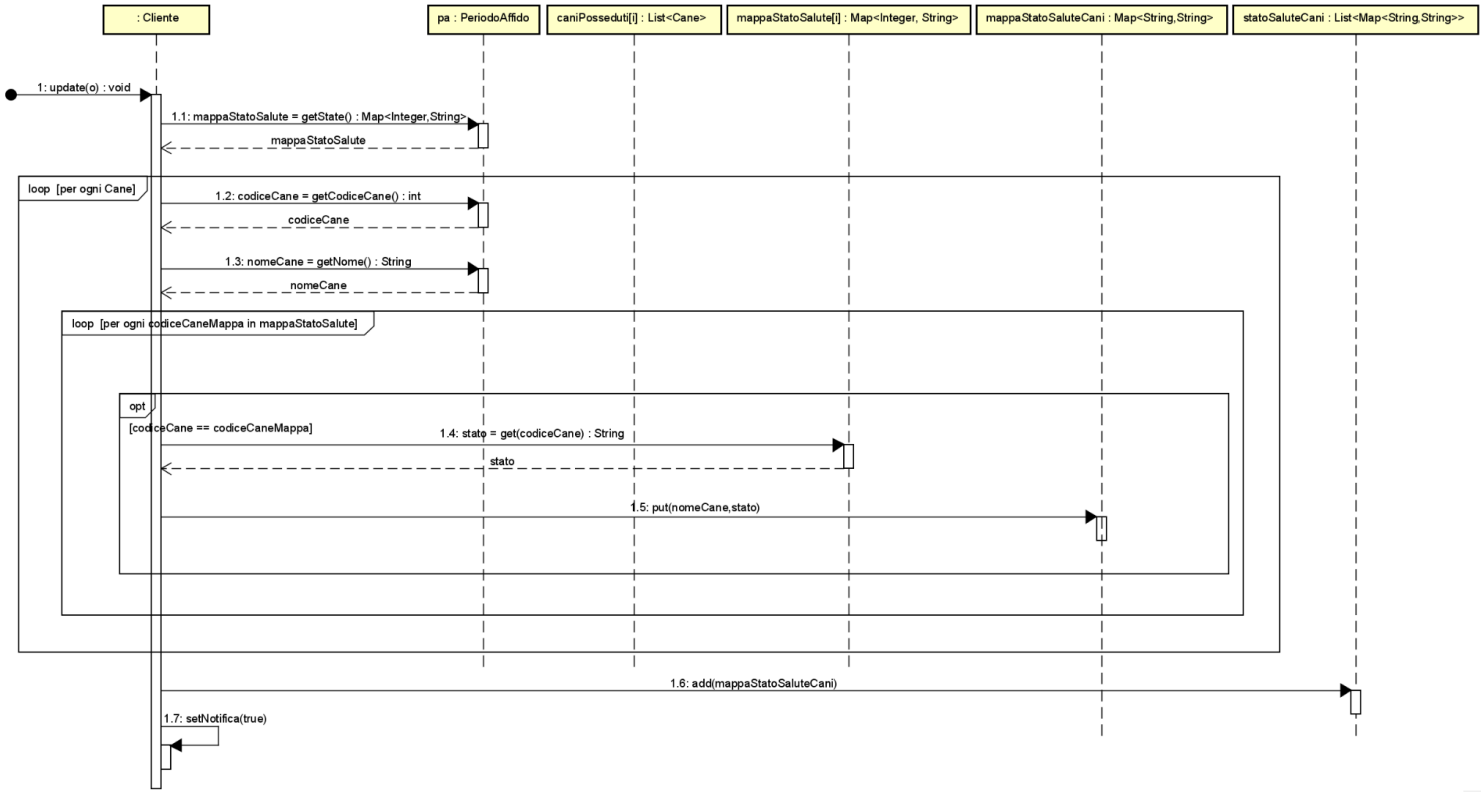
2.2. Diagrammi di Sequenza (SD)

2.2.1. SD delle iterazioni precedenti (aggiornati)

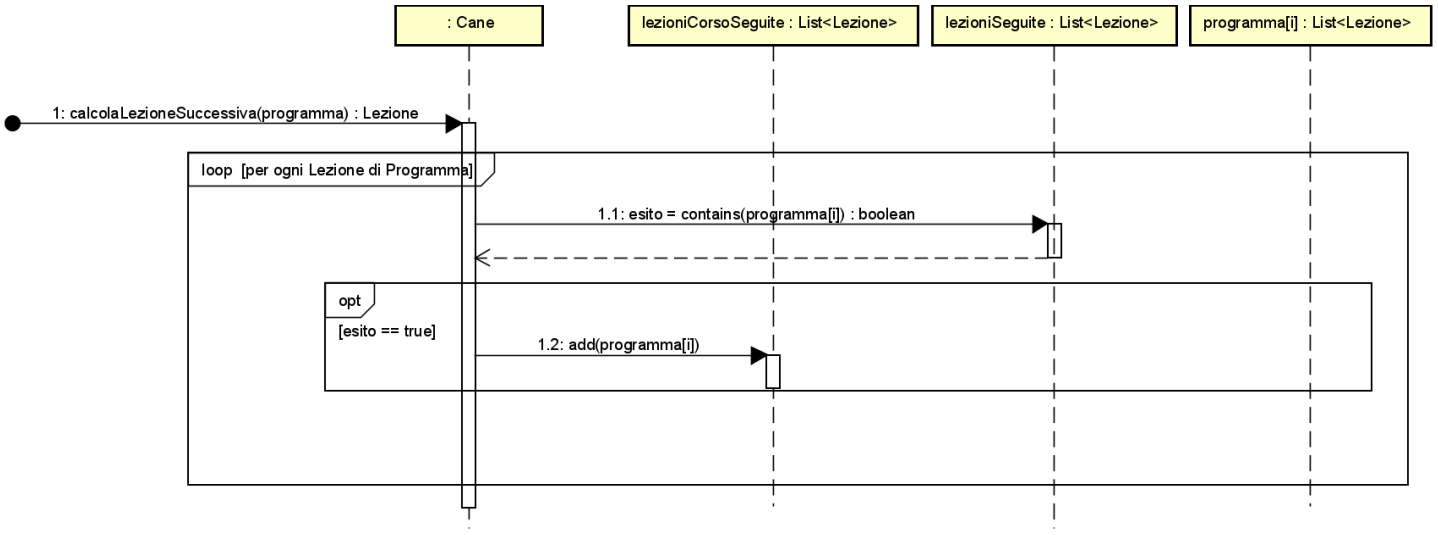
2.2.1.1. UC2 SD1: prenotaTurnoLezione



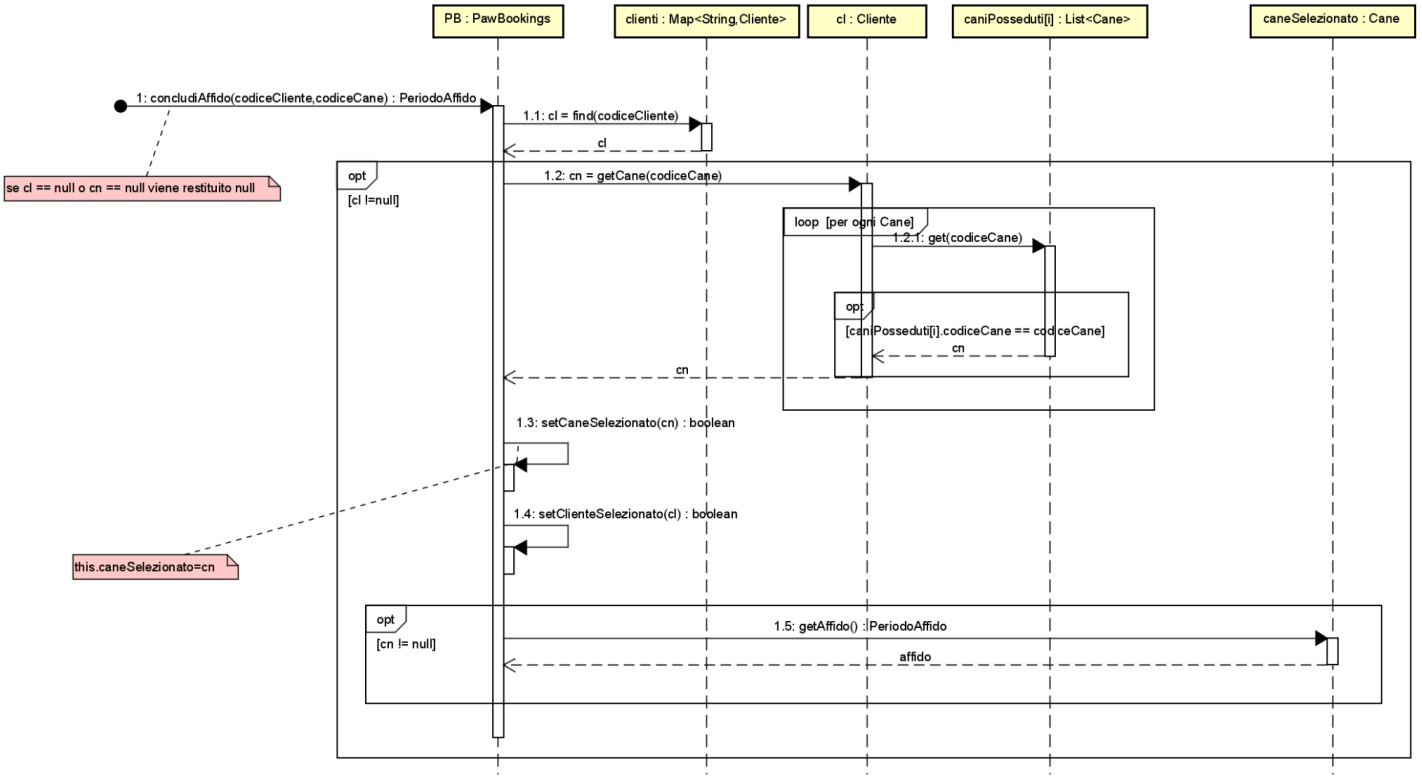
2.2.1.2. update



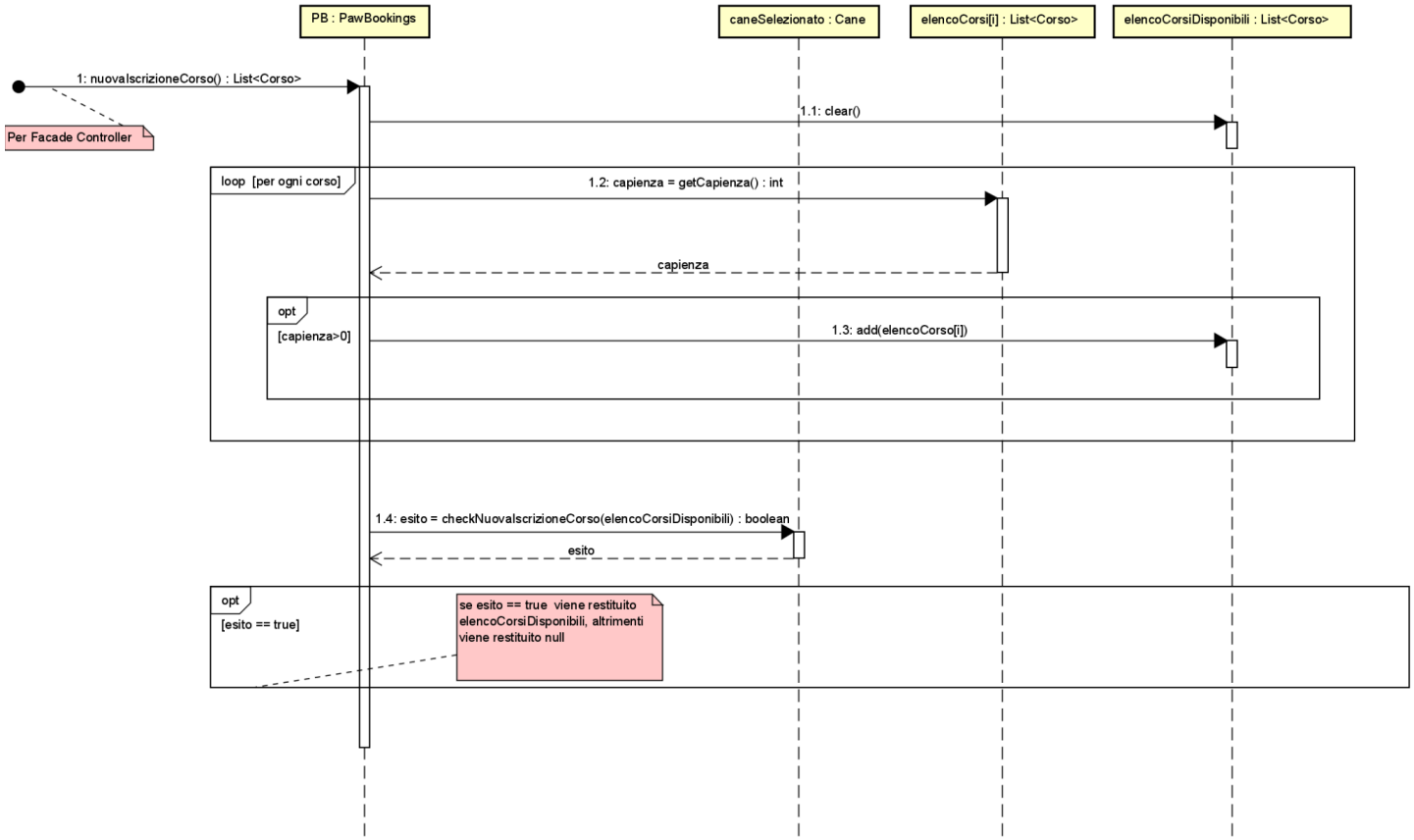
2.2.1.3. calcolaLezioneSuccessiva



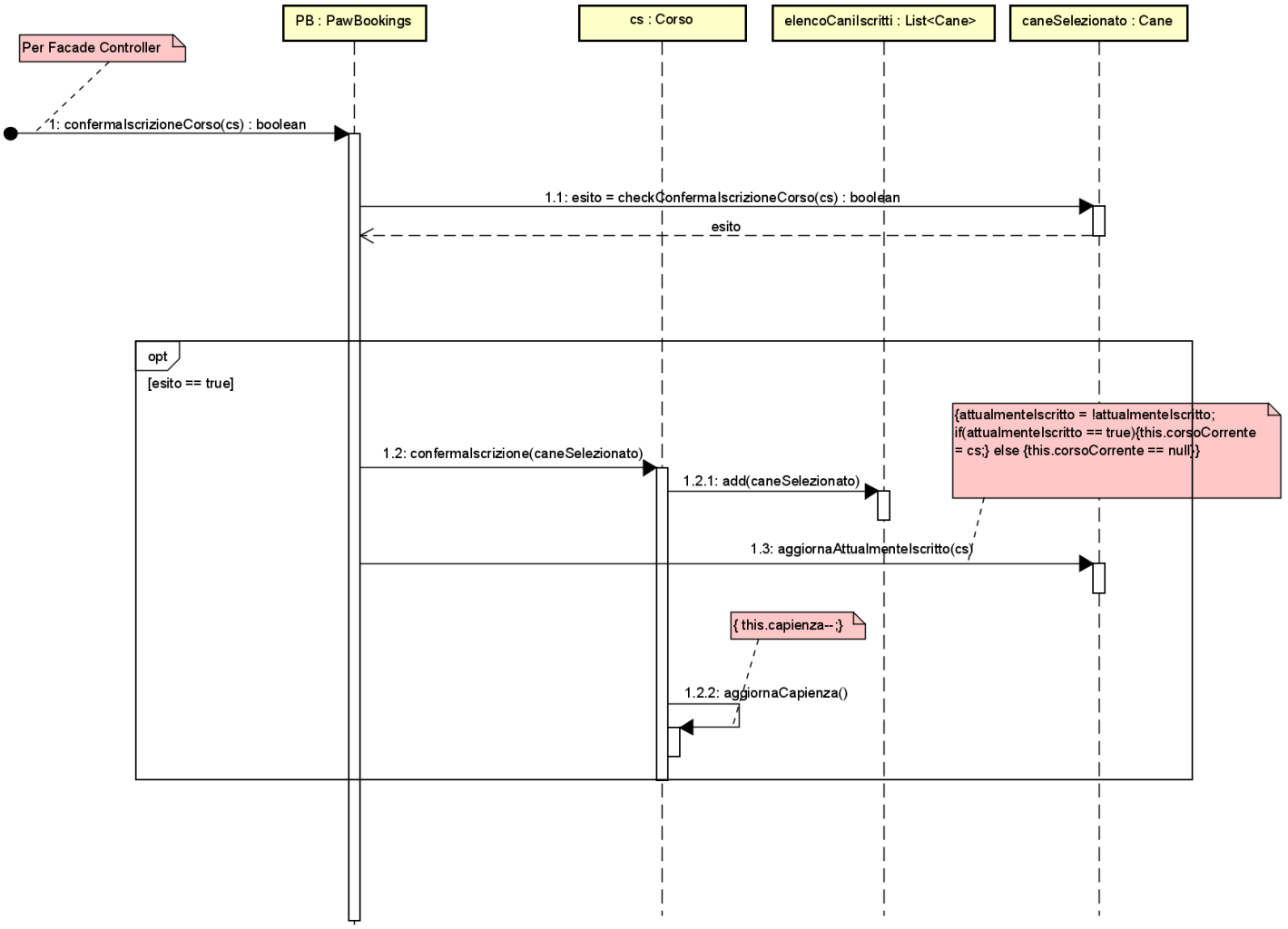
2.2.1.4. UC4 SD1: concludiAffido



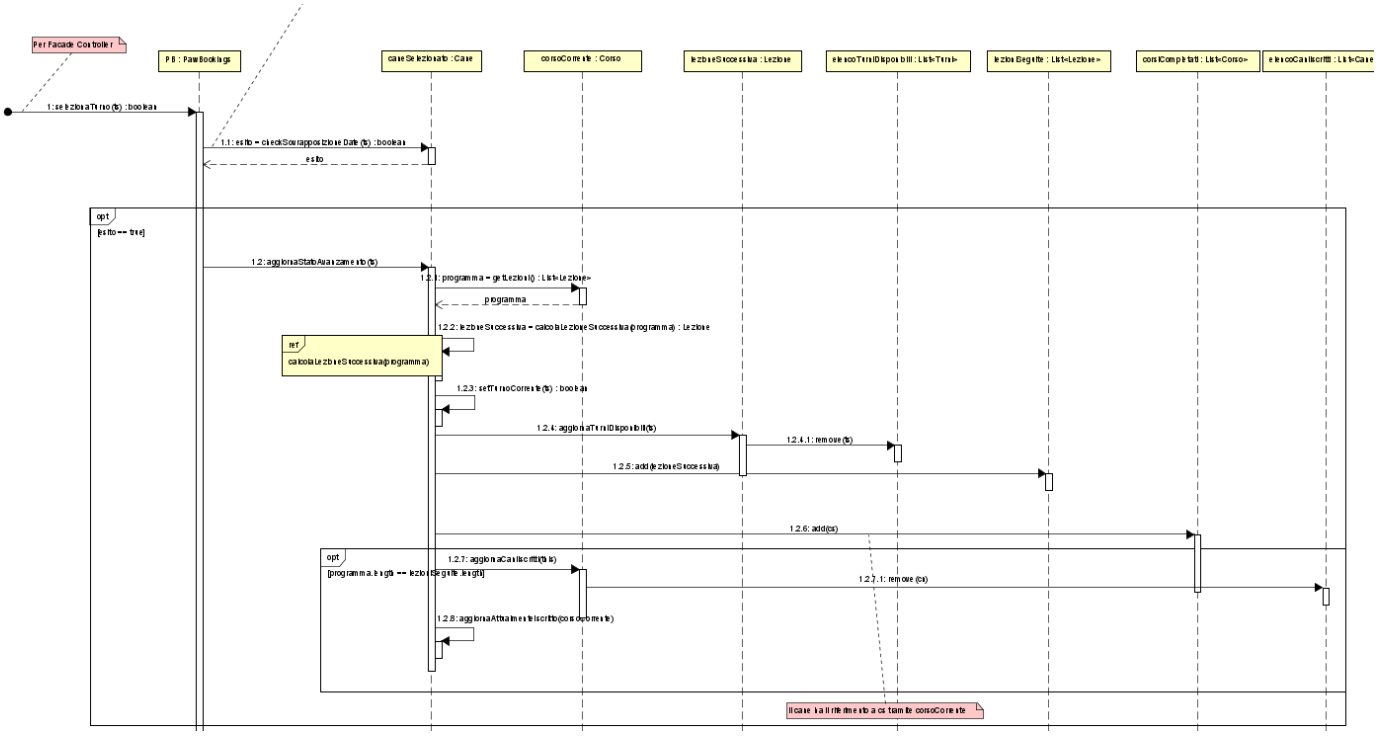
2.2.1.5. UC1 SD1: nuovaIscrizioneCorso



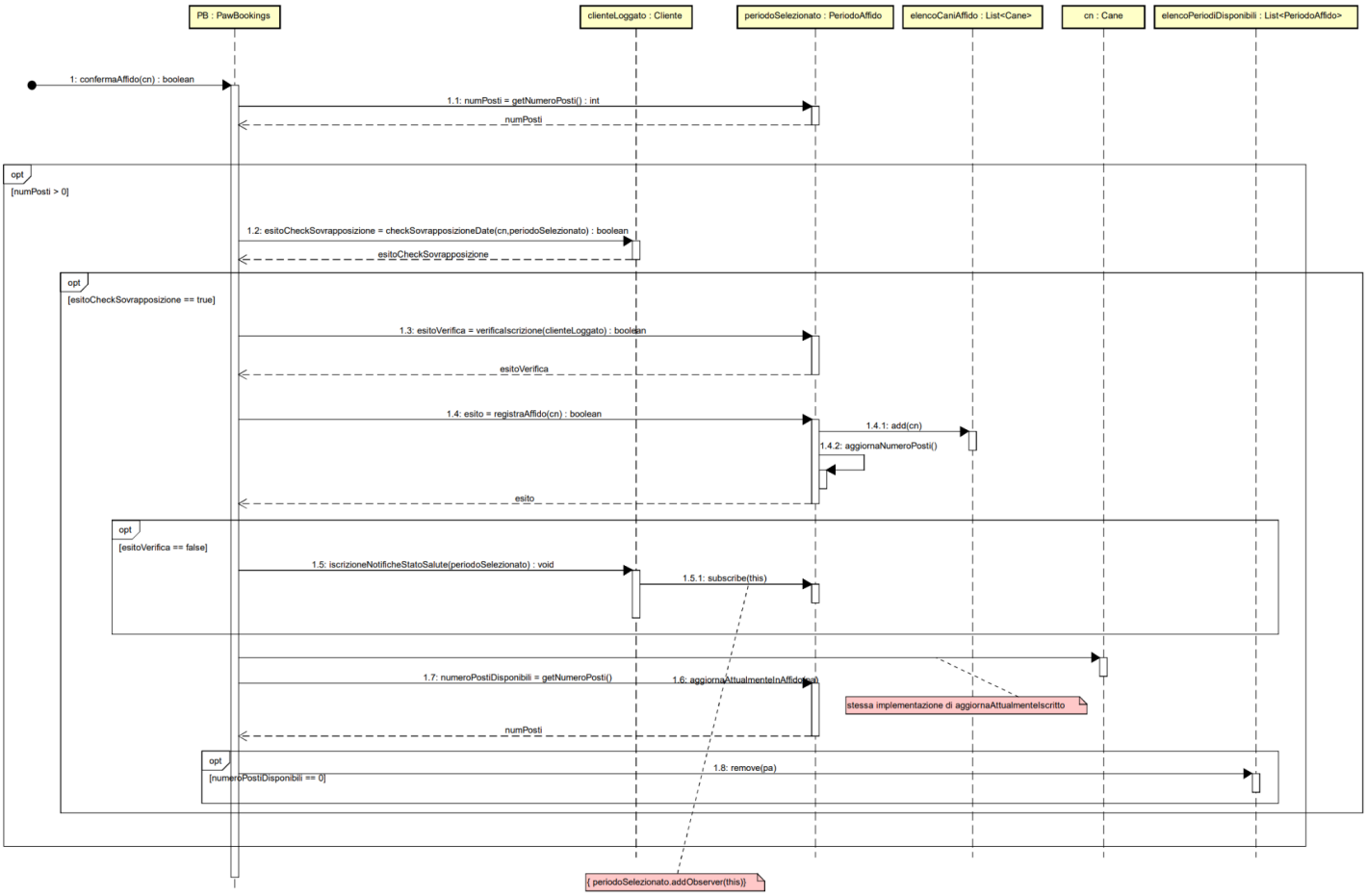
2.2.1.6. UC1 SD2: confermaIscrizioneCorso



2.2.1.7. UC2 SD2: selezionaTurno

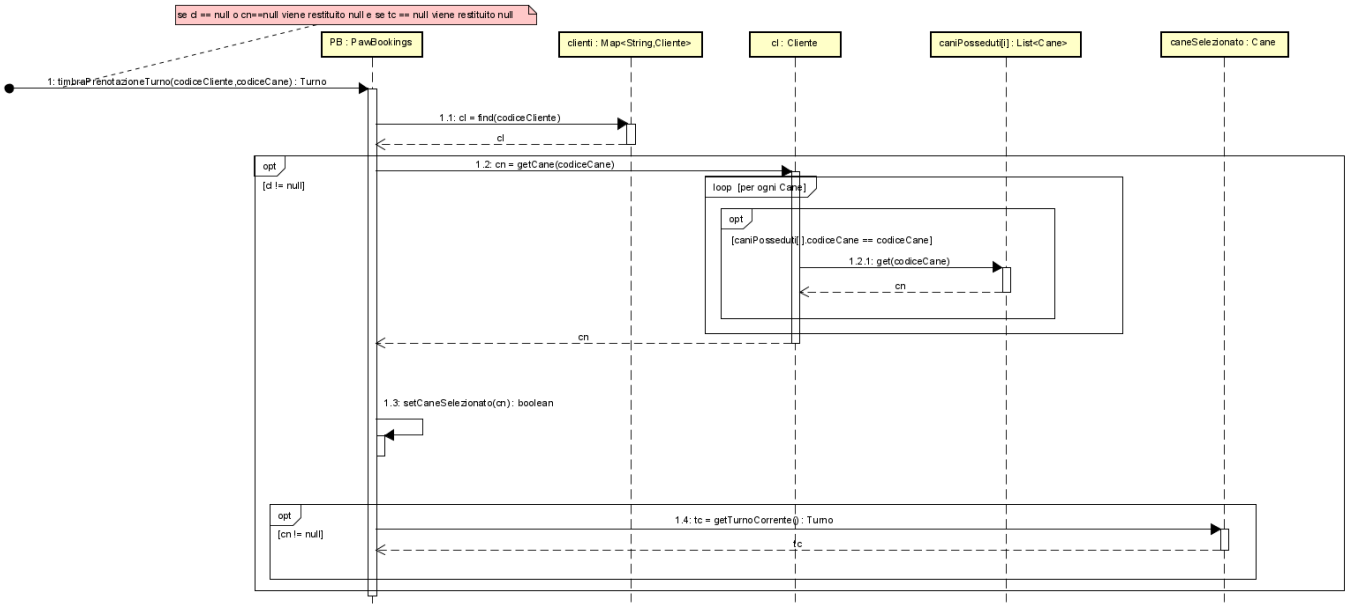


2.2.1.8. UC3 SD3: confermaAffido

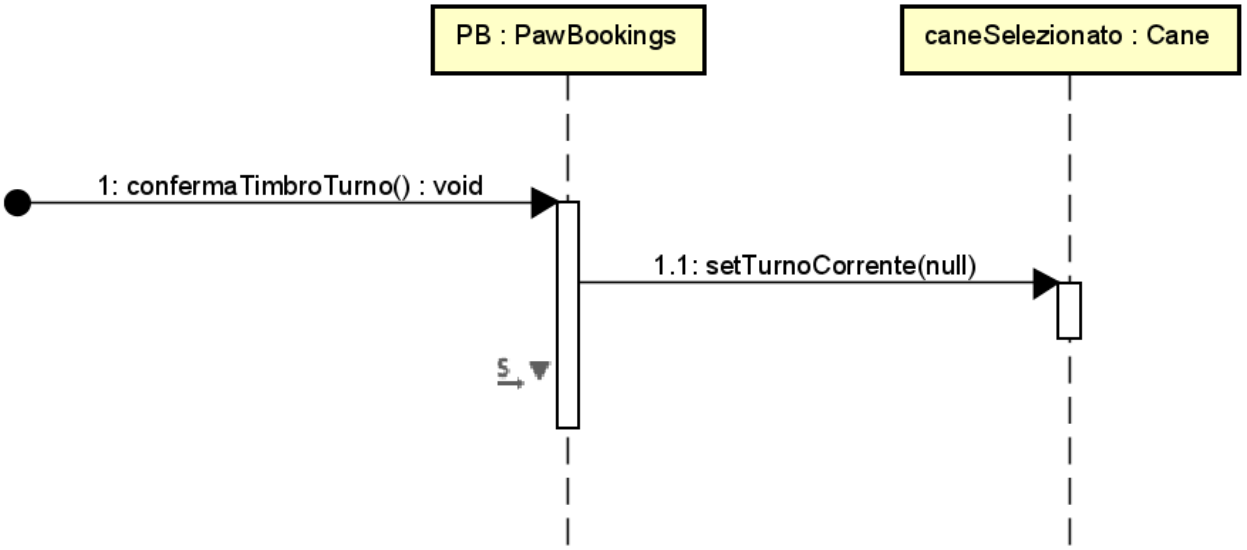


2.2.2. Nuovi SD

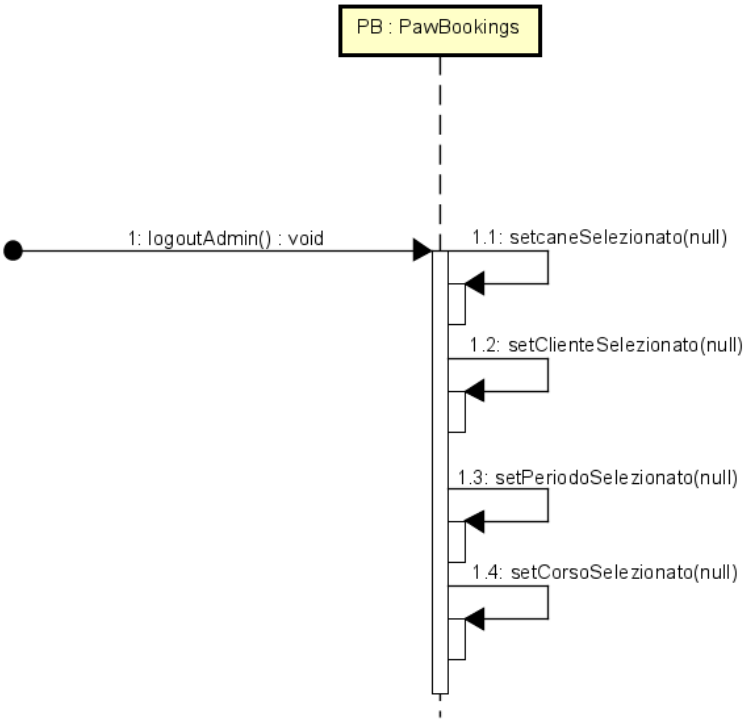
2.2.2.1. UC14 SD1: timbraPrenotazioneTurno



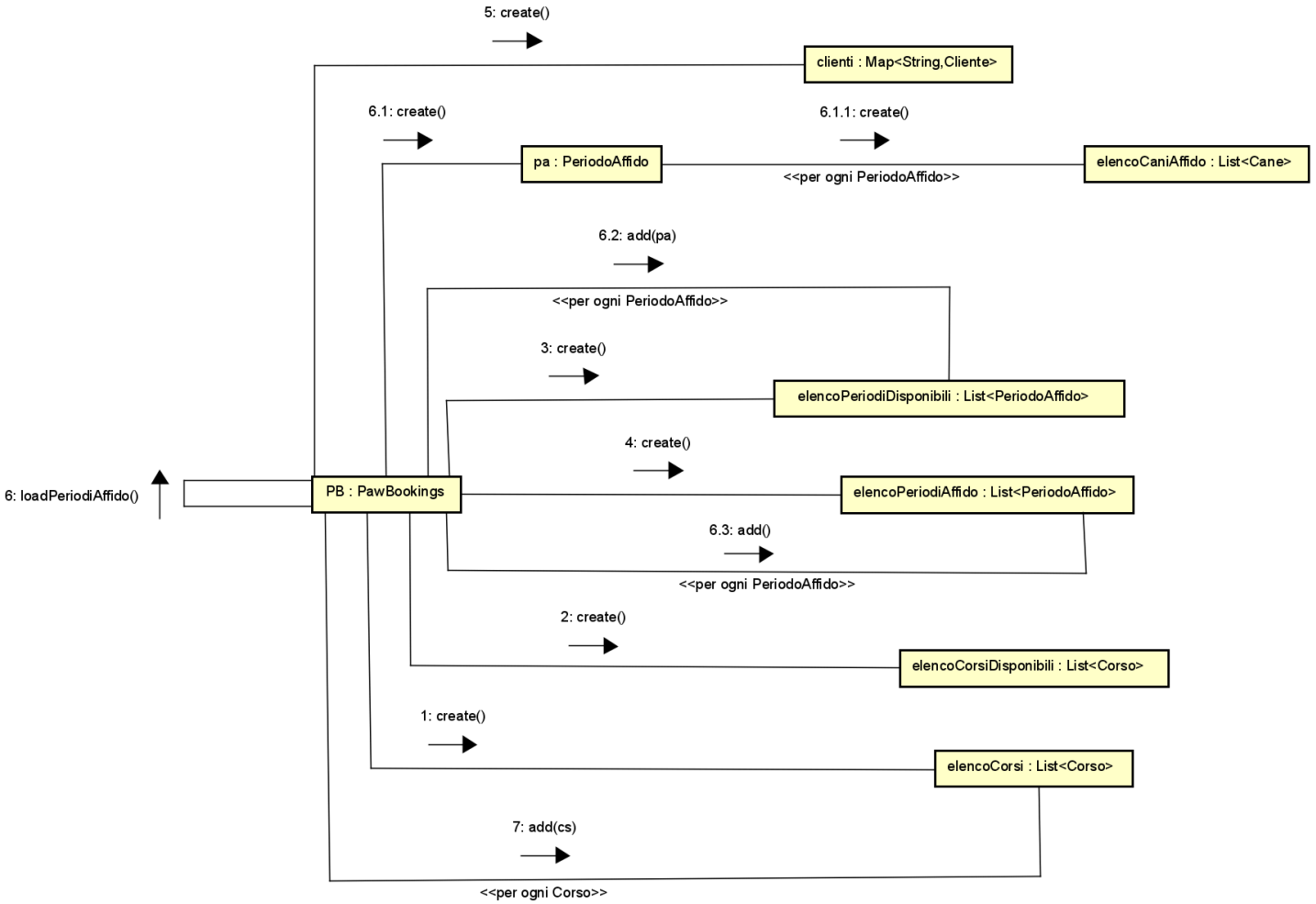
2.2.2.2. UC14 SD2: confermaTimbroTurno



2.2.2.3. logoutAdmin



2.2.3. Caso d'uso d'avviamento



2.3. Diagramma delle Classi di Progetto

Per la visione del DCD, si invita il lettore ad aprire il file "*iterazione4.asta*".

3. Testing

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 quarta iterazione, tenga conto del flusso degli scenari alternativi dei casi d'uso UC1-UC2-UC3-UC4 e del flusso dello scenario principale di successo dell' UC14. Per questa quarta iterazione, per quanto riguarda la classe Pawbookings, è stata aggiunta una nuova classe di test *PawBookingsTest5* che comprende i test unitari relativi agli scenari alternativi dei casi d'uso UC1-UC2-UC3-UC4 e del caso d'uso UC14.
Per quanto riguarda la classe Cane è stata aggiunta la classe di test CaneTest3 che comprende i test unitari relativi agli scenari alternativi dei casi d'uso UC1-UC2-UC3-UC4 e del caso d'uso UC14 e del metodo calcolaLezioneSuccessiva modificato nel corso di questa iterazione.

I test unitari implementati per le precedenti iterazioni sono stati adattati alle modifiche introdotte dalla nuova iterazione e sono funzionanti.
In particolar modo sono stati individuati i seguenti metodi delle relative classi:

- PawBookings
    - nuovaIscrizioneCorso

        - viene verificato che, selezionando un cane già iscritto ad un corso, l'invocazione del metodo sotto test restituisca null
        - viene verificato che, selezionando un cane che ha completato tutti i corsi messa a disposizione dal centro cinofilo, l'invocazione del metodo restituisca null
    - confermaIscrizioneCorso
        - viene verificato che, selezionando un cane che non è iscritto a nessun corso e che non ne ha mai seguito uno, l'invocazione del metodo sotto test restituisca *false* quando viene passato come parametro un'istanza di un corso avanzato
        - viene verificato che, selezionando un cane che non è iscritto a nessun corso, l'invocazione del metodo getAttualmenteIscritto restituisca *false*
    - prenotaTurnoLezione
        - viene verificato che, selezionando un cane che non è iscritto ad alcun corso e non ha completato alcun corso, l'invocazione del metodo sotto test restituisca null
        - viene verificato che, selezionando un cane che risulta essere iscritto ad un corso, l'invocazione del metodo sotto test restituisca un valore diverso da null
    - selezionaTurno
        - viene verificato che, selezionando una cane che risulta essere in affido nel periodo 1 la cui durata è sovrapposta ai turni della lezione 1 del corsoBase, l'invocazione del metodo sotto test restituisca *false*
    - confermaAffido
        - viene verificato che, selezionando un periodo che non è più disponibile, l'invocazione del metodo sotto test restituisca *false*
        - viene verificato che, selezionando un cane che è già prenotato al primo turno della prima lezione del corso base, l'invocazione del metodo sotto test restituisca *false*
    - delega
        - viene verificato che, effettuando l'accesso del cliente, l'invocazione del metodo getCodiceDelega restituisca 0
        - viene verificato che, effettuando l'accesso del cliente ed invocando il metodo sotto test, l'invocazione del metodo getCodiceDelega restituisca un valore diverso da 0
    - concludiAffidoDelega
        - viene verificato che, invocando il metodo sotto test e passando come parametri il codice delega errato e il codice cane corretto, venga restituito *null*
        - viene verificato che, invocando il metodo sotto test e passando come parametri il codice delega corretto e il codice cane errato, venga restituito *null*
        - viene verificato che, invocando il metodo sotto test e passando come parametri il codice delega errato e il codice cane errato, venga restituito *false*
        - viene verificato che, invocando il metodo sotto test e passando come parametri il codice delega e il codice cane entrambi corretti, il cane venga rimosso dal periodo di affido
    - mostraStatoAvanzamentoCorso
        - viene verificato che, selezionando un cane che non è iscritto ad alcun corso, l'invocazione del metodo sotto test restituisca *false*
    - accediComeAdmin
        - viene verificato che, invocando il metodo sotto test e passando come parametro il pin corretto, venga restituito *true*
        - viene verificato che, invocando il metodo sotto test e passando come parametro il pin errato, venga restituito *false*
    - concludiAffido

        - viene verificato che, invocando il metodo sotto test e passando come parametro un codice cliente errato, venga restituito *null*
        - viene verificato che, invocando il metodo sotto test e passando come parametro un codice cane errato, venga restituito *null*
        - viene verificato che, invocando il metodo sotto test e passando entrambi i parametri errati, venga restituito *null*
        - viene verificato che, invocando il metodo sotto test e passando entrambi i parametri corretti, venga restituito *null*
    - timbraPrenotazioneTurno
        - viene testato che, invocando il metodo con parametri che sappiamo essere entrambi errati, questo restituisca null
        - viene testato che, invocando il metodo con il primo parametro errato ed il secondo corretto, questo restituisca null
        - viene testato che, invocando il metodo con il primo corretto ed il secondo errato questo restituisca ancora null
        - viene verificato che, invocando il metodo con i due parametri corretti (cioè il codice cliente appare nell'elenco dei clienti ed il codice cane appare nell'elenco dei cani posseduti del cliente), ma con il codice cane appartenente ad un cane che non ha effettuato alcuna prenotazione di alcun turno, ci aspettiamo comunque null
        - viene verificato che, invocando il metodo con entrambi i parametri giusti e con un codice cane di un cane che ha effettuato una prenotazione relativa ad un turno, questo restituisca l'istanza di turno al quale il cane è prenotato
    - confermaTimbroTurno
        - viene verificato che il getTurnoCorrente() invocato sul cane selezionato restituisca null a seguito dell'invocazione del metodo (dopo esserci accertati che prima dell'invocazione il metodo restituisce, invece, un'istanza di turno).
- Cliente
    - checkSovrapposizioneDate
        - viene verificato che,impostando il periodo di affido e settando il turno corrente del cane a null, l'invocazione del metodo sotto test restituisca *true*
        - viene verificato che, impostando il turno tale che sia sovrapposto con la dataInizio del periodo di affido, l'invocazione del metodo sotto test restituisca *false*
        - viene verificato che, impostando il turno tale che sia sovrapposto con la dataFine del periodo di affido, l'invocazione del metodo sotto test restituisca *false*
        - viene verificato che, impostando il turno tale che sia sovrapposto tra dataInizio e dataFine del periodo di affido, l'invocazione del metodo sotto test restituisca *false*
        - viene verificato che, impostando il turno tale che non sia sovrapposto al periodo di affido, l'invocazione del metodo sotto test restituisca *true*
- Cane
    - checkNuovaIscrizioneCorso
        - viene verificato che, selezionando un cane attualmente iscritto ad un corso, l'invocazione del metodo sotto test restituisca *false*
        - viene verificato che, selezionando un cane che ha completato tutti i corsi, l'invocazione del metodo sotto test restituisca *false*
    - checkConfermaIscrizioneCorso
        - viene verificato che, selezionando una cane che non ha seguito alcun corso, invocando il metodo sotto test e passando come parametro corsoAvanzato venga restituito *false*

        - viene verificato che, selezionando una cane che non ha seguito alcun corso, invocando il metodo sotto test e passando come parametro corsoBase venga restituito *true*
        - viene verificato che, selezionando un cane che ha completato il corsoBase, invocando il metodo sotto test e passando come parametro corsoBase venga restituito *false*
        - viene verificato che, selezionando un cane che ha completato il corsoBase, invocando il metodo sotto test e passando come parametro corsoAgility venga restituito *false*
        - viene verificato che, selezionando una cane che ha completato il corsoBase, invocando il metodo sotto test e passando come parametro corsoAvanzato venga restituito *true*
    - checkSovrapposizioneDate
        - viene verificato che, settando l'affido corrente del cane ad un periodo di affido sovrapposto al turno, l'invocazione del metodo restituisca *false*
        - viene verificato che, settando l'affido corrente del cane ad un periodo di affido non sovrapposto al turno, l'invocazione del metodo restituisce *true*
    - calcolaLezioneSuccessiva
        - viene verificato che, selezionando un cane iscritto al corsoBase che non ha ancora seguito alcuna lezione, l'invocazione del metodo sotto test restituisca la prima lezione del corsoBase
        - viene verificato che, aggiungendo la prima lezione del corsoBase alle lezioni seguite dal Cane, l'invocazione del metodo sotto test restituisca la seconda lezione del corsoBase
        - viene verificato che, aggiungendo l'ultima lezione del corsoBase alle lezioni seguite dal Cane ed iscrivendo il Cane al corsoAvanzato, l'invocazione del metodo sotto test restituisca la prima lezione del corsoAvanzato
        - viene verificato che, aggiungendo la prima lezione del corsoAvanzato alle lezioni seguite dal Cane, l'invocazione del metodo sotto test restituisca la seconda lezione del corsoAvanzato