PawBookings: Elaborazione - Iterazione 3

Giuseppe Leocata, Alberto Provenzano, Daniele Lucifora

Introduzione

Per l'iterazione 3, sono stati scelti i seguenti requisiti:

- Implementazione del caso d'uso *UC7: Gestisci Corso.*
- Implementazione del caso d'uso *UC8: Scambio Turno Corso.*
- Implementazione del caso d'uso *UC9: Notifica sullo stato di salute del Cane in Affido.*
- Implementazione del caso d'uso *UC10: Visualizza programma di un Corso.*
- Implementazione del caso d'uso *UC11: Mostra avanzamento stato Corso.*
- implementazione del caso d'uso UC12: *Notifica sullo stato di salute del cane in affido*
- implementazione del caso d'uso UC13: *Visualizza stato di salute*
- 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 riguarda tutti i casi d'uso rientranti nei requisiti di quest'iterazione, sono stati sviluppati i passi dei rispettivi scenari di successo (ed in alcuni anche gli scenari alternativi) al fine di facilitare l'analisi in termini di sviluppo dei diagrammi di sequenza di sistema. E' stato, inoltre, aggiornato il Glossario.

1. Analisi Orientata agli Oggetti

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

1.1. Modello di Dominio

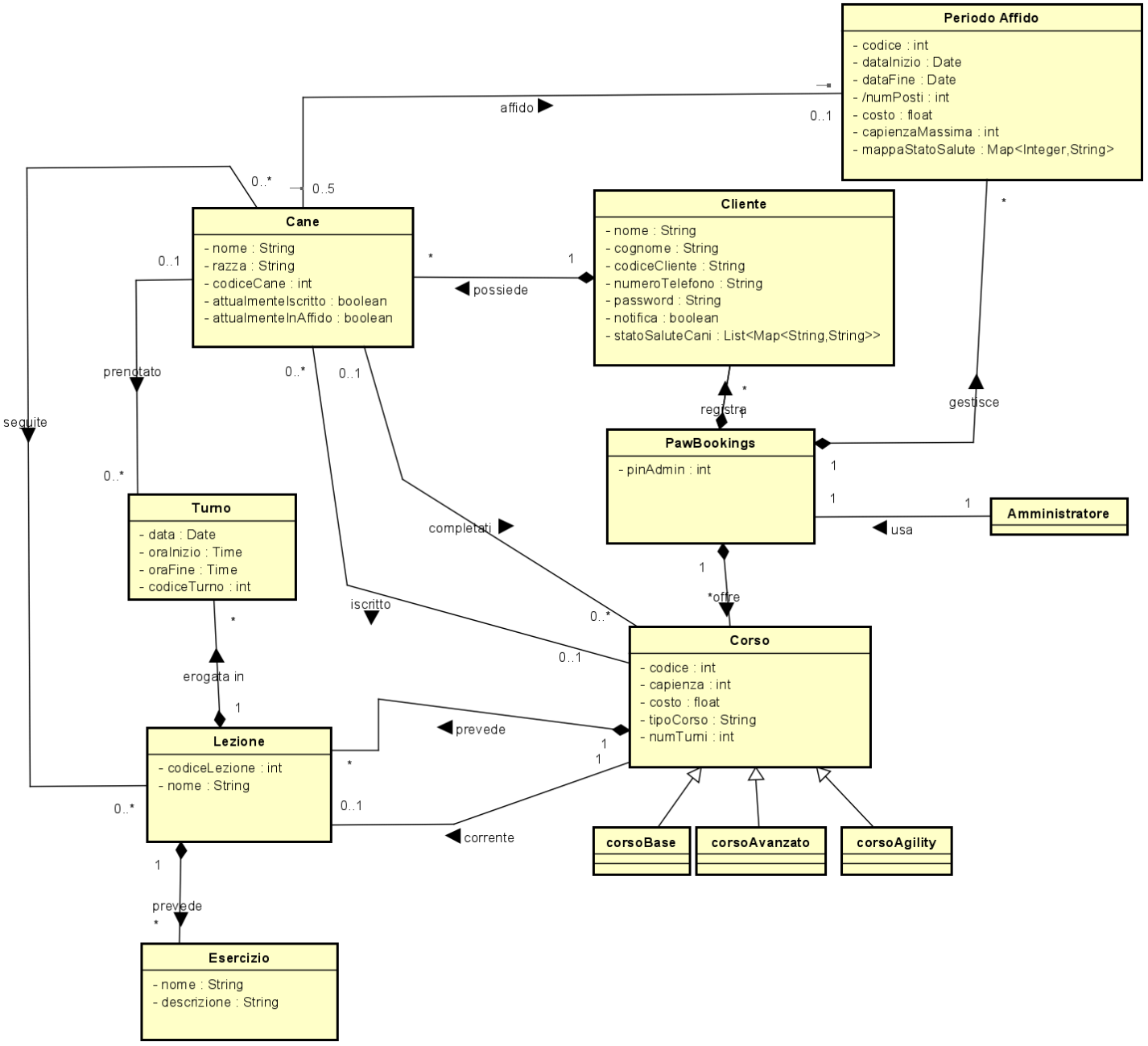
Dopo un'attenta analisi, nasce l'esigenza di inserire le seguenti classi concettuali:

- **Esercizio**

Sono stati, inoltre, aggiunti i seguenti attributi:

- "*codice*" alla classe concettuale **Turno**.
- "*notifica*" alla classe **Cliente**.
- "numTurni" alla classe **Corso**.

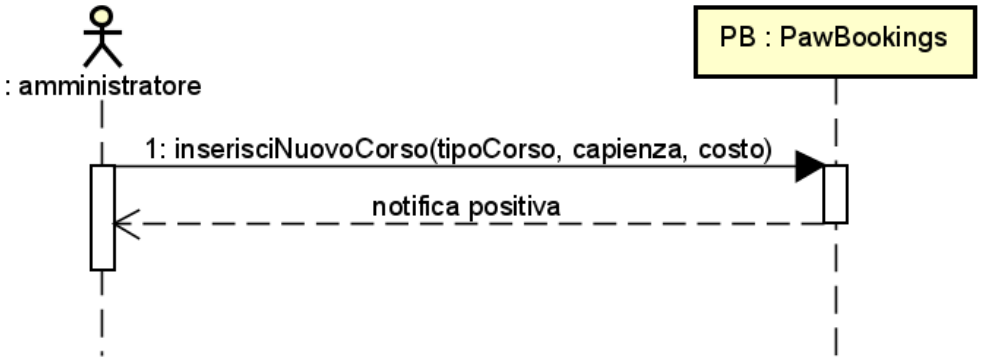
Dall'integrazione di queste nuove classi a quelle già esistenti, tenendo conto di associazioni e attributi, è stato ricavato il seguente Modello di Dominio:



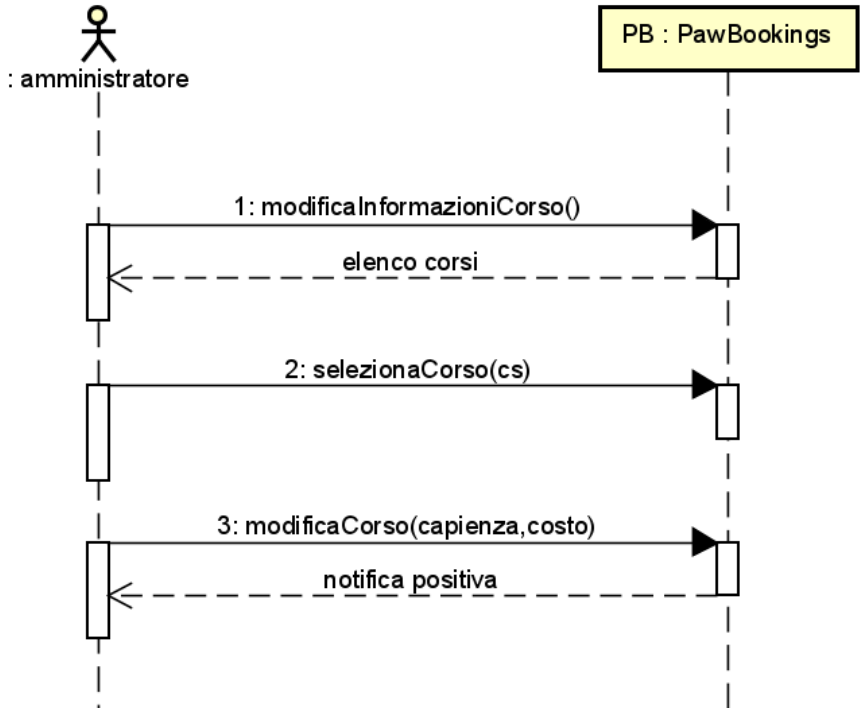
1.2. Diagrammi di Sequenza di Sistema (SSD)

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

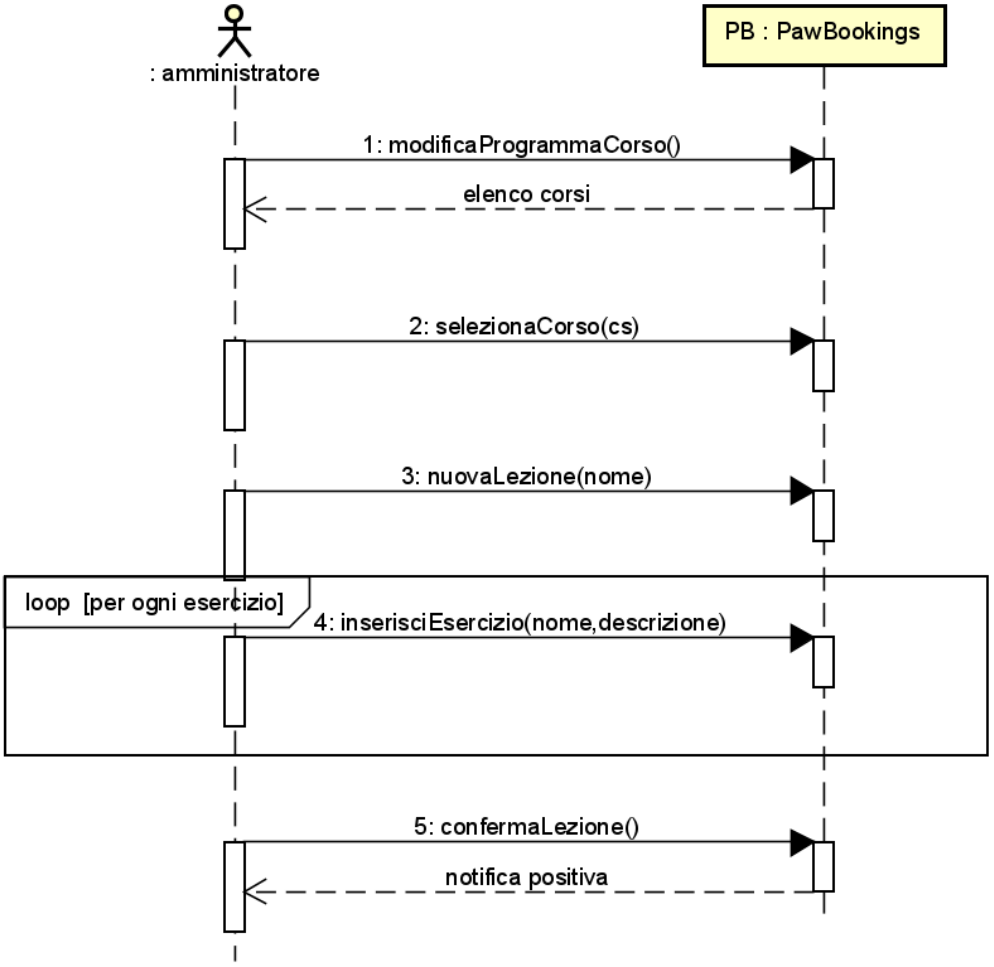
1.2.1. SSD UC7



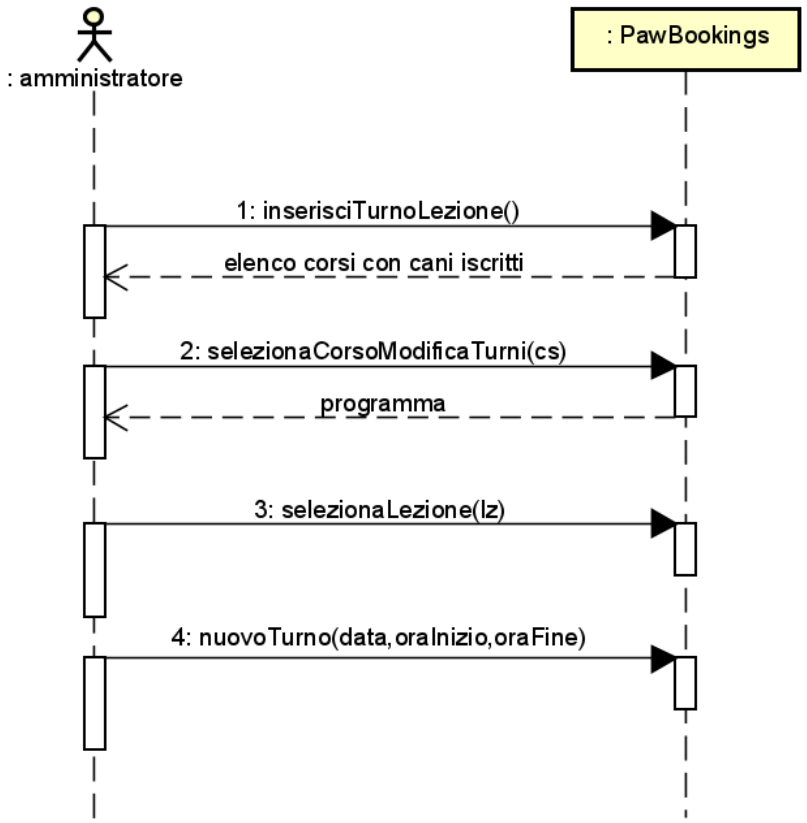
1.2.2. SSD UC7 - Estensione 1



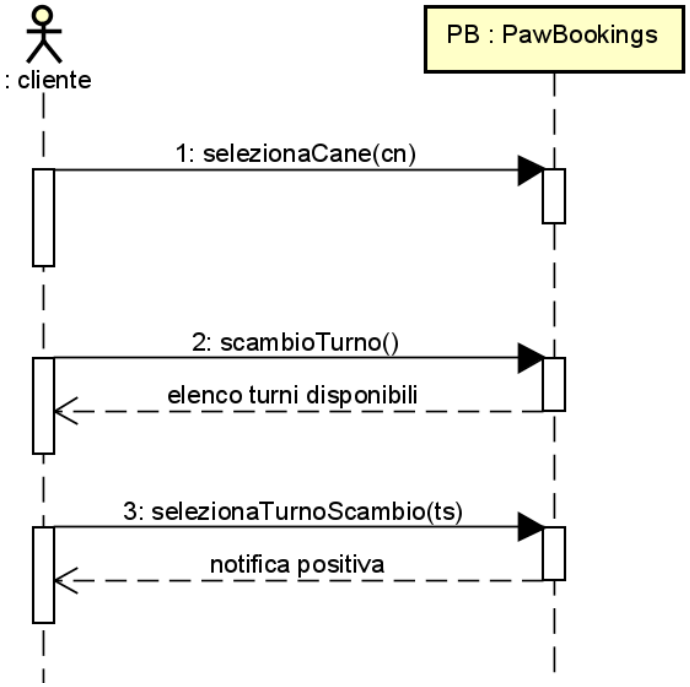
1.2.3. SSD UC7 - Estensione 2



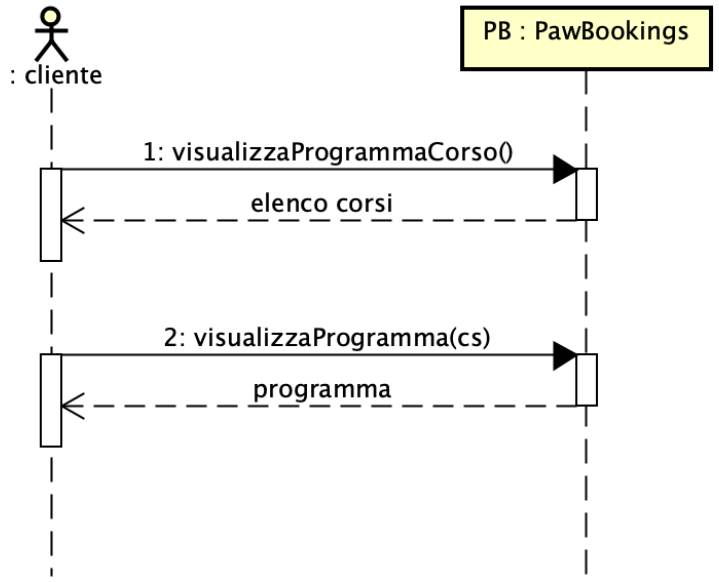
1.2.4. SSD UC8



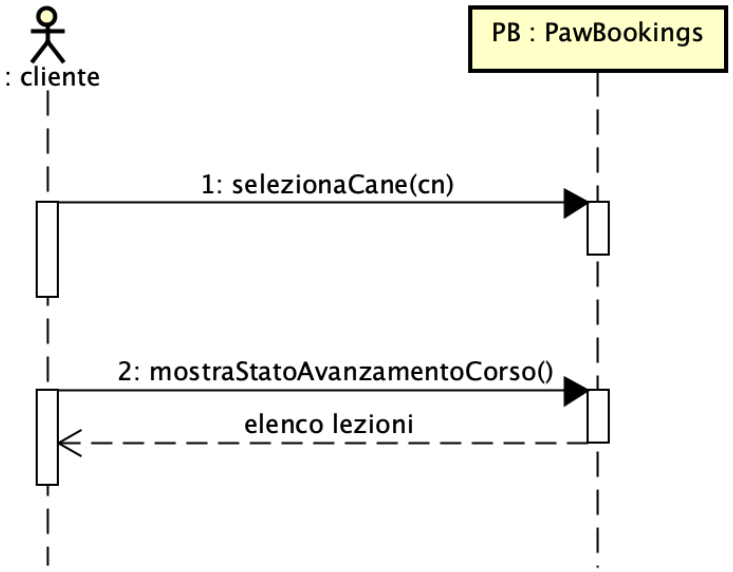
1.2.5. SSD UC9



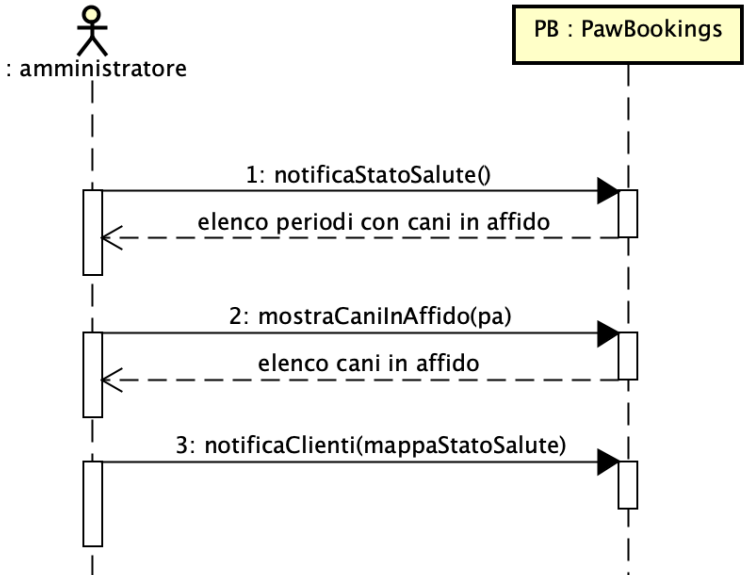
1.2.6. SSD UC10



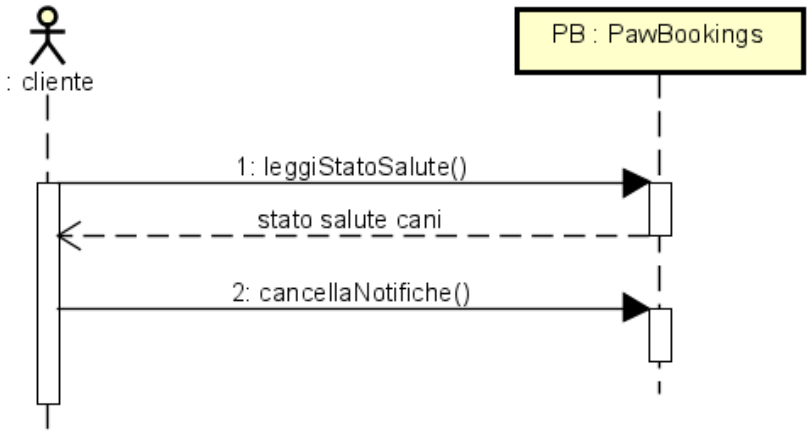
1.2.7. SSD UC11



1.2.8. SSD UC12



1.2.9. SSD UC13



1.3. Contratti delle Operazioni

1.3.1. C01: inserisciNuovoCorso

operazione:	inserisciNuovoCorso(tipoCorso: String, capienza: int, costo: float)
riferimenti:	Caso d'uso UC7: Gestisci Corso
pre-condizioni:	-
post-condizioni:	- è stata creata l'istanza c di Corso - gli attributi di c sono stati inizializzati - *c* è stata associata a *PawBookings* tramite l'associazione "offre"

1.3.2. C02: modificaInformazioniCorso

operazione:	modificaCorso(capienza: int, costo: float)
riferimenti:	Caso d'uso UC7: Gestisci Corso
pre-condizioni:	- è stata recuperata l'istanza cs della classe Corso
post-condizioni:	- gli attributi di cs sono stati aggiornati

1.3.3. C03: nuovaLezione

operazione:	nuovaLezione(nome: String)
riferimenti:	Caso d'uso UC7: Gestisci Corso
pre-condizioni:	- è stata recuperata l'istanza cs della classe Corso
post-condizioni:	- è stata creata l'istanza lz della classe Lezione - gli attributi di lz sono stati inizializzati - lz è stata associata a cs tramite l'associazione "prevede"

1.3.4. C04: inserisciEsercizio

operazione:	inserisciEsercizio(nome: String, descrizione: String)
riferimenti:	Caso d'uso UC7: Gestisci Corso
pre-condizioni:	- è stata recuperata l'istanza lz della classe Lezione
post-condizioni:	- è stata creata l'istanza es della classe Esercizio - gli es sono stati inizializzati - es è stata associata a lz tramite l'associazione "prevede"

1.3.5. C05: nuovoTurno

operazione:	nuovoTurno(data: LocalDate, oraInizio: LocalTime, oraFine: LocalTime)
riferimenti:	Caso d'uso UC8: Inserimento Turno Lezione
pre-condizioni:	- è stata recuperata l'istanza lz della classe Lezione
post-condizioni:	- è stata creata l'istanza t della classe Turno - gli attributi di t sono stati inizializzati - t è stata associata all'istanza - lz è stata associata a t tramite l'associazione "erogata in"

1.3.6. C06: selezionaTurnoScambio

operazione:	selezionaTurnoScambio(ts: Turno)
riferimenti:	Caso d'uso UC9: Scambio Turno Lezione
pre-condizioni:	- è stata recuperata l'istanza cn di Cane - è stata recuperato un Turno tc dall'associazione "prenotato" con cn
post-condizioni:	- tc è stato dissociata da cn - ts è stata associata a cn tramite l'associazione "prenotato"

1.3.7. C07: notificaClienti

operazione:	notificaClienti(mappaStatoSalute: Map<Integer, String>)
riferimenti:	Caso d'uso UC12: Notifica sullo stato di salute del cane in affido
pre-condizioni:	- è stata recuperata l'istanza pa di PeriodoAffido
post-condizioni:	- l'attributo *mappaStatoSalute* è stato aggiornato

2. Progettazione Orientata agli Oggetti

Si prosegue adesso, in linea con quanto fatto nelle precedenti iterazioni, con lo sviluppo dei Diagrammi di Sequenza e del Diagramma delle Classi di Progetto. Da segnalare, inoltre, la rimozione della gerarchia di classi relativa al corso, non più necessaria per via della rimozione del caso d'uso d'avviamento relativo al precaricamento di corsi.

2.1. Pattern applicati

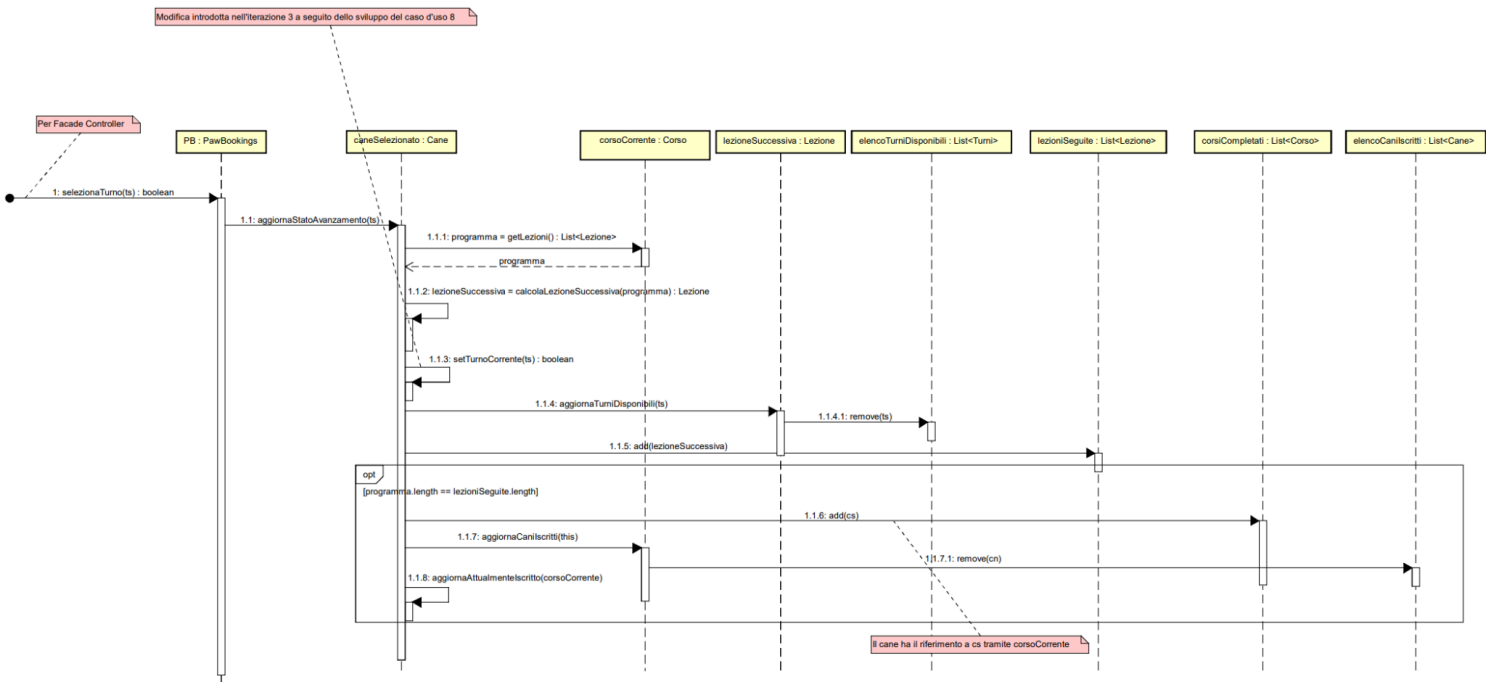
In questa iterazione si è ritenuto necessario ricorrere al pattern GoF Observer in quanto risulta essere il modo più appropriato per gestire la notifica dello stato di salute dei cani in affido ai rispettivi clienti.

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

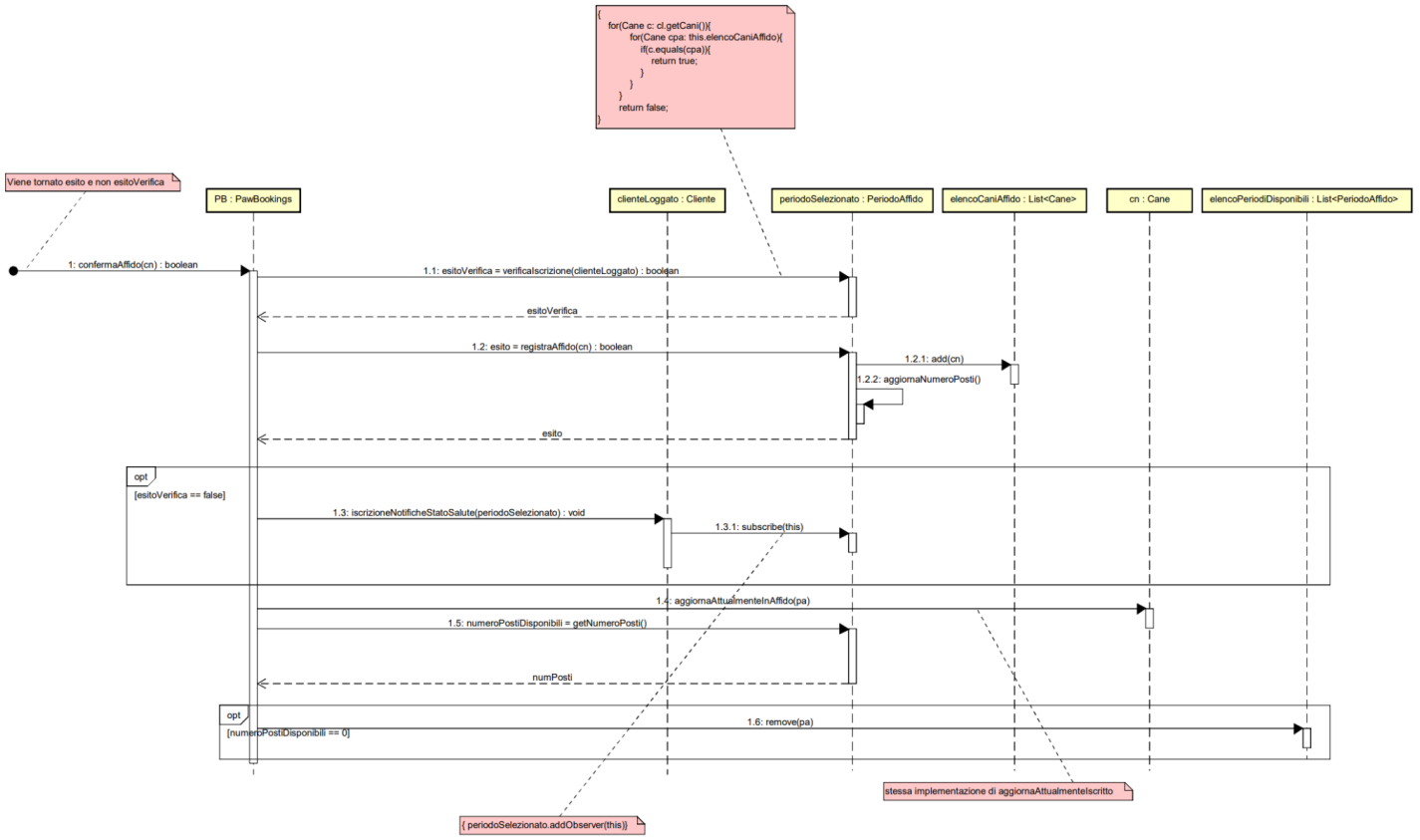
2.2. Diagrammi di Sequenza (SD)

Oltre ad aver prodotto i diagrammi che verranno elencati di seguito, si segnalano diverse modifiche ai seguenti diagrammi: *UC2 SD2: Seleziona Turno*, *UC3 SD3: Conferma Affido*, *UC4 SD1: Concludi Affido*, *UC4 SD2: Conferma Conclusione Affido*, *UC5 SD2: registrati*

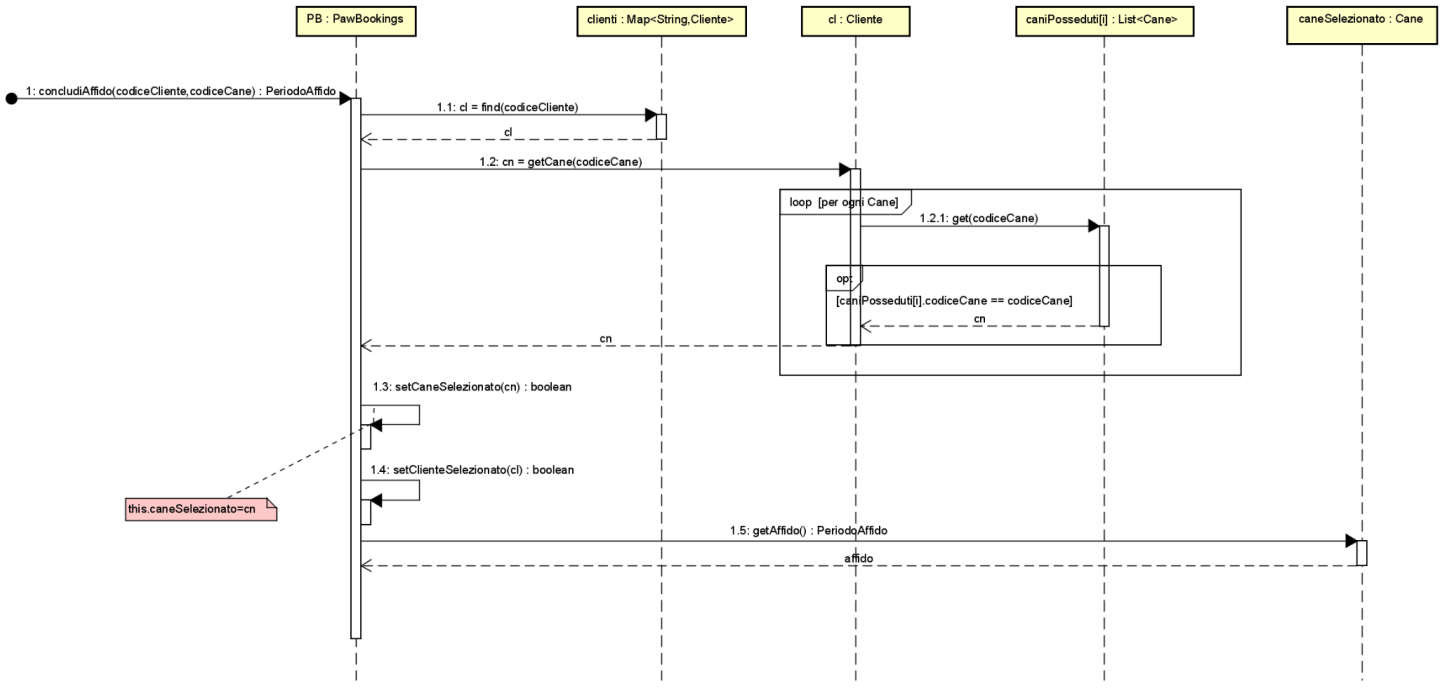
2.2.1. UC2 SD2



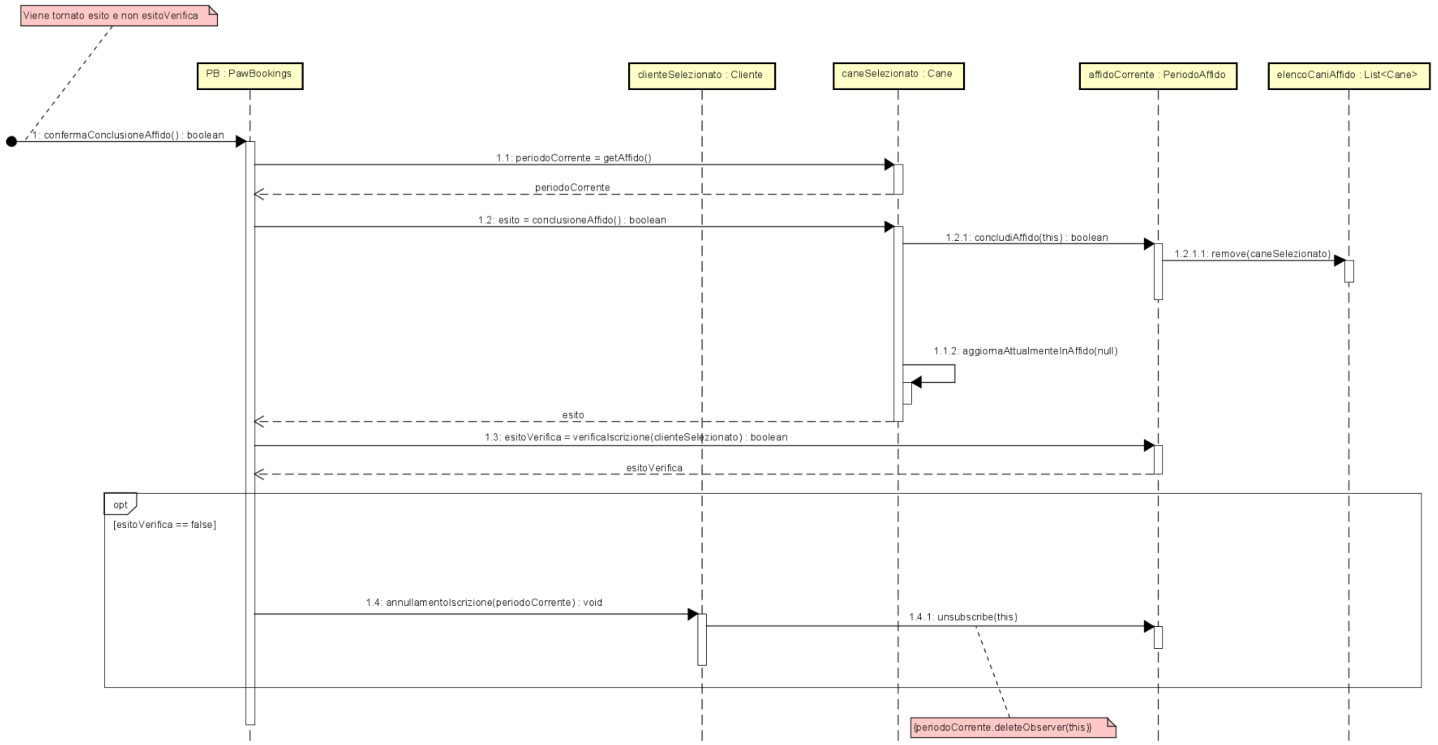
2.2.2. UC3 SD3



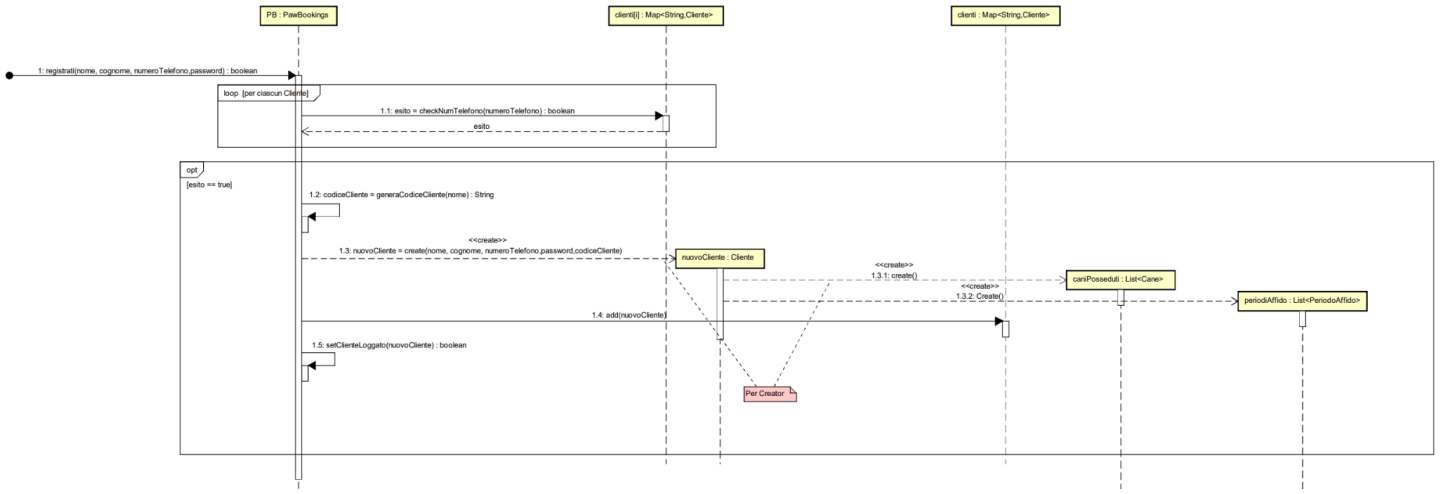
2.2.3. UC4 SD1



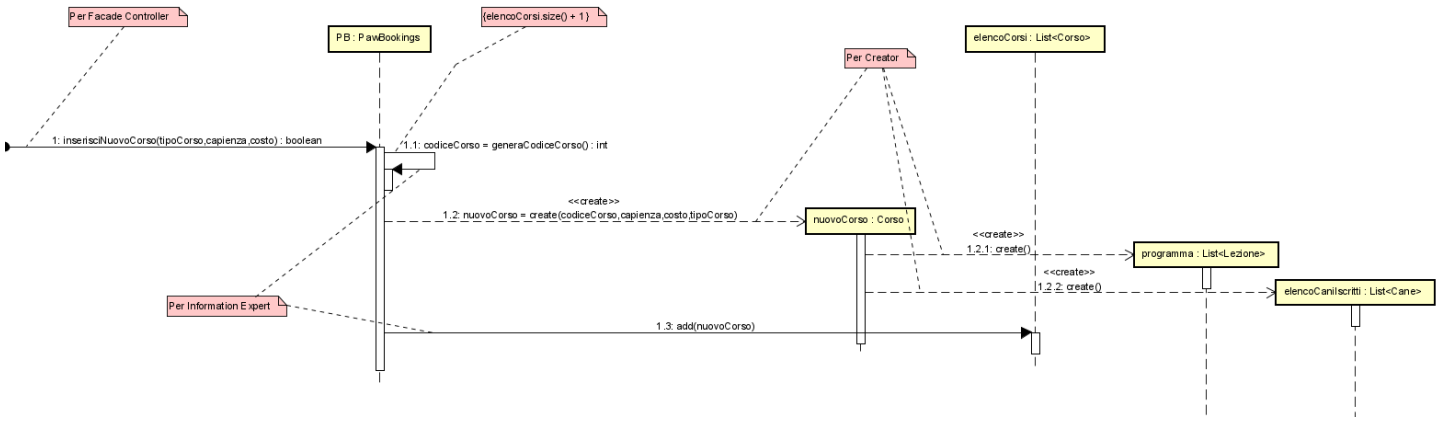
2.2.4. UC4 SD2



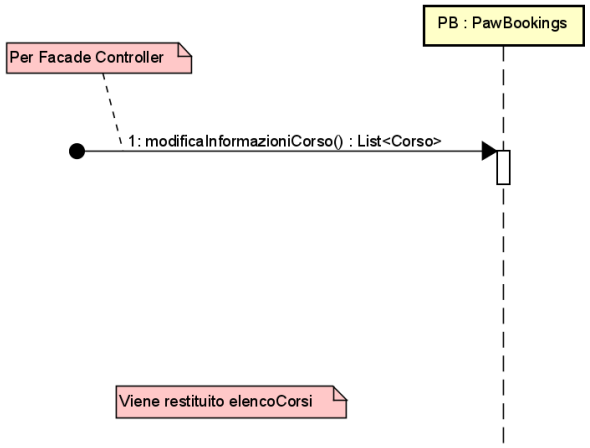
2.2.5. UC5 SD2



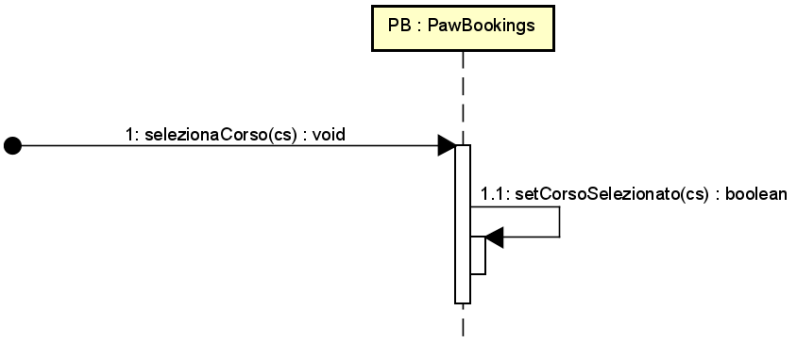
2.2.6. UC7 SD1



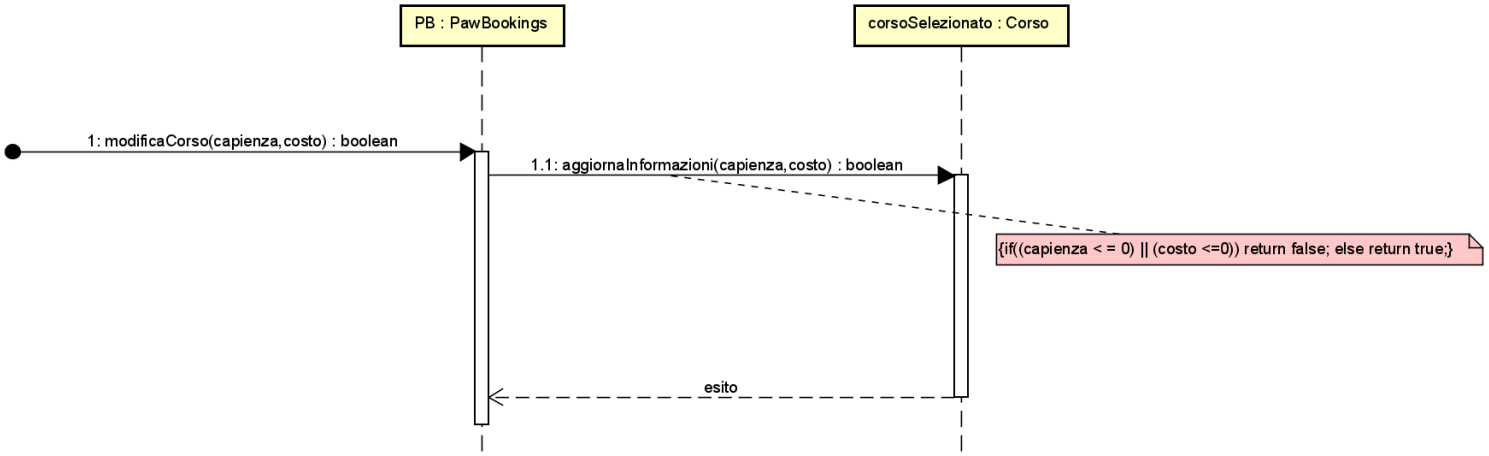
2.2.7. UC7 SD2



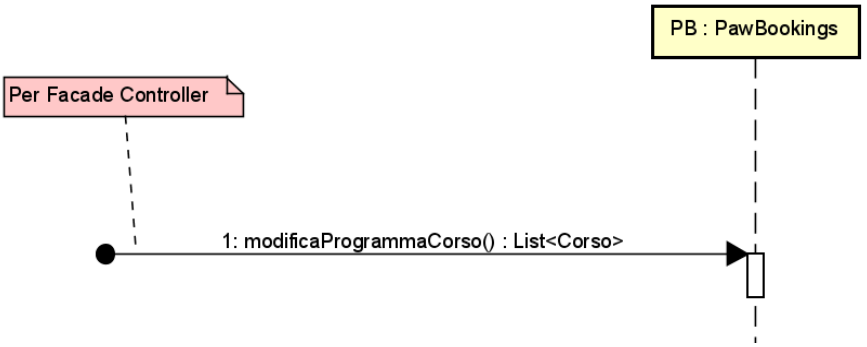
2.2.8. UC7 SD3



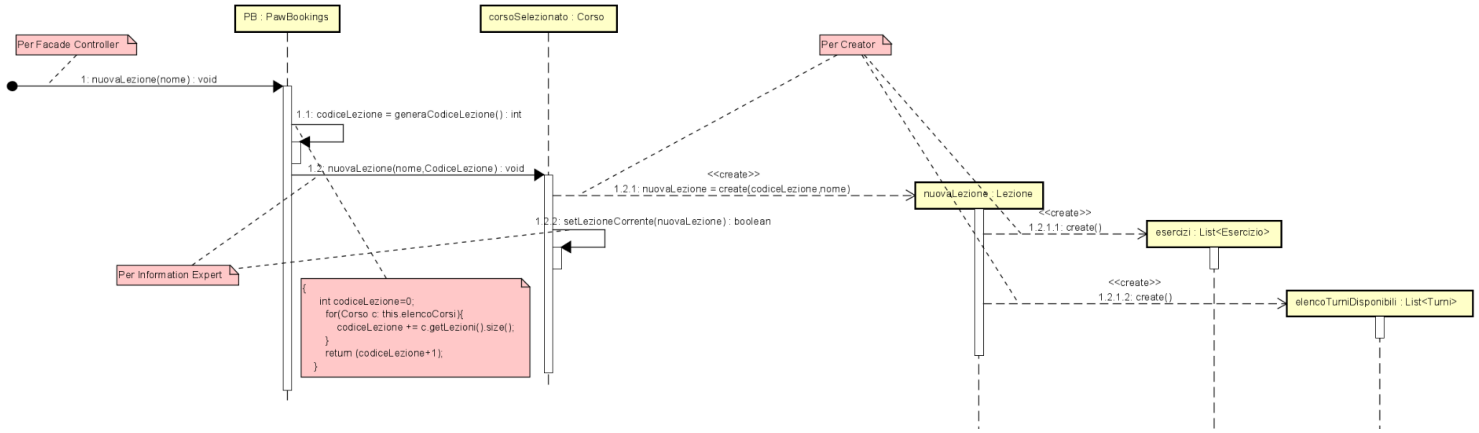
2.2.9. UC7 SD4



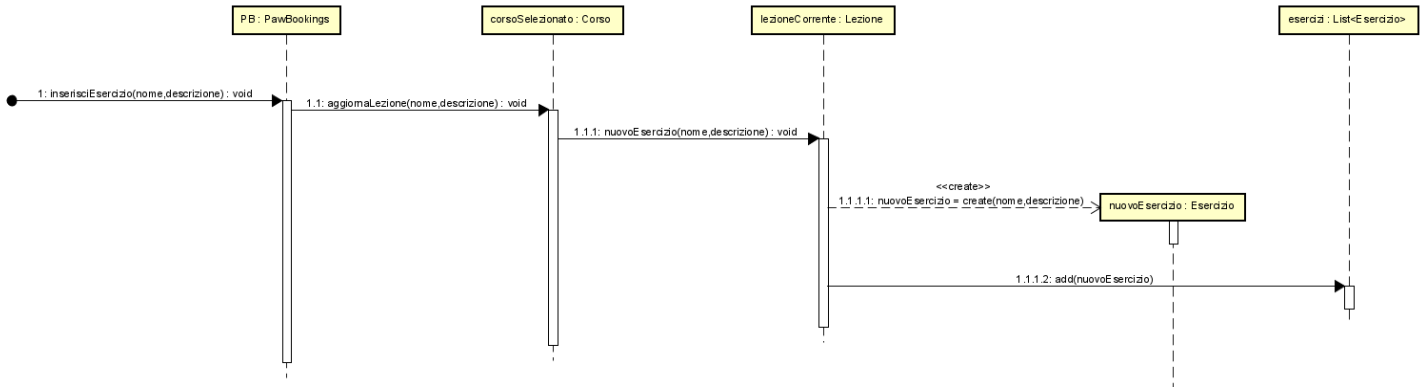
2.2.10. UC7 SD5



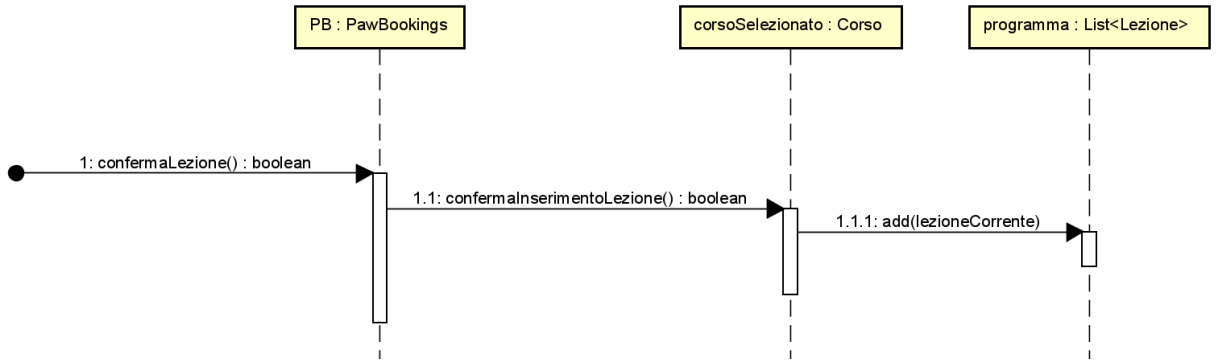
2.2.11. UC7 SD6



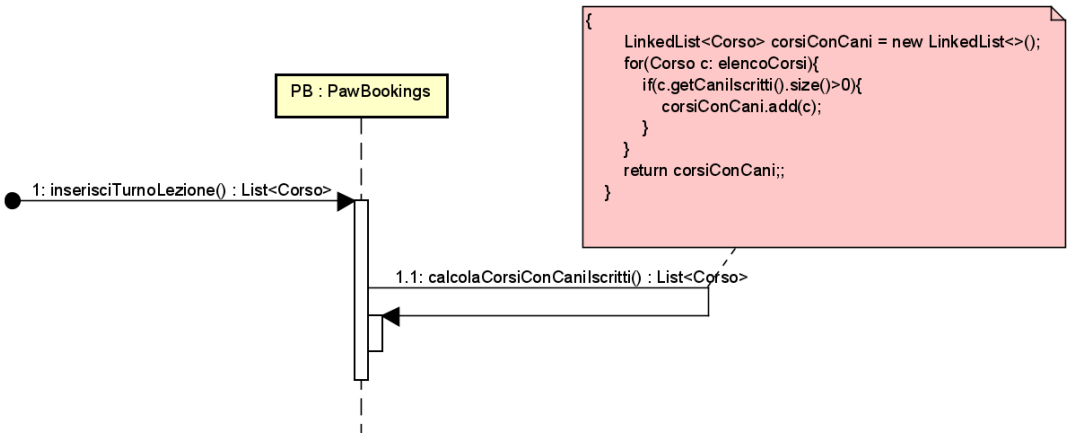
2.2.12. UC7 SD8



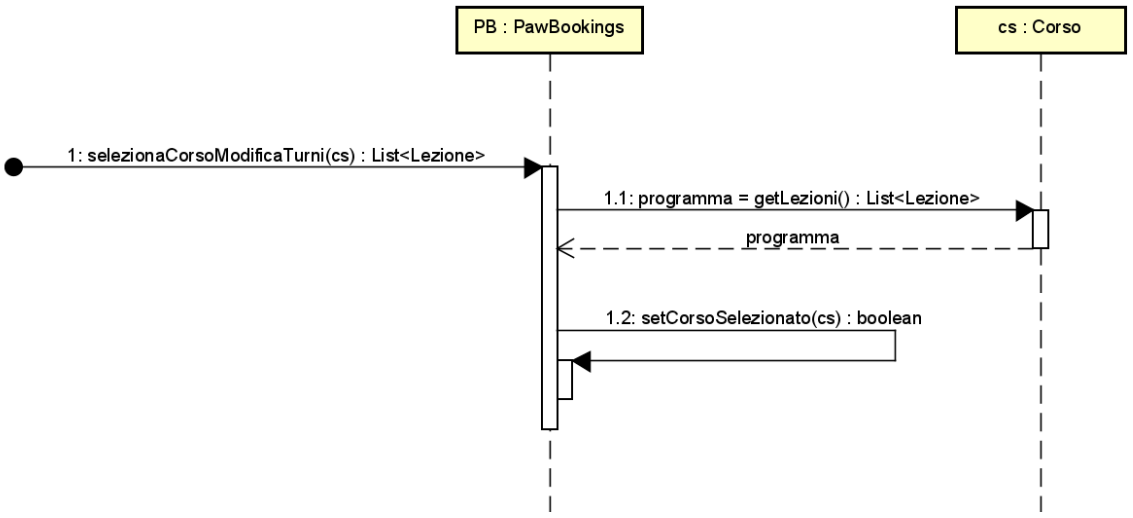
2.2.13. UC7 SD9



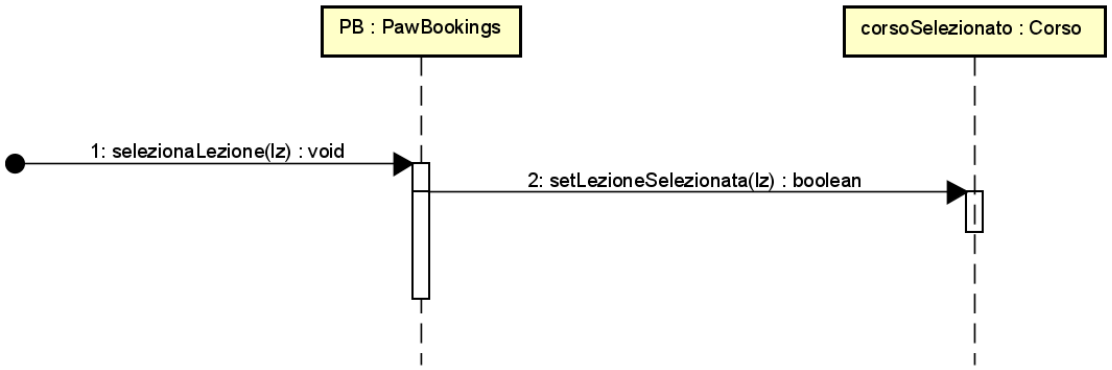
2.2.14. UC8 SD1



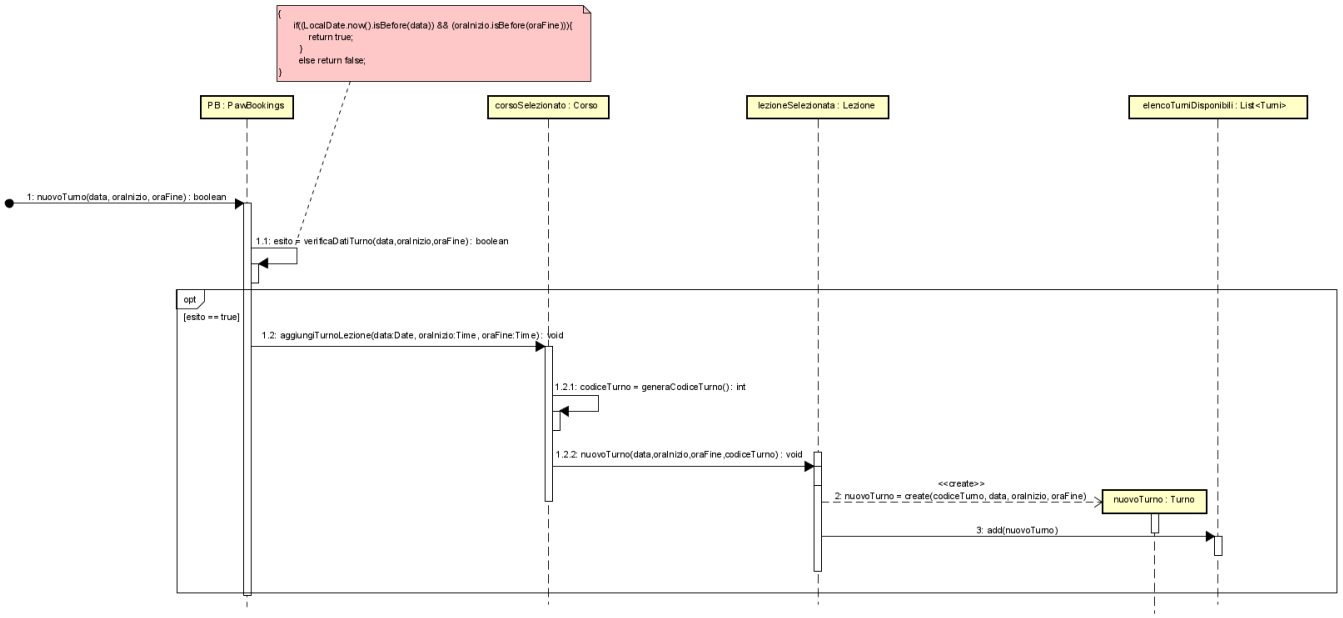
2.2.15. UC8 SD2



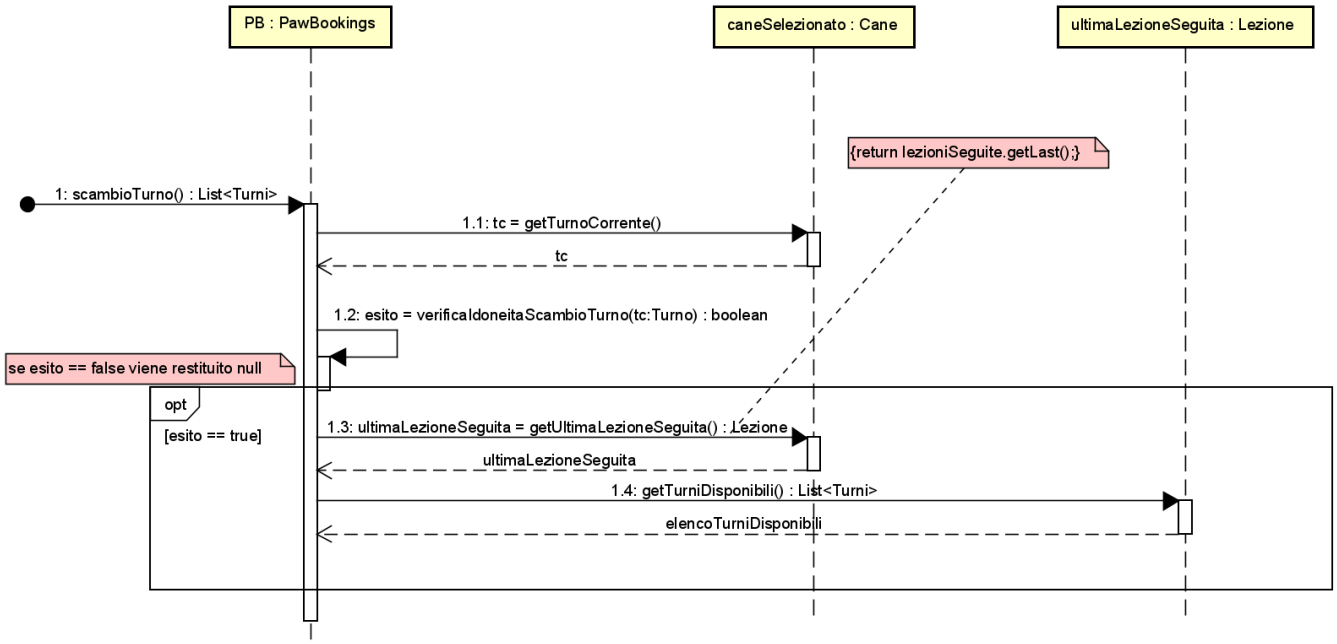
2.2.16. UC8 SD3



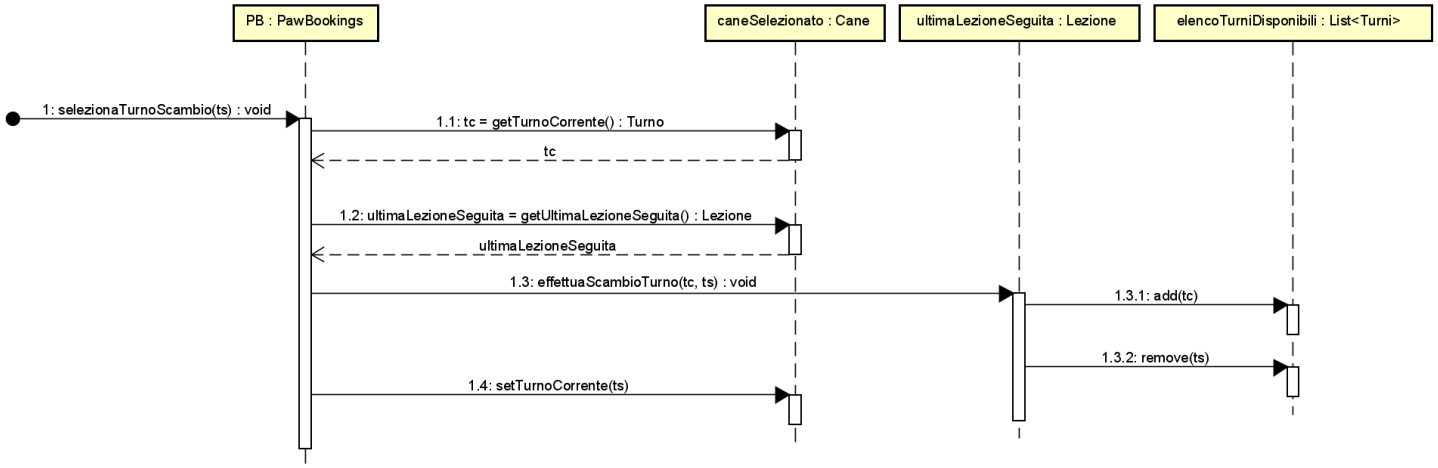
2.2.17. UC8 SD4



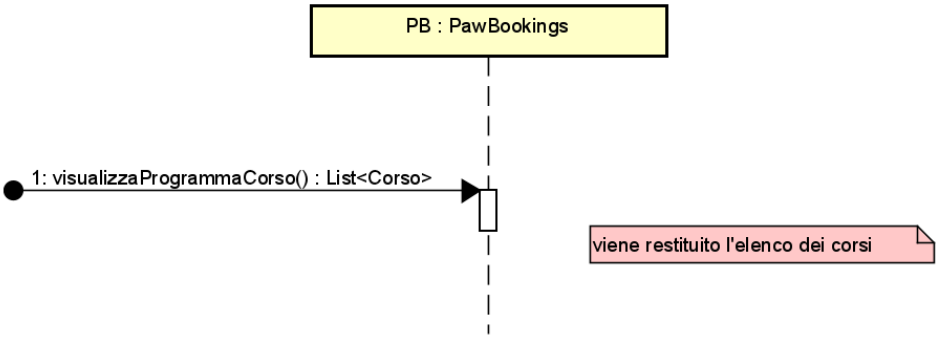
2.2.18. UC9 SD1



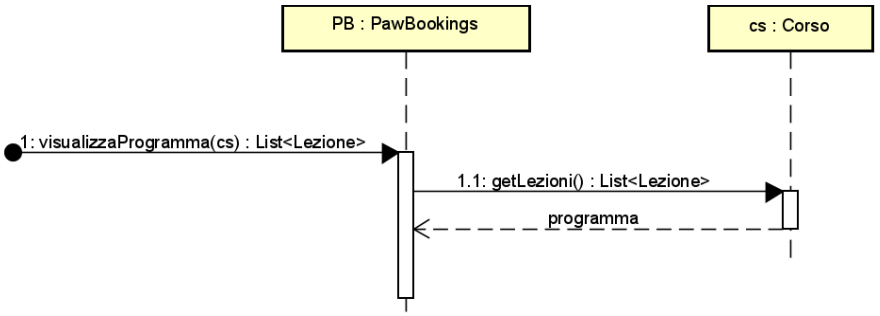
2.2.19. UC9 SD2



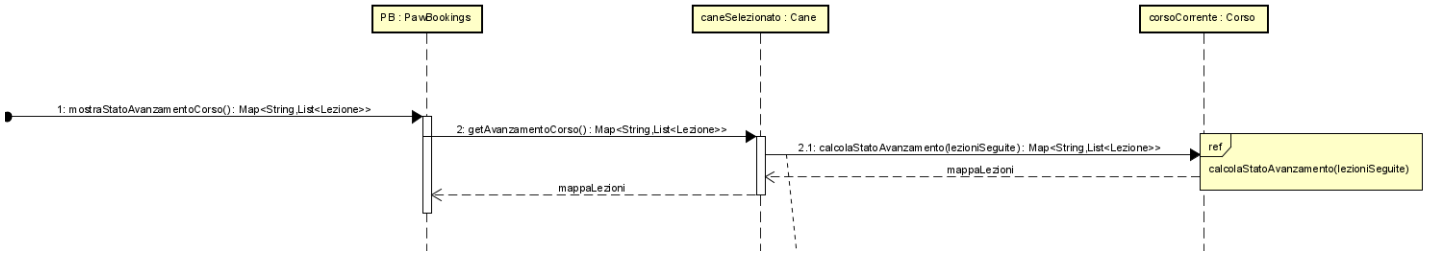
2.2.20. UC10 SD1



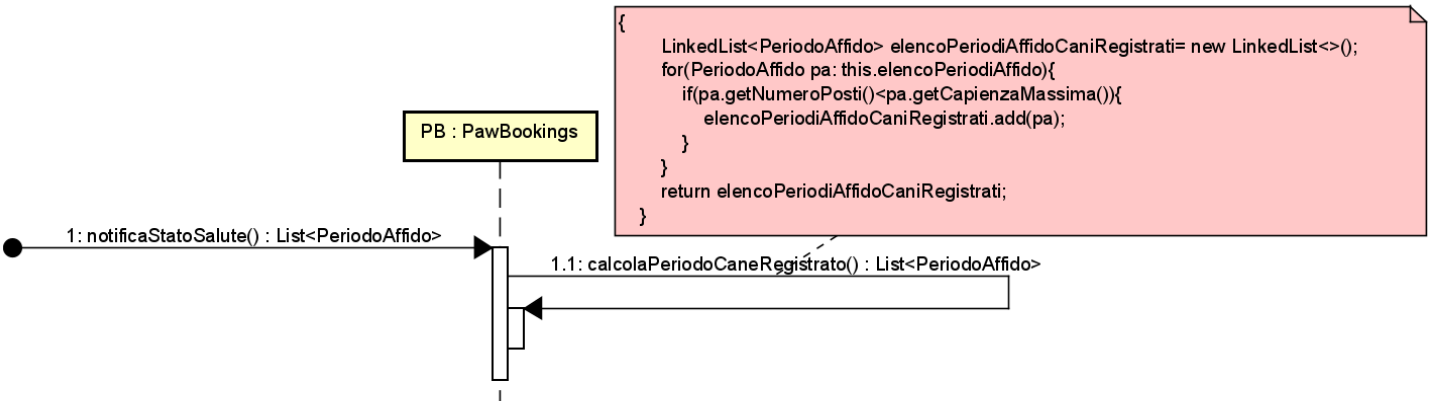
2.2.21. UC10 SD2



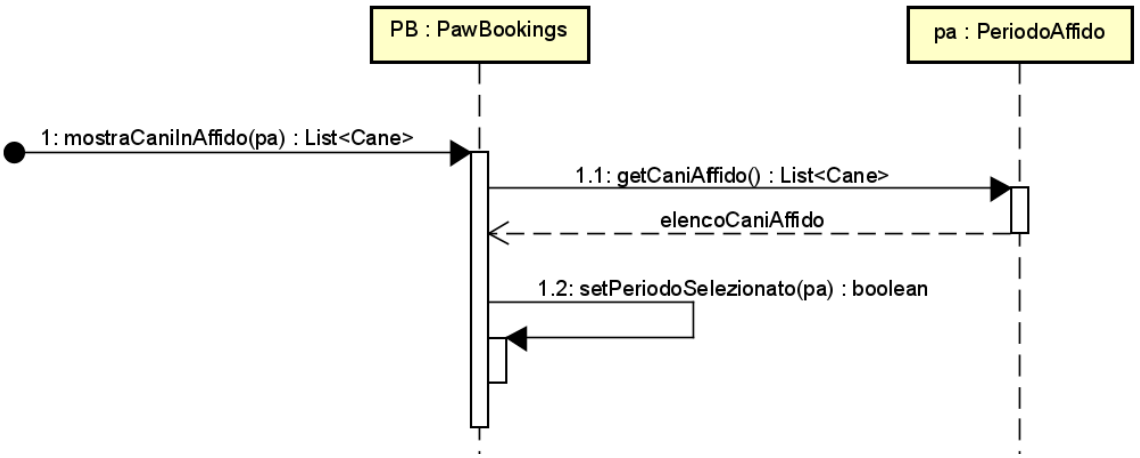
2.2.22. UC11 SD1



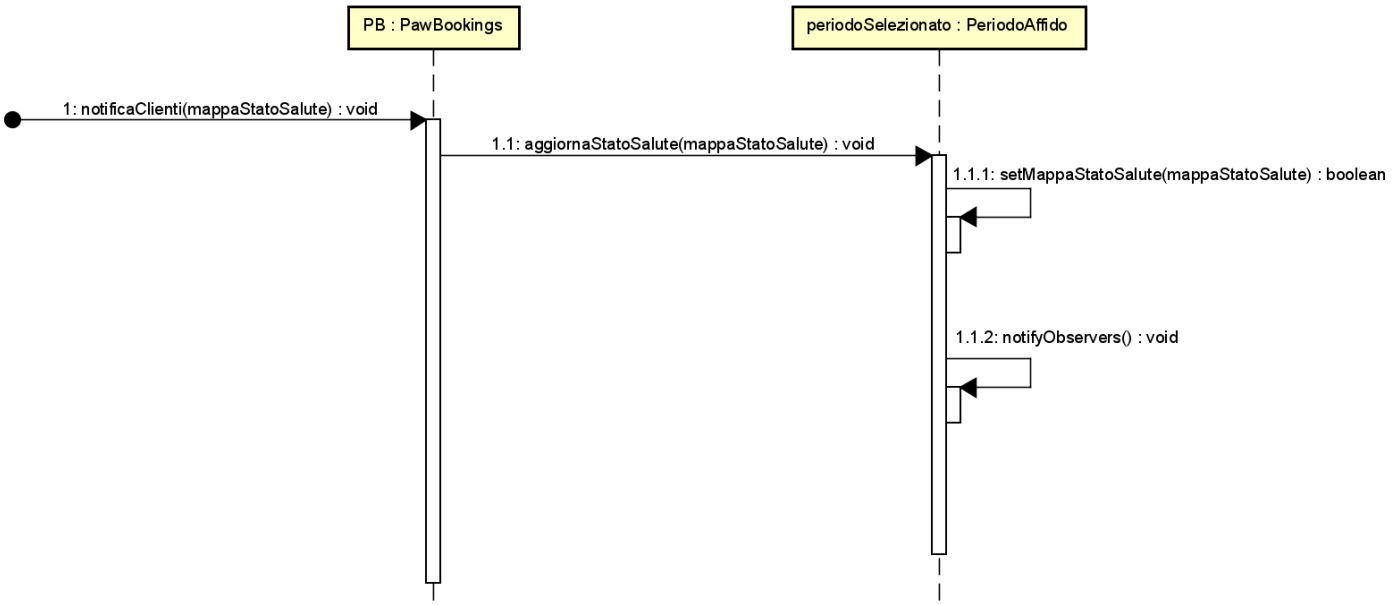
2.2.23. UC12 SD1



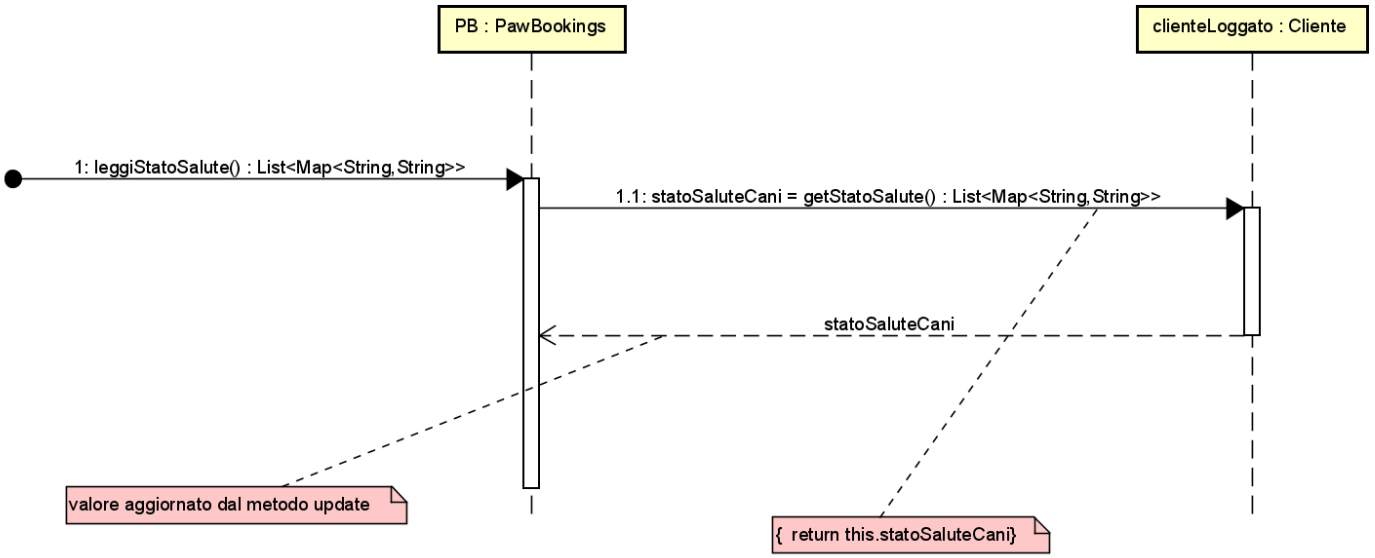
2.2.24. UC12 SD2



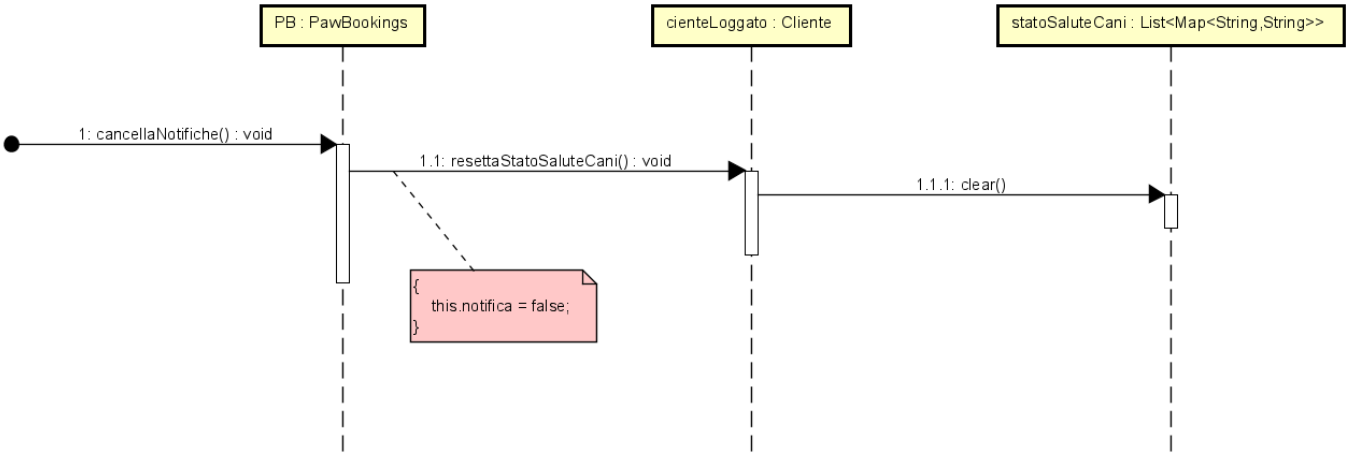
2.2.25. UC12 SD3



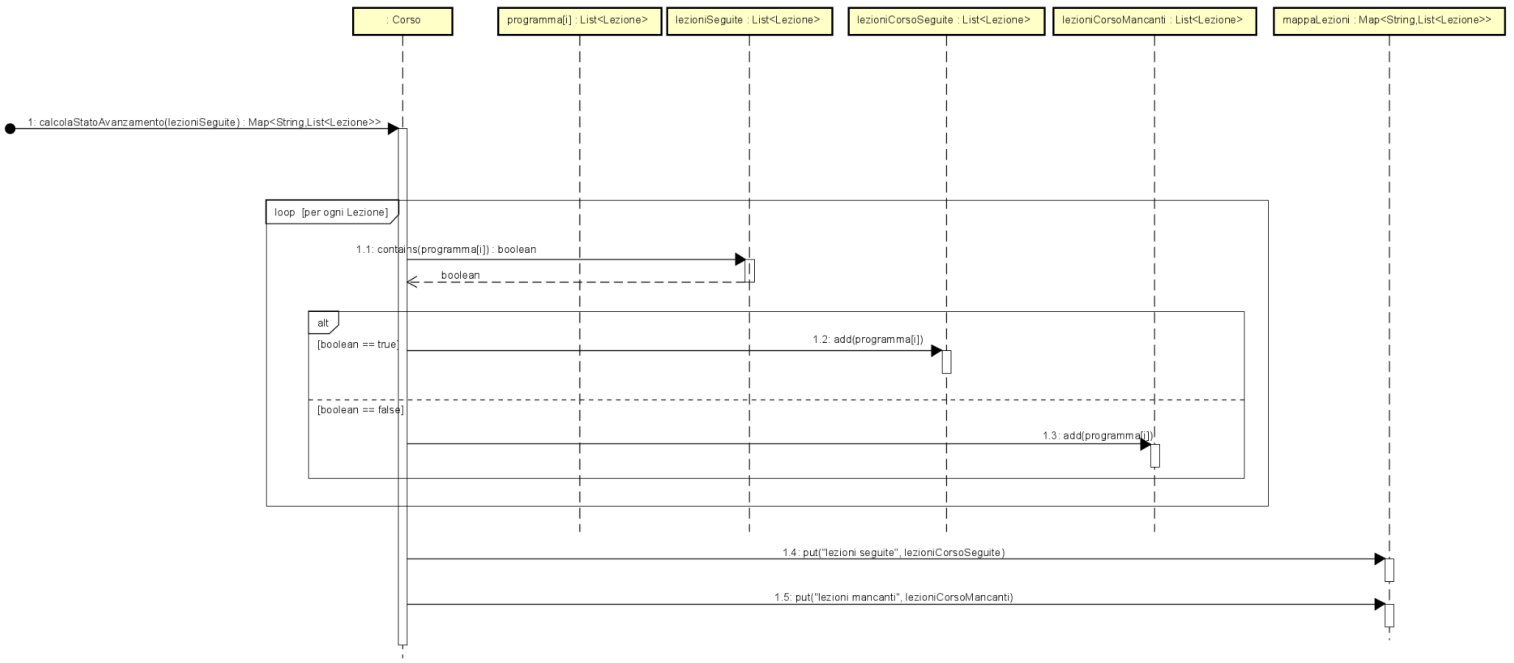
2.2.26. UC13 SD1



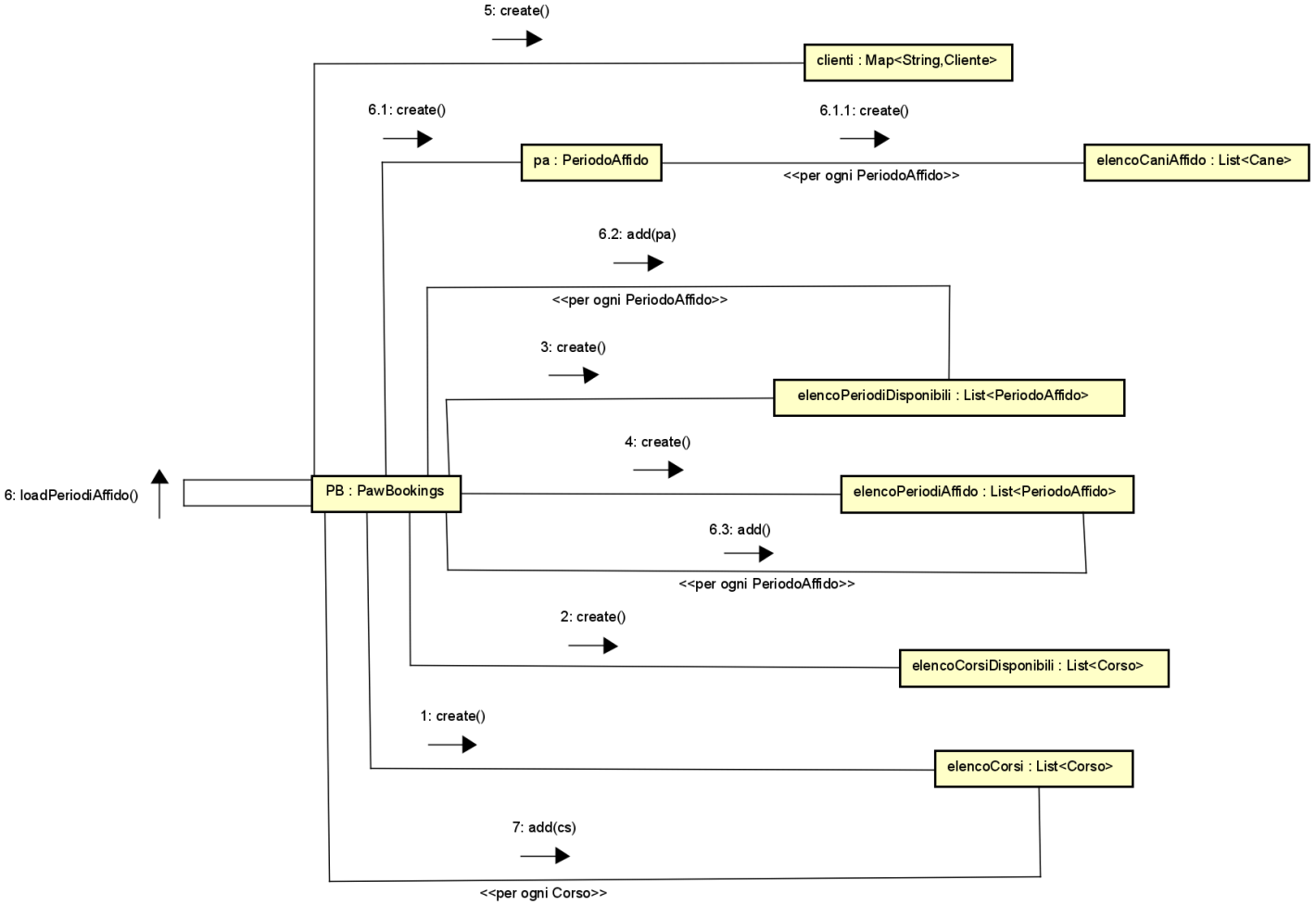
2.2.27. UC13 SD2



2.2.28. SD: calcolaStatoAvanzamento



2.2.29. 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 "*iterazione3.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 terza iterazione, tenga conto del flusso degli scenari del caso d'uso UC7 e degli scenari principali di successo dei casi d'uso UC8, UC9, UC10, UC11, UC12, UC13.
Per questa terza iterazione i test unitari relativi alla classe PawBookings sono stati organizzati in diverse classi di test nel modo seguente:

- PawBookingsTest1: classe di test che comprende i test unitari relativi ai casi d'uso UC1-UC2.
- PawBookingsTest2: classe di test che comprende i test unitari relativi ai casi d'uso UC3-UC4-UC5-UC6.
- PawBookingsTest3: classe di test che comprende i test unitari relativi ai casi d'uso UC7-UC8.
- PawBookingsTest4: classe di test che comprende i test unitari relativi ai casi d'uso UC9-UC11-UC12-UC13 con l'aggiunta di alcuni test unitari relativi ai casi d'uso UC2-UC3-UC4 necessari per i nuovi casi d'uso introdotti.

Per quanto riguarda la classe Cane i test unitari sono stati organizzati in due diverse classi di test:

- CaneTest1: classe di test che comprende i test unitari relativi ai casi d'uso UC1-UC2.
- CaneTest2: classe di test che comprende i test unitari relativi ai casi d'uso UC3-UC4-UC5-UC6-UC7-UC8-UC9-UC11-UC12-UC13.

I test unitari implementati per le precedenti iterazioni sono stati adattati alle modifiche introdotte dalla nuova iterazione e sono funzionanti.
Sono poi state introdotte nuove classi di test quali: ClienteTest, PeriodoAffidoTest, LezioneTest.

In particolar modo sono stati individuati i seguenti metodi delle relative classi:

- PawBookings
  - loadPeriodiAffido
    - viene verificato che la lunghezza dell'elenco dei periodi di affido di PawBookings sia uguale al numero di istanze di periodi di affido
    - viene verificato che la lunghezza dell'elenco dei periodi di affido disponibili di PawBookings sia uguale al numero di istanze di periodi di affido
  - inserisciNuovoCorso
    - viene verificato che chiamando il metodo sotto test, la lunghezza dell'elencoCorsi di PawBookings venga incrementata di 1
    - viene verificato che chiamando il metodo sotto test e passando come parametri il tipo di corso, la capienza ed il costo, l'elenco dei corsi di PawBookings contenga il tipo di corso inserito come parametro del metodo sotto test
  - generaCodiceCorso
    - viene verificato che chiamando il metodo sotto test, venga restituito un numero pari al numero di corsi dell'elencoCorsi di PawBookings più 1
  - selezionaCorso
    - viene verificato che chiamando il metodo sotto test e passando come parametro il Corso, il valore della variabile corsoSelezionato sia uguale al Corso passato come parametro metodo sotto test
  - setCorsoSelezionato
    - viene verificato che chiamando il metodo sotto test e passando come parametro il Corso, il valore della variabile corsoSelezionato sia uguale al Corso passato come parametro metodo sotto test

        - viene verificato che chiamando il metodo sotto test e passando come parametro il Corso, il valore booleano restituito sia true
        - viene verificato che chiamando il metodo sotto test e passando come parametro null, il valore della variabile corsoSelezionato sia null
        - viene verificato che chiamando il metodo sotto test e passando come parametro null, il valore booleano restituito sia false
    - modificaCorso
        - viene verificato che chiamando il metodo sotto test e passando come parametri la capienza e il costo, il valore della variabile capienza del corso selezionato di PawBookings sia uguale al valore della capienza passato come parametro del metodo sotto test
        - viene verificato che chiamando il metodo sotto test e passando come parametri la capienza e il costo, il valore della variabile costo del corso selezionato di PawBookings sia uguale al valore del costo passato come parametro del metodo sotto test
    - nuovaLezione
        - viene verificato che chiamando il metodo sotto test, il valore della variabile lezioneCorrente di PawBookings non sia null
    - generaCodiceLezione
        - viene verificato che chiamando il metodo sotto test, il valore della variabile codiceLezione sia uguale al numero di lezioni del programma del Corso selezionato più 1
    - inserisciEsercizio
        - viene verificato che chiamando il metodo sotto test, la lunghezza del della lista di esercizi della lezione del corso selezionato sia uguale a 1
        - viene verificato che chiamando nuovamente il metodo sotto test, la lunghezza del della lista di esercizi della lezione del corso selezionato sia uguale a 2
    - confermaLezione
        - viene verificato che chiamando il metodo sotto test, la nuova lezione sia presente all'interno del programma del corso selezionato
        - viene verificato che chiamando il metodo sotto test, la lunghezza del programma del corso selezionato sia stata incrementata di 1
    - calcolaCorsiConCaniIscritti
        - viene verificato che l'elenco dei corsi restituito a seguito dell'invocazione del metodo sotto test non contenga un corso che sappiamo per certo non contenere alcun cane
        - viene verificato che l'elenco dei corsi restituito a seguito dell'invocazione del metodo sotto test contenga un corso che sappiamo per certo contenere almeno un cane
    - selezionaLezione
        - viene verificato che chiamando il metodo sotto test e passando come parametro la Lezione, il valore delle variabile lezioneSelezionata di Corso sia uguale alla Lezione passata come parametro del metodo sotto test
    - verificaDatiTurno
        - viene verificato che chiamando il metodo sotto test e passando come parametri la Data corretta, l'oraInizio corretta e l'oraFine corretta, venga restituito true
        - viene verificato che chiamando il metodo sotto test e passando come parametri la Data errata, l'oraInizio corretta e l'oraFine corretta, venga restituito false
        - viene verificato che chiamando il metodo sotto test e passando come parametri la Data corretta, l'oraInizio errata e l'oraFine corretta, venga restituito false
        - viene verificato che chiamando il metodo sotto test e passando come parametri la Data errata, l'oraInizio errata e l'oraFine corretta, venga restituito false

- nuovoTurno
  - viene verificato che chiamando il metodo sotto test e passando come parametri la Data corretta, l'oraInizio corretta e l'oraFine corretta, la lunghezza dell'elencoTurniDisponibili della Lezione selezionata sia stata incrementata di 1
  - viene verificato che chiamando il metodo sotto test e passando come parametri la Data errata, l'oraInizio corretta e l'oraFine corretta, la lunghezza dell'elencoTurniDisponibili della Lezione selezionata non sia variata
- verificaIdoneitaScambioTurno
  - viene verificato che chiamando il metodo sotto test e passando come parametro il Turno valido, venga restituito true
  - viene verificato che chiamando il metodo sotto test e passando come parametro il Turno non valido, venga restituito false
  - viene verificato che chiamando il metodo sotto test e passando come parametro null, venga restituito false
- scambioTurno
  - viene verificato che chiamando il metodo sotto test, venga restituita la lista dei turni disponibili che non contiene il turno selezionato
- selezionaTurnoScambio
  - viene verificato che chiamando il metodo sotto test e passando come parametro un Turno disponibile, il Turno corrente del Cane selezionato sia uguale al Turno passato come parametro all'interno del metodo sotto test
- mostraStatoAvanzamentoCorso
  - viene verificato che chiamando il metodo sotto test, la lista delle Lezioni seguite dal Cane selezionato sia uguale a 0
  - viene verificato che chiamando il metodo sotto test, la lista delle Lezioni mancanti contenga le Lezioni del programma del Corso che il Cane selezionato deve ancora seguire
  - viene verificato che prenotando il Cane selezionato ad una lezione e chiamando il metodo sotto test, la lista delle Lezioni seguite dal Cane selezionato sia uguale a 1
  - viene verificato che chiamando il metodo sotto test, la lista delle Lezioni mancanti contenga le Lezioni del programma del Corso che il Cane selezionato deve ancora seguire
- calcolaPeriodoCaneRegistrato
  - viene verificato che chiamando il metodo sotto test, venga restituito un elenco di periodi di affido con cani registrati che abbia lunghezza pari alla lunghezza iniziale dell'elenco di periodi di affido con cani registrati più 1
  - viene verificato che chiamando il metodo sotto test, l'elenco dei periodi di affido con cani registrati contenga il periodo selezionato
  - viene verificato che ponendo un secondo cane in affido in un periodo di affido differente e chiamando il metodo sotto test, venga restituito un elenco di periodi di affido con cani registrati che abbia lunghezza pari alla lunghezza dell'elenco di periodi di affido con cani registrati più 1
  - viene verificato che chiamando il metodo sotto test, l'elenco dei periodi di affido con cani registrati contenga l'ultimo periodo selezionato
- periodoSelezionato
  - viene verificato che chiamando il metodo sotto test e passando come parametro il Periodo di Affido, il periodo selezionato sia uguale al periodo di affido passato come parametro all'interno del metodo sotto test
  - viene verificato che chiamando il metodo sotto test e passando come parametro null, il periodo selezionato sia uguale a null

    - mostraCaniAffido
        - viene verificato che, selezionando un periodo di affido e ponendo in affido un cane, chiamando il metodo sotto test e passando come parametro il periodo di affido, la lunghezza della lista dei cani in affido del periodo selezionato sia uguale alla lunghezza della lista dei cani in affido, precedente all'affido del cane selezionato, più 1
        - viene verificato che, selezionando un periodo di affido e ponendo in affido un cane, chiamando il metodo sotto test e passando come parametro il periodo di affido, il cane contenuto nella lista dei cani in affido del periodo selezionato sia il cane posto in affido
        - viene verificato che, concludendo l'affido del cane selezionato, chiamando il metodo sotto test e passando come parametro il periodo di affido, la lunghezza della lista dei cani in affido del periodo selezionato sia uguale alla lunghezza della lista dei cani in affido, precedente all'affido del cane selezionato
    - notificaClienti
        - viene verificato che, avendo 2 clienti con cani in affido ed un cliente che non ha cani in affido, chiamando il metodo sotto test e passando come parametro la mappa dello stato salute dei cani in affido, l'attributo notifica dei clienti che hanno cani in affido sia true
        - viene verificato che, avendo 2 clienti con cani in affido ed un cliente che non ha cani in affido, chiamando il metodo sotto test e passando come parametro la mappa dello stato salute dei cani in affido, l'attributo notifica del cliente che non ha cani in affido sia false
        - viene verificato che, avendo 2 clienti con cani in affido ed un cliente che non ha cani in affido, chiamando il metodo sotto test e passando come parametro la mappa dello stato salute dei cani in affido, la lista statoSaluteCani dei clienti che hanno cani in affido non sia vuota
        - viene verificato che, avendo 2 clienti con cani in affido ed un cliente che non ha cani in affido, chiamando il metodo sotto test e passando come parametro la mappa dello stato salute dei cani in affido, la lista statoSaluteCani del cliente che non ha cani in affido sia vuota
    - cancellaNotifiche
        - viene verificato che, avendo 2 clienti con cani in affido ed effettuando il login con uno di essi, chiamando il metodo sotto test, l'attributo notifica del cliente loggato sia diventato false
        - viene verificato che, avendo 2 clienti con cani in affido ed effettuando il login con uno di essi, chiamando il metodo sotto test, la lista statoSaluteCani del cliente loggato sia vuota
        - viene verificato che, avendo 2 clienti con cani in affido ed effettuando il login con uno di essi, chiamando il metodo sotto test, l'attributo notifica del cliente non loggato sia true
        - viene verificato che, avendo 2 clienti con cani in affido ed effettuando il login con uno di essi, chiamando il metodo sotto test, la lista statoSaluteCani del cliente non loggato non sia vuota
- Cane
    - aggiornaAttualmenteInAffido
        - viene verificato che chiamando il metodo sotto test, venga restituito true se il valore dell'attributo attualmenteInAffido di Cane è true
    - conclusioneAffido
        - viene verificato che chiamando il metodo sotto test, venga restituito false se il valore dell'attributo attualmenteInAffido di Cane è false

        - viene verificato che chiamando il metodo sotto test, il valore dell'attributo affidoCorrente di Cane sia null
    - aggiornaAssociazioniCane
        - viene verificato che chiamando il metodo sotto test, l'elenco dei cani iscritti al corso non contenga più il cane selezionato
- Cliente
    - calcolaCaniNonInAffido
        - viene verificato che, avendo posto in affido uno dei due Cani in possesso del Cliente, chiamando il metodo sotto test, la lista dei Cani non in affido posseduti dal Cliente contenga un solo Cane
        - viene verificato che, avendo posto in affido uno dei due Cani in possesso del Cliente, chiamando il metodo sotto test, la lista dei Cani non in affido posseduti dal Cliente contenga il Cane non in affido
        - viene verificato che, avendo posto in affido uno dei due Cani in possesso del Cliente, chiamando il metodo sotto test, la lista dei Cani non in affido posseduti dal Cliente non contenga il Cane in affido
    - registraCane
        - viene verificato che chiamando il metodo sotto test, la lunghezza della lista dei Cani posseduti dal Cliente sia stata incrementata di 1
    - rimuoviCane
        - viene verificato che, selezionando il cliente, chiamando il metodo sotto test e passando come parametro il Cane posseduto dal Cliente, la lista dei cani posseduti del cliente contenga un cane in meno
    - iscrizioneNotificheStatoSalute
        - viene verificato che, selezionando il cliente, chiamando il metodo sotto test e passando come parametro il periodo di affido, la lista degli observers contenga un solo elemento
    - annullamentoIscrizione
        - viene verificato che, selezionando il cliente, chiamando il metodo sotto test e passando come parametro il periodo di affido, la lista degli observers sia vuota
    - resettaStatoSaluteCane
        - viene verificato che, selezionando il cliente e chiamando il metodo sotto test, la mappaStatoSaluteCani sia stata svuotata
    - update
        - viene verificato che, selezionando il periodo di affido, chiamando il metodo iscrizioneNotificheStatoSalute e chiamando il metodo sotto test, la lista statoSaluteCani di Cliente contenga un solo elemento
- Corso
    - setLezioneCorrente
        - viene verificato che, chiamando il metodo sotto test e passando come parametro la Lezione, il valore restituito sia true
        - viene verificato che, chiamando il metodo sotto test e passando come parametro la Lezione, la lezione corrente sia uguale alla lezione passata come parametro del metodo sotto test
        - viene verificato che, chiamando il metodo sotto test e passando come parametro null, il valore restituito sia false
        - viene verificato che, chiamando il metodo sotto test e passando come parametro la Lezione, la lezione corrente sia null
    - aggiornaCaniIscritti
        - viene verificato che, chiamando il metodo sotto test e passando come parametro il Cane iscritto al Corso, nella lista dei cani iscritti al corso non sia presente il Cane passato come parametro del metodo sotto test

- annullaIscrizione
  - viene verificato che, chiamando il metodo sotto test e passando come parametro il Cane iscritto al Corso, nella lista dei cani iscritti al corso non sia presente il Cane passato come parametro del metodo sotto test
- aggiornaInformazioni
  - viene verificato che, chiamando il metodo sotto test e passando come parametri la capienza e il costo del Corso, sia stata aggiornata la capienza del corso selezionato
  - viene verificato che, chiamando il metodo sotto test e passando come parametri la capienza e il costo del Corso, sia stato aggiornato il costo del corso selezionato
- nuovaLezione
  - viene verificato che, chiamando il metodo sotto test e passando come parametri il codice lezione e il nome lezione, la nuova lezione sia stata creata e settata come lezione corrente
- confermaInserimentoLezione
  - viene verificato che, chiamando il metodo sotto test relativamente al corso selezionato, la nuova lezione sia stata aggiunta al programma del corso
- aggiornaLezione
  - viene verificato che, chiamando due volte il metodo sotto test e passando come parametri il nome e la descrizione dell'esercizio, la lunghezza della lista di esercizi della lezione corrente del corso selezionato sia stata incrementata di 2
- generaCodiceTurno
  - viene verificato che, chiamando il metodo sotto test, il valore della variabile numTurni di Corso venga incrementata di 1
- aggiungiTurnoLezione
  - viene verificato che, chiamando il metodo sotto test e passando come parametri Data, oraInizio e oraFine, la variabile numTurni di Corso venga incrementata di 1
  - viene verificato che, chiamando il metodo sotto test e passando come parametri Data, oraInizio e oraFine, il codice turno generato sia uguale al numTurni + 1
  - viene verificato che, chiamando il metodo sotto test e passando come parametri Data, oraInizio e oraFine, i turni siano stati aggiunti all'elenco dei turni disponibili della lista Turni

- Lezione
  - aggiornaTurniDisponibili
    - viene verificato che, chiamando il metodo sotto test e passando come parametro l'ultimo turno disponibile dell'elenco turni disponibili della lezione, il turno non sia più presente nell'elenco dei turni disponibili della lezione
  - nuovoEsercizio
    - viene verificato che, chiamando il metodo sotto test e passando come parametri il nome e la descrizione dell'esercizio, la lunghezza della lista esercizi della lezione sia stata incrementata
  - nuovoTurno
    - viene verificato che, chiamando il metodo sotto test e passando come parametri il codice turno, la data, l'oraInizio e l'oraFine, la lunghezza lista dei turni disponibili sia stata incrementata
  - effettuaScambioTurno
    - viene verificato che, associato il primo turno della lista turni disponibili al turnoCorrente e l'ultimo turno della lista turni disponibili al turnoSelezionato, chiamando il metodo sotto test e passando come parametri turnoCorrente e turnoSelezionato, il turnoCorrente sia stato aggiunto all'elenco dei turni disponibili delle lezione del corso

        - viene verificato che, associato il primo turno della lista turni disponibili al turnoCorrente e l'ultimo turno della lista turni disponibili al turnoSelezionato, chiamando il metodo sotto test e passando come parametri turnoCorrente e turnoSelezionato, il turnoSelezionato sia stato rimosso dall'elenco dei turni disponibili delle lezione del corso
- Periodo Affido
    - registraAffido
        - viene verificato che, chiamando il metodo sotto test e passando come parametro il Cane del periodo selezionato, la lunghezza della lista dei cani in affido del periodo selezionato sia stata incrementata
        - viene verificato che, chiamando il metodo sotto test e passando come parametro il Cane del periodo selezionato, la variabile numeroPosti di Periodo Affido sia stata decrementata
    - concludiAffido
        - viene verificato che, chiamando il metodo sotto test e passando come parametro il Cane del periodo selezionato, la lunghezza della lista dei cani in affido del periodo selezionato sia stata decrementata
    - verificaIscrizione
        - viene verificato che, chiamando il metodo sotto test e passando come parametro il Cliente, uno dei cani posseduti dal Cliente non sia presente nell'elenco dei cani in affido nel periodo selezionato