Sommario

1. Ideazione e analisi dei requisiti

1.1 Requisiti

“EasyDrive” è un sistema software per la gestione di una scuola guida. Il sistema deve occuparsi di tutte le attività che fornisce una scuola guida, quali ad esempio l’iscrizione di nuovi clienti, la gestione delle lezioni teoriche, delle guide e degli esami. Inoltre, il sistema tiene traccia della frequenza di ciascun iscritto in modo da tener aggiornato l’insegnante ed eventualmente poter prenotare l’iscritto per il prossimo esame fissato; nel caso in cui non riesca a passarlo per la seconda volta dovrà pagare nuovamente l’iscrizione al corso di guida.

Il sistema software deve occuparsi dei vari aspetti:

- Registrazione nel sistema di un nuovo cliente della scuola guida;
- Prenotazione esame teorico o finale;
- Aggiornamento della programmazione degli esami (teorici e finali) con modifica, aggiunta o eliminazione di date e orari;
- Aggiornamento della programmazione delle lezioni con modifica, aggiunta o eliminazione di date e orari;
- Aggiornamento frequenza degli iscritti;
- Prenotazione guide (solo per chi ha superato l’esame teorico);
- Annullamento prenotazione guide;
- Pubblicazione esiti esami;

1.2 Obiettivi e Casi d’uso

Attore	**Obiettivo**	**Caso d’uso**
Amministratore	Inserire, rimuovere, cercare o modificare un cliente all’interno della scuola guida	UC1: Gestisci cliente (CRUD)
Amministratore	Inserire, rimuovere, cercare o modificare la programmazione di un esame teorico	UC2: Gestisci programmazione esame teorico (CRUD)
Amministratore	Gestire la prenotazione dell’esame teorico dei propri iscritti	UC3: Prenota esame teorico

Amministratore	Gestire la prenotazione dell'esame finale dei propri iscritti	UC4: Prenota esame finale
Amministratore	Inserire, rimuovere, cercare o modificare la programmazione delle lezioni	UC5: Gestisci programmazione lezioni (CRUD)
Amministratore	Gestire la frequenza dei clienti alle lezioni	UC6: Aggiorna frequenza clienti
Amministratore	Inserire, rimuovere, cercare o modificare la programmazione delle guide	UC7: Gestisci programmazione guide (CRUD)
Amministratore	Gestire la prenotazione delle guide (solo per chi ha superato l'esame teorico)	UC8: Prenota guida
Amministratore	Aggiornare il numero di guide sostenute dai clienti	UC9: Aggiorna numero guide sostenute
Amministratore	Pubblicare gli esiti degli esami teorici sostenuti dai clienti	UC10: Pubblica esiti esami teorico
Amministratore	Pubblicare gli esiti degli esami finali sostenuti dai clienti	UC11: Pubblica esiti esami finale
Amministratore	Inserire, rimuovere, cercare o modificare la programmazione di un esame finale	UC12: Gestisci programmazione esame finale (CRUD)

1.3 Modello dei casi d'uso

Tra tutti i casi d'uso individuati, si è scelto di fornire una descrizione in formato dettagliato per i seguenti casi d'uso:

- Prenota esame teorico;
- Prenota esame finale;
- Aggiorna frequenza clienti;
- Prenota guida;
- Aggiorna numero guide sostenute;
- Pubblica esiti esame teorico;
- Pubblica esiti esame finale;

Per i restanti casi d'uso si fornisce una descrizione in formato breve.

UC3: Prenota esame teorico

Nome del caso d'uso	UC3: Prenota esame teorico
Portata	Software EasyDrive
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- Amministratore: vuole prenotare un cliente ad un esame teorico. - Cliente: desidera effettuare l'esame.
Pre-condizioni	- Il cliente deve essere registrato nel sistema. - Il cliente deve aver frequentato almeno il 70% delle lezioni per essere prenotato.
Garanzia di successo	Il cliente è stato prenotato per una data d'esame fissata.
Scenario principale di successo	1. Il cliente vuole essere prenotato per l'esame teorico. 2. L'amministratore sceglie l'attività "Prenota esame teorico". 3. Il sistema mostra le date e orari disponibili per sostenere l'esame selezionato. 4. L'amministratore elenca al cliente le date e orari disponibili per sostenere l'esame. 5. Il cliente sceglie tra le date e orari disponibili e comunica la propria disponibilità. 6. L'amministratore seleziona la data e ora scelta dal cliente. 7. L'amministratore inserisce il codice fiscale del cliente. 8. L'amministratore sceglie "conferma prenotazione esame teorico".
Estensioni	***a.** In un qualsiasi momento il sistema fallisce e si arresta improvvisamente.

	1. L’amministratore riavvia il software e ripristina lo stato precedente del sistema. 2. Il sistema ripristina lo stato. **3a.** Il sistema non ha nessuna data disponibile per l’esame. 1. Il sistema fa presente all’amministratore dell’assenza di una data per l’esame. 2. L’amministratore comunica al cliente che verrà informato quando saranno disponibili nuove date per l’esame. **7a.** Il sistema non trova il codice fiscale del cliente. 1. Il sistema chiede all’amministratore se il codice fiscale richiesto sia corretto. 2. L’amministratore verifica il codice fiscale e lo inserisce nuovamente ripetendo il passo 7. **7b.** Il sistema rileva che il cliente non ha frequentato almeno il 70% delle lezioni. 1. Il sistema annulla la prenotazione. 2. L’amministratore informa il cliente di raggiungere la soglia di frequenze a lezione minima.
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	
Frequenza di ripetizioni	Legato alla necessità di prenotare un cliente ad un esame teorico.
Varie	

UC4: Prenota esame finale

Nome del caso d'uso	UC4: Prenota esame finale
Portata	Software EasyDrive
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- Amministratore: vuole prenotare un cliente ad un esame finale. - Cliente: desidera effettuare l'esame.
Pre-condizioni	- Il cliente deve essere registrato nel sistema. - Il cliente deve aver superato l'esame teorico ed effettuato almeno 15 guide.
Garanzia di successo	Il cliente è stato prenotato per una data d'esame fissata.
Scenario principale di successo	1. Il cliente vuole essere prenotato per l'esame finale. 2. L'amministratore sceglie l'attività "Prenota esame finale". 3. Il sistema mostra le date disponibili per sostenere l'esame selezionato. 4. L'amministratore elenca al cliente le date disponibili per sostenere l'esame. 5. Il cliente sceglie tra le date disponibili e comunica la propria disponibilità. 6. L'amministratore seleziona la data e l'ora scelta dal cliente. 7. L'amministratore inserisce il codice fiscale del cliente. 8. L'amministratore sceglie "conferma prenotazione esame finale".
Estensioni	***a.** In un qualsiasi momento il sistema fallisce e si arresta improvvisamente.

	1. L'amministratore riavvia il software e ripristina lo stato precedente del sistema. 2. Il sistema ripristina lo stato. **3a.** Il sistema non ha nessuna data disponibile per l'esame. 1. Il sistema fa presente all'amministratore dell'assenza di una data per l'esame. 2. L'amministratore comunica al cliente che verrà informato quando saranno disponibili nuove date per l'esame. **7a.** Il sistema non trova il codice fiscale del cliente. 1. Il sistema chiede all'amministratore se il codice fiscale richiesto sia corretto. 2. L'amministratore verifica il codice fiscale e lo inserisce nuovamente ripetendo il passo 7. **7b.** Il sistema rileva che il cliente non ha ancora superato l'esame teorico. 1. Il sistema avvisa l'amministratore che il cliente non può prenotarsi per l'esame finale. 2. L'amministratore invita il cliente a sostenere prima l'esame teorico. **7c.** Il sistema rileva che il cliente non ha ancora effettuato almeno 15 guide. 1. Il sistema avvisa l'amministratore che il cliente non può prenotarsi per l'esame finale. 2. L'amministratore invita il cliente a prenotarsi per le guide.
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	

Frequenza di ripetizioni	Legato alla necessità di prenotare un cliente ad un esame finale.
Varie	

UC6: Aggiorna frequenza clienti

Nome del caso d'uso	UC6: Aggiorna frequenza clienti
Portata	Software EasyDrive
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- Amministratore: vuole tenere aggiornato il conto delle frequenze dei clienti presenti a lezione. - Cliente: vuole frequentare almeno il 70% delle lezioni.
Pre-condizioni	- Il cliente deve essere registrato nel sistema. - Il sistema deve avere registrata la lezione corrente, per la quale l'amministratore sta aggiornando le frequenze.
Garanzia di successo	- La lezione corrente ha una lista con tutti i presenti. - Ogni cliente presente nella lezione corrente ha aggiornata la sua frequenza alle lezioni.
Scenario principale di successo	1. L'amministratore sceglie l'attività "Aggiorna frequenza clienti" e inserisce la data e l'ora della lezione in cui aggiornare le frequenze. 2. L'amministratore inserisce il codice fiscale di un cliente presente. *Il passo 2 viene ripetuto finché serve.* 3. L'amministratore conferma di aver finito.

Estensioni	***a.** In un qualsiasi momento il sistema fallisce e si arresta improvvisamente. 1. L'amministratore riavvia il software e ripristina lo stato precedente del sistema. 2. Il sistema ripristina lo stato. **1a.** Il sistema non ha nessuna lezione programmata per la data e l'ora selezionata dall'amministratore. 1. Il sistema fa presente all'amministratore di non avere nessuna lezione programmata per la data e ora selezionate. 2. L'amministratore verifica la data e ora ed eventualmente ripete il passo 3. **2a.** Il sistema non trova il codice fiscale del cliente. 1. Il sistema chiede all'amministratore se il codice fiscale richiesto sia corretto. 2. L'amministratore verifica il codice fiscale e lo inserisce nuovamente ripetendo il passo 5.
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	
Frequenza di ripetizioni	Legato al numero di lezioni programmate.
Varie	

UC8: Prenota guida

Nome del caso d'uso	UC8: Prenota guida
Portata	Software EasyDrive
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- Amministratore: vuole prenotare un cliente per la lezione di guida.

	- Cliente: desidera essere prenotato per la lezione di guida.
Pre-condizioni	- Il cliente deve essere già registrato nel sistema. - Il cliente deve aver superato l'esame teorico.
Garanzia di successo	Il cliente viene prenotato per la lezione di guida.
Scenario principale di successo	1. L'amministratore vuole prenotare un cliente ad una lezione di guida. 2. L'amministratore sceglie l'attività "Prenota guida". 3. Il sistema mostra le date disponibili per sostenere le lezioni di guida. 4. L'amministratore chiede al cliente per quale data vuole essere prenotato. 5. Il cliente sceglie tra le date disponibili e comunica la propria disponibilità. 6. L'amministratore seleziona la data scelta dal cliente. 7. L'amministratore inserisce il codice fiscale del cliente. 8. L'amministratore sceglie "conferma prenotazione guida"
Estensioni	***a.** In un qualsiasi momento il sistema fallisce e si arresta improvvisamente. 1. L'amministratore riavvia il software e ripristina lo stato precedente del sistema. 2. Il sistema ripristina lo stato. **3a.** Il sistema non ha nessuna data disponibile per prenotare una guida. 1. Il sistema fa presente all'amministratore dell'assenza di una data per sostenere la lezione di guida.

	2. L'amministratore comunica al cliente che verrà informato quando saranno disponibili nuove date per la lezione. **7a.** Il sistema non trova il codice fiscale del cliente. 1. Il sistema chiede all'amministratore se il codice fiscale richiesto sia corretto. 2. L'amministratore verifica il codice fiscale e lo inserisce nuovamente ripetendo il passo 7.
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	
Frequenza di ripetizioni	Legato alla necessità di prenotare un cliente ad una nuova lezione di guida
Varie	

UC9: Aggiorna numero guide sostenute

Nome del caso d'uso	UC9: Aggiorna numero guide sostenute
Portata	Software EasyDrive
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- Amministratore: vuole aggiornare il numero di guide sostenute da un cliente. - Cliente: desidera che il conteggio del numero di guide sia aggiornato.
Pre-condizioni	- Il cliente deve essere già registrato nel sistema. - Il cliente deve essere prenotato per una guida.
Garanzia di successo	Il numero di guide sostenute da un cliente viene aggiornato.

Scenario principale di successo	1. L'amministratore vuole aggiornare il numero di guide sostenute da un cliente. 2. L'amministratore sceglie l'attività "Aggiorna numero guide". 3. Il sistema mostra la data e ora di tutte le guide antecedenti alla data e ora odierna. 4. L'amministratore inserisce la data e l'ora della guida. 5. Il sistema ritorna una lista di tutti i prenotati alla guida selezionata. 6. L'amministratore inserisce il codice fiscale del cliente del quale si desidera aggiornare il numero di guide. *Il passo 6 viene ripetuto finché serve.* 7. L'amministratore indica di aver finito. 8. Il sistema in base alla scelta dell'amministratore aggiorna il numero di guide dei clienti.
Estensioni	***a.** In un qualsiasi momento il sistema fallisce e si arresta improvvisamente. 1. L'amministratore riavvia il software e ripristina lo stato precedente del sistema. 2. Il sistema ripristina lo stato. **4a.** L'amministratore inserisce una data e ora non presente tra quelle disponibili per gli esami precedentemente prenotabili 1. Il sistema fa presente all'amministratore dell'assenza di una data e ora per l'esame indicato. 2. L'amministratore verifica data e ora e li inserisce nuovamente ripetendo il passo 4. **6a.** Il sistema non trova il codice fiscale del cliente.

	1. Il sistema chiede all'amministratore se il codice fiscale richiesto sia corretto. 2. L'amministratore verifica il codice fiscale e lo inserisce nuovamente ripetendo il passo 6.
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	
Frequenza di ripetizioni	Legato alla necessità di aggiornare il numero di guide di un cliente.
Varie	

UC10: Pubblica esiti esame teorico

Nome del caso d'uso	UC10: Pubblica esiti esame teorico
Portata	Software EasyDrive
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- Amministratore: vuole pubblicare gli esiti degli esami teorici sostenuti dai suoi clienti. In tal modo potrà tenere traccia di chi è momentaneamente in possesso solo del foglio rosa.
Pre-condizioni	- Il cliente deve essere già registrato nel sistema.
Garanzia di successo	L'esito dell'esame teorico del cliente viene registrato all'interno del sistema.
Scenario principale di successo	1. L'amministratore vuole pubblicare gli esiti dell'esame teorico sostenuto dai suoi clienti. 2. L'amministratore sceglie l'attività "Pubblica esiti esame teorico".

	3. Il sistema mostra la data e ora di tutti gli esami teorici antecedenti alla data e ora odierna. 4. L'amministratore inserisce la data e l'ora dell'esame teorico. 5. Il sistema ritorna una lista di tutti i prenotati all'esame selezionato. 6. Il sistema chiede di inserire il codice fiscale del cliente che ha passato l'esame teorico. 7. L'amministratore inserisce il codice fiscale del cliente che ha passato l'esame. *Il passo 7 viene ripetuto finché serve.* 8. L'amministratore indica di aver finito. 9. Il sistema in base alla scelta dell'amministratore aggiorna gli attributi di cliente.
Estensioni	***a.** In un qualsiasi momento il sistema fallisce e si arresta improvvisamente. 1. L'amministratore riavvia il software e ripristina lo stato precedente del sistema. 2. Il sistema ripristina lo stato. **4a.** L'amministratore inserisce una data e ora non presente tra quelle disponibili per gli esami precedentemente prenotabili 3. Il sistema fa presente all'amministratore dell'assenza di una data e ora per l'esame indicato. 4. L'amministratore verifica data e ora e li inserisce nuovamente ripetendo il passo 4. **7a.** Il sistema non trova il codice fiscale del cliente. 3. Il sistema chiede all'amministratore se il codice fiscale richiesto sia corretto.

	4. L'amministratore verifica il codice fiscale e lo inserisce nuovamente ripetendo il passo 7.
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	
Frequenza di ripetizioni	Legato alla necessità di pubblicare gli esiti degli esami teorici dei propri clienti.
Varie	

UC11: Pubblica esiti esame finale

Nome del caso d'uso	UC11: Pubblica esiti esame finale
Portata	Software EasyDrive
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- Amministratore: vuole pubblicare gli esiti degli esami pratici sostenuti dai suoi clienti. In tal modo potrà tenere traccia di chi è riuscito a prendere la patente dei suoi clienti e chi no.
Pre-condizioni	- Il cliente deve essere già registrato nel sistema.
Garanzia di successo	L'esito dell'esame finale del cliente viene registrato all'interno del sistema.
Scenario principale di successo	1. L'amministratore vuole pubblicare gli esiti dell'esame finale sostenuto dai suoi clienti. 2. L'amministratore sceglie l'attività "Pubblica esiti esame finale". 3. Il sistema mostra la data e ora di tutti gli esami finali antecedenti alla data e ora odierna. 4. L'amministratore inserisce la data e l'ora dell'esame finale.

	5. Il sistema ritorna una lista di tutti i prenotati all'esame selezionato. 6. Il sistema chiede di inserire il codice fiscale del cliente che ha passato l'esame finale. 7. L'amministratore inserisce il codice fiscale del cliente che ha passato l'esame. *Il passo 7 viene ripetuto finché serve.* 8. L'amministratore indica di aver finito. 9. Il sistema in base alla scelta dell'amministratore aggiorna gli attributi di cliente.
Estensioni	***a.** In un qualsiasi momento il sistema fallisce e si arresta improvvisamente. 1. L'amministratore riavvia il software e ripristina lo stato precedente del sistema. 2. Il sistema ripristina lo stato. **4a.** L'amministratore inserisce una data e ora non presente tra quelle disponibili per gli esami precedentemente prenotabili 1. Il sistema fa presente all'amministratore dell'assenza di una data e ora per l'esame indicato. 2. L'amministratore verifica data e ora e li inserisce nuovamente ripetendo il passo 6. **7a.** Il sistema non trova il codice fiscale del cliente. 1. Il sistema chiede all'amministratore se il codice fiscale richiesto sia corretto. 2. L'amministratore verifica il codice fiscale e lo inserisce nuovamente ripetendo il passo 7.
Requisiti speciali	
Elenco delle varianti tecnologiche e dei dati	

Frequenza di ripetizioni	Legato alla necessità di pubblicare gli esiti degli esami finali dei propri clienti.
Varie	

UC1: Gestisci cliente (CRUD)
L'amministratore vuole modificare, aggiungere o eliminare i dati di un cliente all'interno del sistema.

UC2: Gestisci programmazione esame teorico (CRUD)
L'amministratore vuole modificare, aggiungere o eliminare la data e l'orario di un tipo di esame teorico all'interno del sistema.

UC5: Gestisci programmazione lezioni (CRUD)
L'amministratore vuole modificare, aggiungere o eliminare la data e l'orario di una lezione all'interno del sistema.

UC7: Gestisci programmazione guide (CRUD)
L'amministratore vuole modificare, aggiungere o eliminare la data e l'orario di una guida all'interno del sistema.

UC12: Gestisci programmazione esame finale (CRUD)
L'amministratore vuole modificare, aggiungere o eliminare la data e l'orario di un tipo di esame finale all'interno del sistema.

1.4 Documento di Visione
Parallelamente alla stesura di questo capitolo è stato redatto il documento di Visione, data la lunghezza di tale elaborato, per la sua visione si rimanda all'**appendice A**.

1.5 Regole di business

Per il corretto utilizzo dell'applicazione devono essere rispettate le seguenti regole di dominio:

ID	Regola	Modificabilità	Sorgente
R1	Se l'esame teorico non viene superato per due volte consecutive il cliente dovrà ripagare per intero la quota di iscrizione	Bassa	Politica interna della scuola guida
R2	Se un iscritto non supera l'esame teorico non può effettuare le prenotazioni per le guide	Bassa	Politica interna della scuola guida
R3	Se un iscritto non ha frequentato almeno il 70% delle lezioni teoriche non può prenotarsi per l'esame teorico	Bassa	Politica interna della scuola guida
R4	Se un iscritto non effettua almeno 15 guide con l'istruttore non può prenotarsi all'esame finale	Bassa	Politica interna della scuola guida
R5	Se l'esame finale non viene superato per due volte consecutive il cliente dovrà sostenere nuovamente l'esame teorico	Bassa	Politica interna della scuola guida

1.6 Glossario

Termine	Definizione
Amministratore/Istruttore	Gestore della scuola guida, si occupa dell'iscrizione degli iscritti, della programmazione e dell'insegnamento delle lezioni teoriche e delle guide.
Cliente	Utente che si reca presso la scuola guida al fine di iscriversi ad un corso di patente.
Lezione	Ciascuno degli incontri fra l'istruttore e uno o più clienti (allievi) nell'ambito dello svolgimento di un programma di studio.
Argomento	Argomento teorico trattato dall'istruttore durante una lezione teorica.
Esame Teorico	Prova teorica a cui un cliente viene sottoposto per verificare l'accertamento della preparazione e dell'idoneità.
Esame Finale	Prova finale a cui un cliente viene sottoposto per verificare l'accertamento della preparazione pratica (guida) e dell'idoneità.
Guida	Lezione pratica che può essere sostenuta da uno o più clienti che possiedono il foglio rosa.
Attività	Attività svolta all'interno della scuola guida, può essere un esame teorico, un esame finale oppure una guida.
EasyDrive	Software per la gestione di tutte le mansioni utili all' interno di una scuola guida.
Esiti Esame Teorico	Esiti della prova teorica a cui un cliente viene sottoposto per verificare l'accertamento della preparazione e dell'idoneità.
Esiti Esame Finale	Esiti della prova finale a cui un cliente viene sottoposto per verificare l'accertamento della preparazione pratica (guida) e dell'idoneità.

Prenotazione Esame Teorico	Consiste nel registrare e riservare un posto per permettere il sostenimento dell'esame teorico al cliente.
Prenotazione Guida	Consiste nel registrare e riservare un posto per permettere il sostenimento di una lezione di guida al cliente.
Prenotazione Esame Finale	Consiste nel registrare e riservare un posto per permettere il sostenimento dell'esame finale al cliente

2. Analisi Orientata agli oggetti

2.1 Introduzione

Conclusa la fase di ideazione, si passa alla fase di elaborazione. Scopo delle iterazioni seguenti sarà quello di andare a sviluppare il software implementando gli altri casi d'uso individuati nel nostro *"modello dei casi d'uso"* tenendo conto anche delle *regole di dominio* da rispettare.

Seguendo l'approccio iterativo evolutivo suggerito da UP, la realizzazione dell'applicazione è stata articolata su 5 iterazioni.

Per ciascuna iterazione in particolare ci si è occupati di gestire le seguenti problematiche:

- **Iterazione 1**

Durante la prima iterazione i requisiti scelti su cui concentrarsi sono i seguenti:

- Implementare lo scenario principale di successo e tutte le estensioni finora individuate del caso d'uso **UC1**: ***Gestisci cliente (CRUD)***;
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate del caso d'uso **UC5**: ***Gestisci programmazione lezioni (CRUD);***
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate del caso d'uso **UC6:** ***Aggiorna frequenza clienti.***

- **Iterazione 2**

Durante questa seconda iterazione ci si concentrerà su:

- Implementare lo scenario principale di successo e tutte le estensioni finora individuate del caso d'uso **UC2**: ***Gestisci programmazione esame (CRUD);***
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate del caso d'uso **UC3**: **Prenota esame teorico;**

- **Iterazione 3**

Durante questa terza iterazione ci si concentrerà su:

- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC10**: ***Pubblica esiti esame teorico***.

- **Iterazione 4**

Durante questa quarta iterazione ci si concentrerà su:

- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC7**: ***Gestisci programmazione guide (CRUD);***
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC8**: ***Prenota guida;***
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC9**: ***Aggiorna numero guide sostenute;***

Relativamente ai casi d'uso in esame **(UC7, UC8, UC9)**, nasce l'esigenza di creare una nuova classe "**Guida**" e, poiché abbiamo riscontrato caratteristiche comuni tra **"Guida"** ed **"EsameTeorico"**, abbiamo deciso di creare una generalizzazione attraverso una nuova classe astratta che prenderà il nome di **"Attività"**. Inoltre, è stato aggiunto anche l'attributo *numeroGuide* in **Cliente.**

- **Iterazione 5**

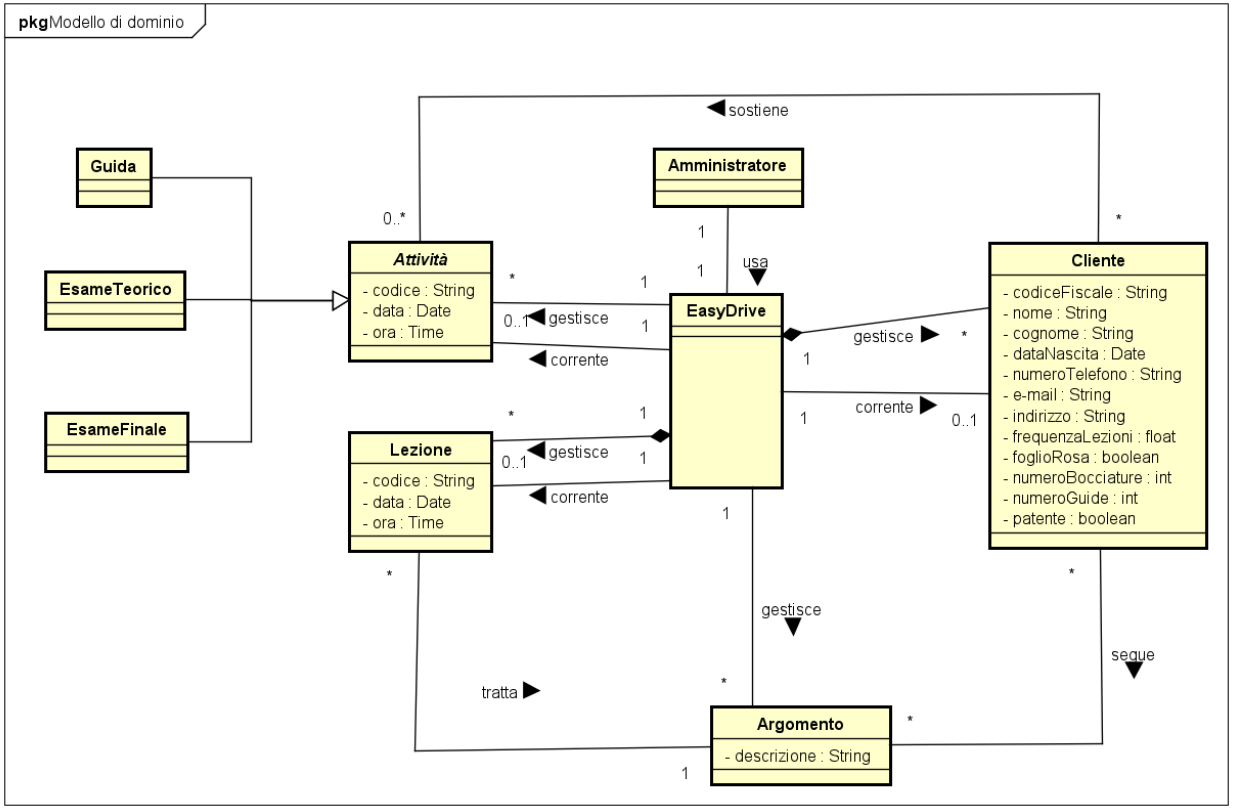
Durante questa quarta iterazione ci si concentrerà su:

- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC12: *Gestisci programmazione esame finale (CRUD);***
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC4: *Prenota esame finale;***
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC11: *Pubblica esiti esame finale;***

L’analisi orientata agli oggetti si basa sulla creazione di una descrizione del dominio da un punto di vista ad oggetti. Al fine di descrivere il dominio da un punto di vista ad oggetti e gestire ulteriori requisiti, saranno utilizzati nuovamente gli stessi strumenti dell’iterazione precedente (Modello di Dominio, SSD - Sequence System Diagram e Contratti delle operazioni). In particolare, i paragrafi seguenti permettono di evidenziare i cambiamenti che tali elaborati hanno subito rispetto alla fase precedente.

- Verranno utilizzati:
- Modello di Dominio *(paragrafo 2.2)*;
- Diagramma di sequenza di sistema [SSD] *(paragrafo 2.3)*;
- Contratti delle operazioni *(paragrafo 2.3)*;

2.2 Modello di Dominio



Come si nota, sono state identificate le seguenti classi concettuali:

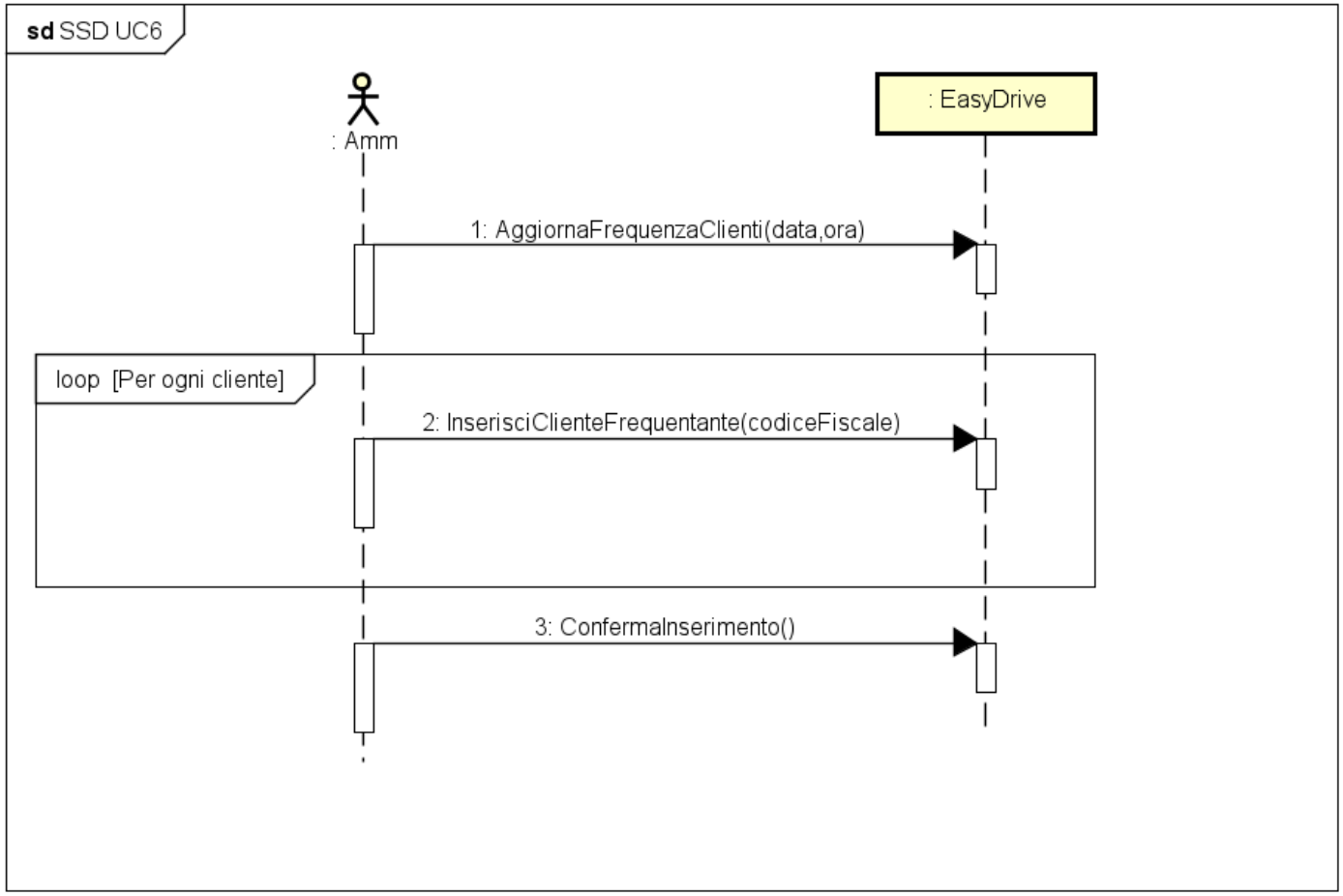
- **Amministratore:** attore primario, interagisce direttamente con il sistema;
- **EasyDrive:** rappresenta il sistema “EasyDrive”;
- **Cliente:** cliente della scuola guida che sostiene un’attività;
- **Lezione**: lezione corrente in cui si stanno prendendo le frequenze;
- **Argomento:** argomento trattato durante la lezione corrente;
- **EsameTeorico:** per poter registrare nel sistema gli esami teorici e permettere ai clienti di sostenerli;
- **Guida:** per poter registrare nel sistema le guide e permettere ai clienti di sostenerle;
- **EsameFinale:** per poter registrare nel sistema gli esami finali e permettere ai clienti di sostenerli;
- **Attività:** classe astratta che generalizza EsameTeorico, Guida ed EsameFinale.

2.3 SSD e Contratti

Procedendo con l'analisi Orienta agli Oggetti, il passo successivo è la creazione dei Diagramma di Sequenza di Sistema (SSD) al fine di illustrare il corso degli eventi di input e di output per i vari casi d'uso esaminati in ciascuna iterazione. Inoltre, le principali operazioni di sistema individuate negli SSD verranno descritte attraverso i Contratti, quindi avremo:

- **Iterazione 1**

Per i casi d'uso **UC1** e **UC5 (CRUD)** non sono stati creati gli SSD.



Vengono ora descritte attraverso i Contratti le principali operazioni di sistema che si occupano di gestire gli eventi di sistema individuati nell'SSD.

Contratto CO1: aggiornaFrequenzaClienti

Operazione: aggiornaFrequenzaClienti(data,ora);

Riferimenti: caso d'uso: Aggiorna frequenza clienti;

Pre-condizioni:

Post-Condizioni: - È stata recuperata l'istanza l di Lezione sulla base di dataOra;

- l è stata associata a EasyDrive tramite l'associazione "corrente";

Contratto CO2: inserisciClienteFrequentante

Operazione: inserisciClienteFrequentante(codiceFiscale)

Riferimenti: caso d'uso: Aggiorna frequenza clienti;

Pre-condizioni: - È in corso l'aggiornamento dell'attributo frequenzaLezioni del cliente c;

PostCondizioni: - È stata recuperata l'istanza c di Cliente sulla base di codiceFiscale;

- c è stata associata a EasyDrive tramite l'associazione "corrente";

Contratto CO3: confermaInserimento

Operazione: confermaInserimento()

Riferimenti: caso d'uso: Aggiorna frequenza clienti;

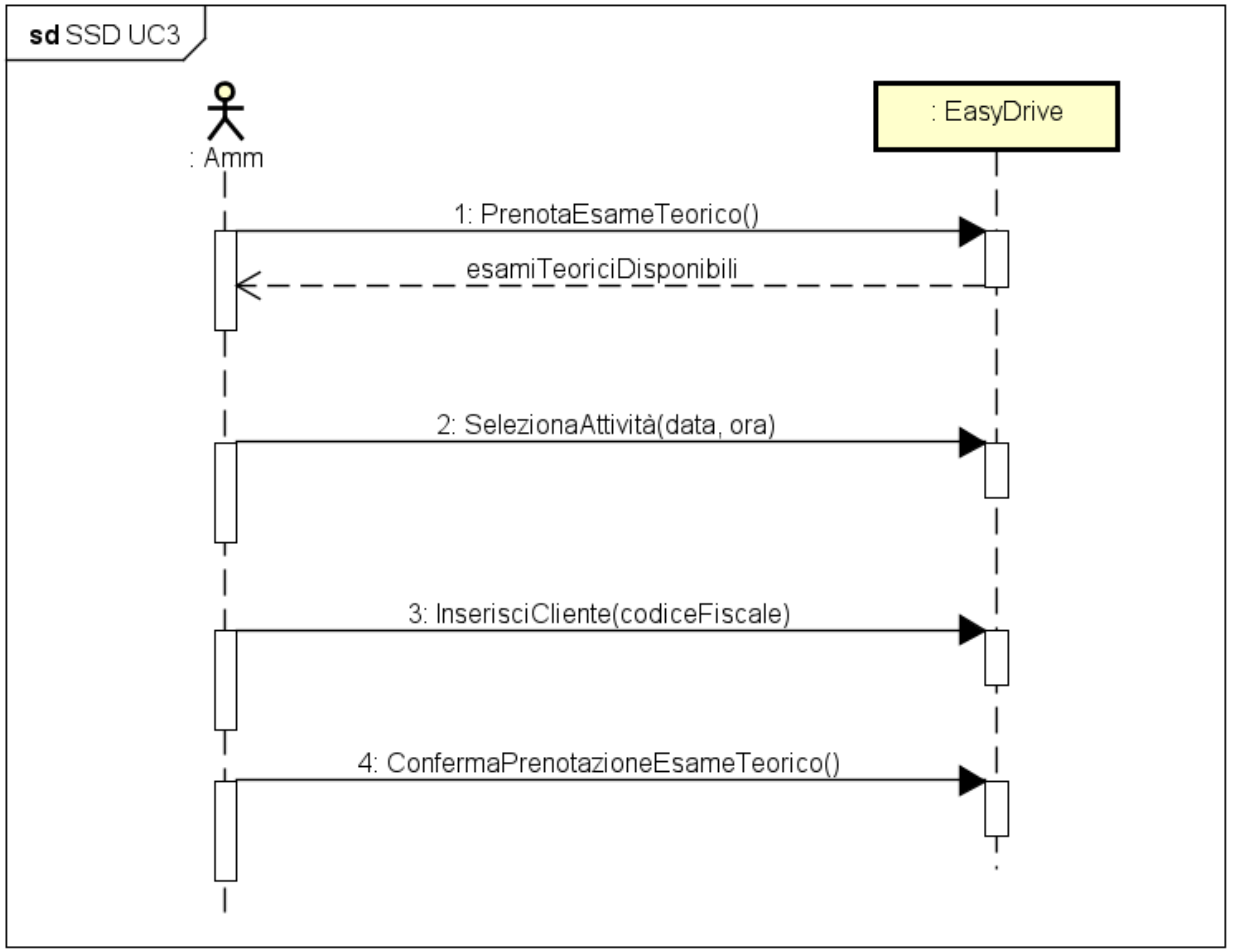
Pre-condizioni: - È in corso l'aggiornamento dell'attributo frequenzaLezioni del cliente c;

Post-Condizioni: - È stata recuperata l'istanza argomento di Argomento;

- È stata recuperata l'istanza numArgomentiTotali;
- L'istanza argomento è stata associata all'istanza c tramite l'associazione "segue";
- È stata recuperata l'istanza numArgomentiSeguiti;
- c.frequenzaLezioni è stato aggiornato.

- **Iterazione 2**

  Per il caso d'uso **UC2 (CRUD)** non è stato creato l'SSD.



Vengono ora descritte attraverso i Contratti le principali operazioni di sistema che si occupano di gestire gli eventi di sistema individuati nell'SSD.

Contratto CO1: prenotaEsameTeorico

Operazione: prenotaEsameTeorico();

Riferimenti: caso d'uso: Prenota Esame Teorico;

Pre-condizioni:

Post-Condizioni: - sono state recuperate le istanze "esame Teorico" sulla base della data e ora attuali;

Contratto CO2: selezionaEsame

Operazione: selezionaEsame(data, ora)

Riferimenti: caso d'uso: Prenota Esame Teorico;

Pre-condizioni: - È noto l'elenco degli esami teorici disponibili;

Post-Condizioni: - È stata recuperata l'istanza e di EsameTeorico sulla base di data e ora;

- "e" è stata associata a EasyDrive tramite l'associazione "corrente";

Contratto CO3: inserisciCliente

Operazione: inserisciCliente(codiceFiscale)

Riferimenti: caso d'uso: Prenota Esame Teorico

Pre-condizioni: - È in corso la prenotazione di un cliente ad un esame Teorico

Post-Condizioni: - È stata recuperata l'istanza c di Cliente sulla base di codiceFiscale;

- “c” è stata associata a EasyDrive tramite l’associazione “corrente”;

Contratto CO4: confermaPrenotazioneEsameTeorico

Operazione: confermaPrenotazioneEsameTeorico()

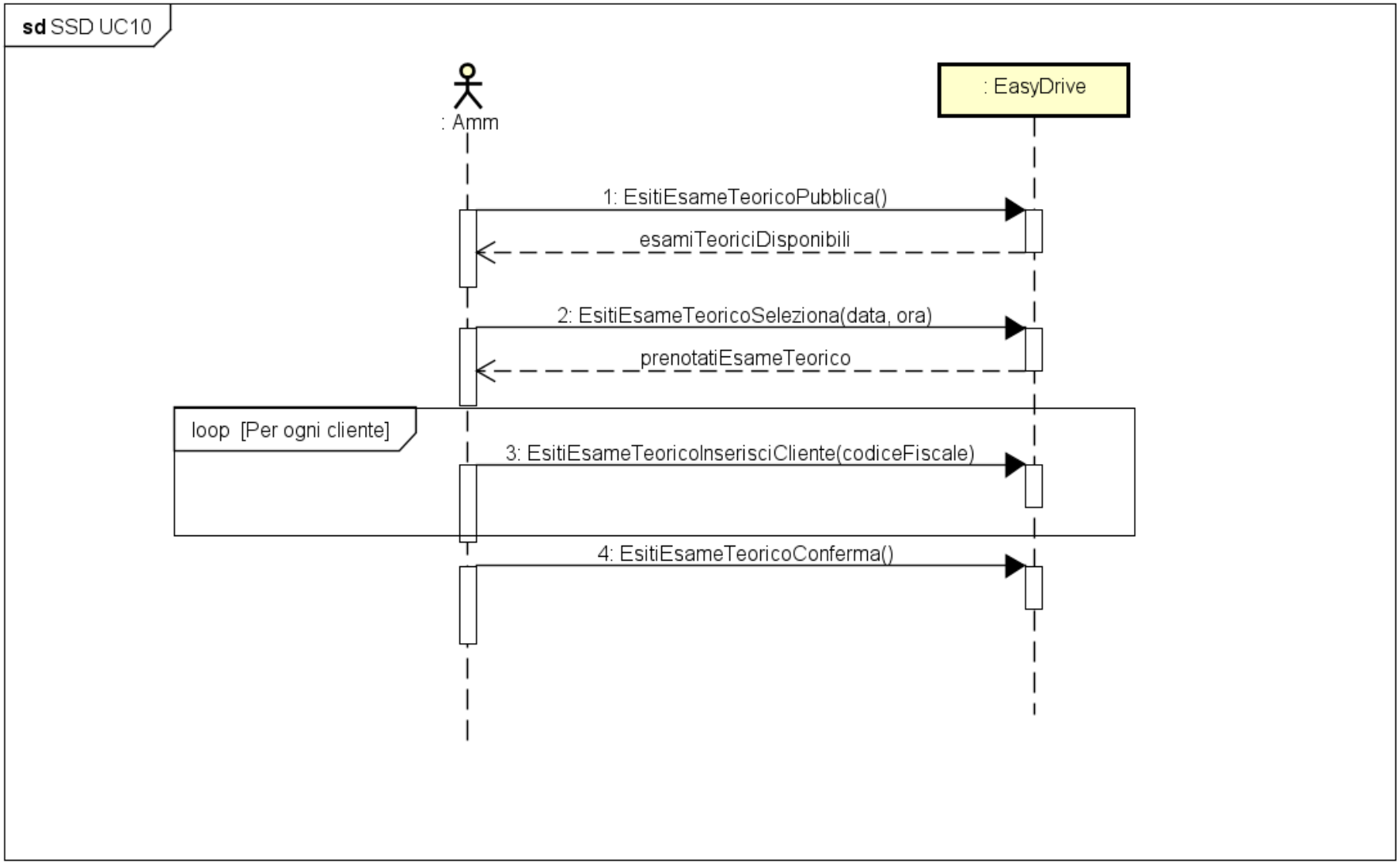
Riferimenti: caso d’uso: Prenota Esame Teorico

Pre-condizioni: - È in corso la prenotazione di un cliente ad un esame Teorico

Post-Condizioni: - È stato recuperato l’attributo frequenza di Cliente c;

- “c” è stata associata ad EsameTeorico tramite l’associazione “sostiene”

- **Iterazione 3**



Vengono ora descritte attraverso i Contratti le principali operazioni di sistema che si occupano di gestire gli eventi di sistema individuati nell'SSD.

Contratto CO1: esitiEsameTeoricoPubblica

Operazione: esitiEsameTeoricoPubblica();

Riferimenti: caso d'uso: Pubblica Esiti Esame Teorico;

Pre-condizioni:

Post-Condizioni: - sono state recuperate le istanze di "esame Teorico" svolte in passato sulla base della data e ora attuali;

Contratto CO2: esitiEsameTeoricoSeleziona

Operazione: esitiEsameTeoricoSeleziona(data, ora);

Riferimenti: caso d'uso: Pubblica Esiti Esame Teorico;

Pre-condizioni: - È noto l'elenco degli esami teorici svolti in passato;

Post-Condizioni: - È stata recuperata l'istanza e di EsameTeorico sulla base di data e ora;

- "e" è stata associata a EasyDrive tramite l'associazione "corrente";
- Sono state recuperate le istanze di "Cliente" che hanno partecipato all'esame "e".

Contratto CO3: esitiEsameTeoricoInserisciCliente

Operazione: esitiEsameTeoricoInserisciCliente(codiceFiscale);

Riferimenti: caso d'uso: Pubblica Esiti Esame Teorico;

Pre-condizioni: - È in corso la promozione o bocciatura di un cliente all'esame Teorico selezionato;

Post-Condizioni: - È stata recuperata l'istanza c di Cliente sulla base di codiceFiscale;

- È stato aggiornato l'attributo "foglioRosa" di c a "true";

Contratto CO4: esitiEsameTeoricoConferma

Operazione: esitiEsameTeoricoConferma ()

Riferimenti: caso d'uso: Pubblica Esiti Esame Teorico

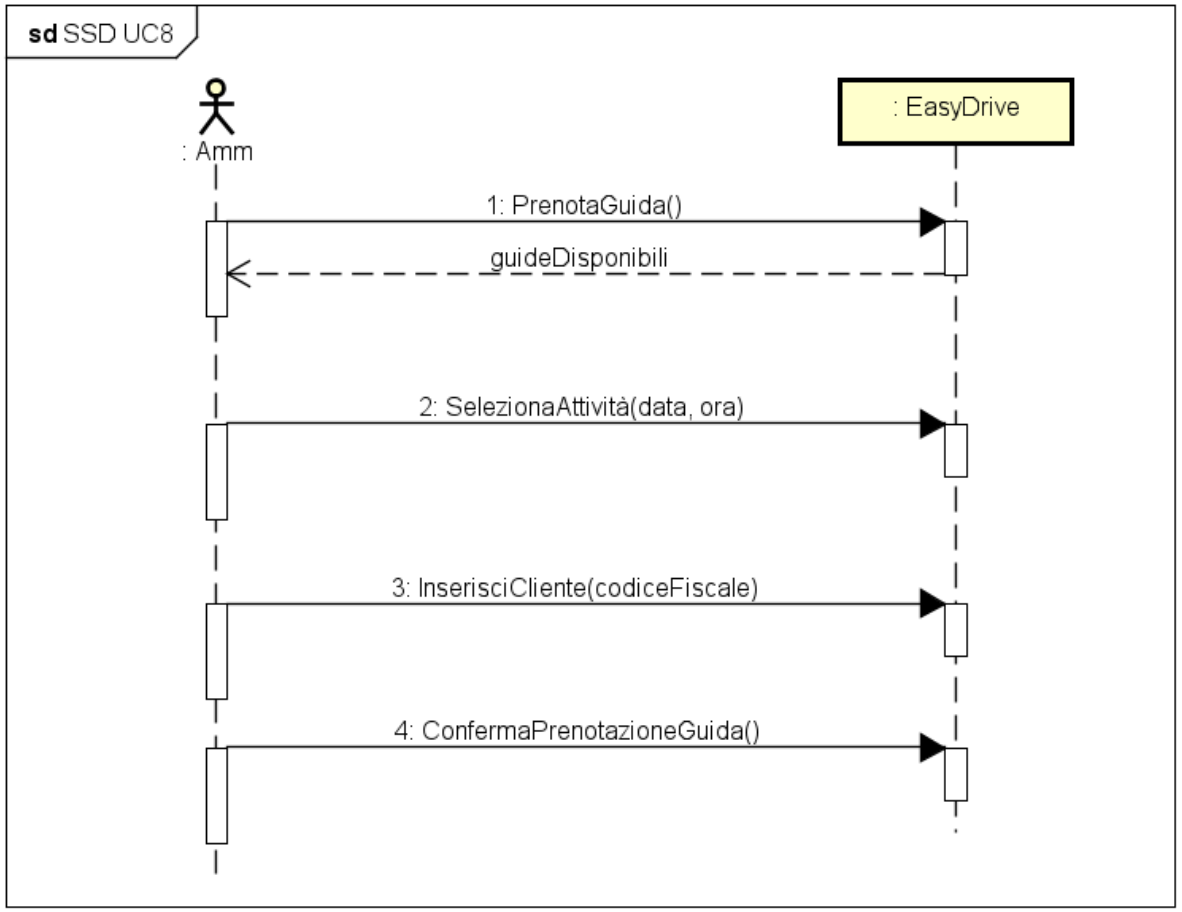
Pre-condizioni: - È in corso la promozione o bocciatura di un cliente all'esame Teorico selezionato;

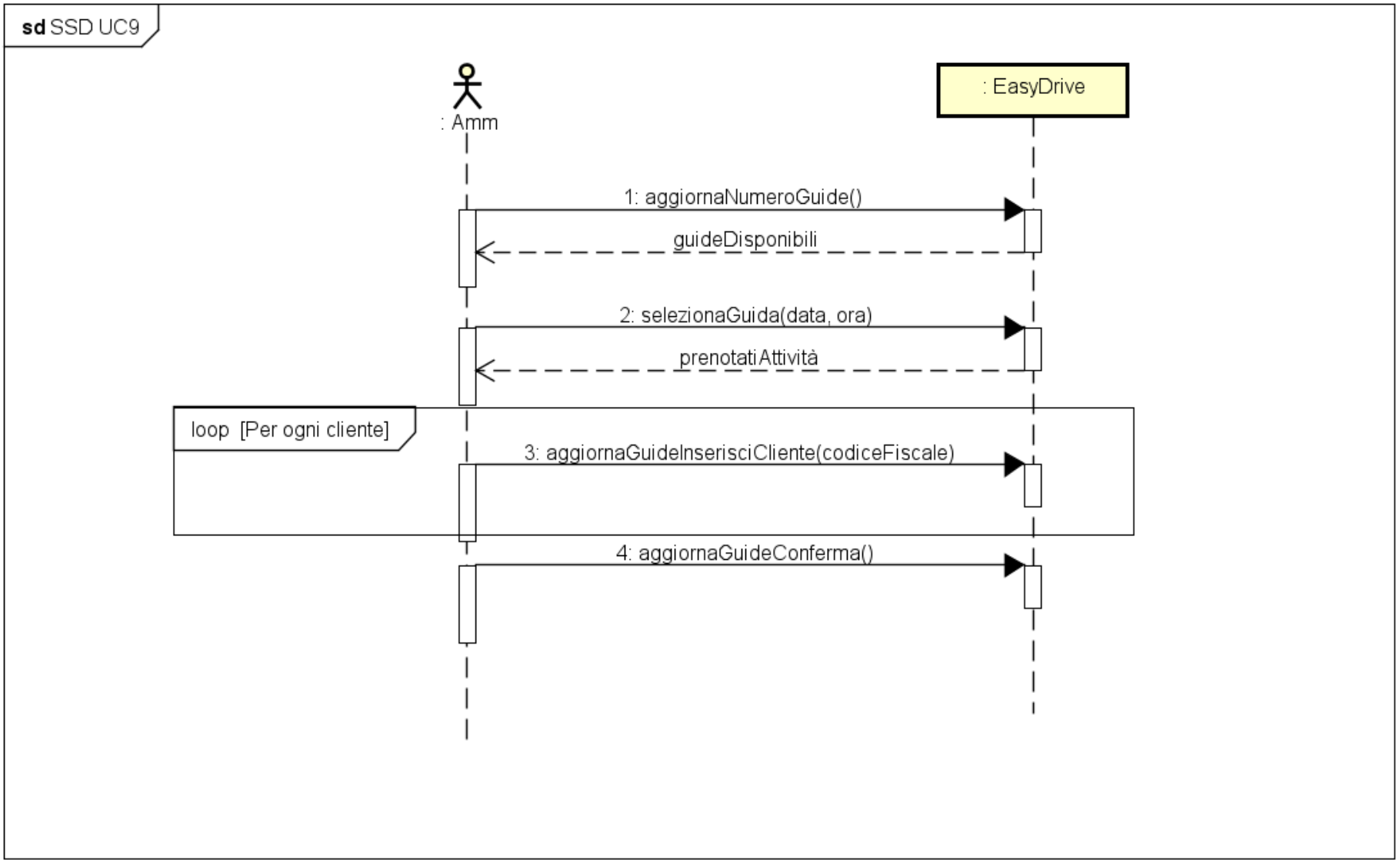
Post-Condizioni: - È stato recuperato l'attributo foglioRosa di Cliente c;

- È stato incrementato l'attributo numeroBocciature nel caso in cui l'attributo foglioRosa sia "false";
- "c" è stata associata ad EasyDrive tramite l'associazione "gestisce" nel caso in cui numeroBocciature sia maggiore o uguale a 2;

- **Iterazione 4**

  Per il caso d'uso **UC7 (CRUD) non è** stato creato l'SSD.




sd SSD UC9
: Amm
: EasyDrive
1: aggiornaNumeroGuide()
guideDisponibili
2: selezionaGuida(data, ora)
prenotatiAttività
loop [Per ogni cliente]
3: aggiornaGuideInserisciCliente(codiceFiscale)
4: aggiornaGuideConferma()

Vengono ora descritte attraverso i Contratti le principali operazioni di sistema che si occupano di gestire gli eventi di sistema individuati nell'SSD.

UC8:

Contratto CO1: prenotaGuida

Operazione: prenotaGuida();

Riferimenti: caso d'uso: Prenota Guida;

Pre-condizioni:

Post-Condizioni: - sono state recuperate le istanze di "Guida" sulla base della data e ora attuali;

Contratto CO2: selezionaAttività

Operazione: selezionaAttività(data, ora);

Riferimenti: caso d'uso: Prenota Guida;

Pre-condizioni: - È noto l'elenco delle guide disponibili;

Post-Condizioni: - È stata recuperata l'istanza a di Guida sulla base di data e ora;

- "a" è stata associata a EasyDrive tramite l'associazione "corrente";

Contratto CO3: InserisciCliente

Operazione: InserisciCliente(codiceFiscale);

Riferimenti: caso d'uso: Prenota Guida;

Pre-condizioni: - È in corso la prenotazione di un cliente ad una guida;

Post-Condizioni: - È stata recuperata l'istanza c di Cliente sulla base di codiceFiscale;

- "c" è stata associata a EasyDrive tramite l'associazione "corrente";

Contratto CO4: confermaPrenotazioneGuida

Operazione: confermaPrenotazioneGuida()

Riferimenti: caso d'uso: Prenota Guida;

Pre-condizioni: - È in corso l'aggiornamento del numero di guide di un cliente;

Post-Condizioni: - È stato recuperato l'attributo foglioRosa di Cliente c;

- "c" è stata associata ad Attività tramite l'associazione "sostiene" nel caso in cui l'attributo "foglioRosa" sia "true"

UC9:

Contratto CO1: aggiornaNumeroGuide

Operazione: aggiornaNumeroGuide();

Riferimenti: caso d'uso: Aggiorna Numero Guide Sostenute;

Pre-condizioni:

Post-Condizioni: - sono state recuperate le istanze di "Guida" svolte in passato sulla base della data e ora attuali;

Contratto CO2: selezionaGuida

Operazione: selezionaGuida(data, ora);

Riferimenti: caso d'uso: Aggiorna Numero Guide Sostenute;

Pre-condizioni: - È noto l'elenco delle guide svolte in passato;

Post-Condizioni: - È stata recuperata l'istanza "a" di Guida sulla base di data e ora;

- "a" è stata associata a EasyDrive tramite l'associazione "corrente";
- Sono state recuperate le istanze di "Cliente" che hanno partecipato alla guida "a".

Contratto CO3: aggiornaGuideInserisciCliente

Operazione: aggiornaGuideInserisciCliente(codiceFiscale);

Riferimenti: caso d'uso: Aggiorna Numero Guide Sostenute;

Pre-condizioni: - È in corso l'aggiornamento del numero di guide di un cliente;

Post-Condizioni: - È stata recuperata l'istanza c di Cliente sulla base di codiceFiscale;

- “c” è stata associata a EasyDrive tramite l’associazione “corrente”;

Contratto CO4: aggiornaGuideConferma

Operazione: aggiornaGuideConferma()

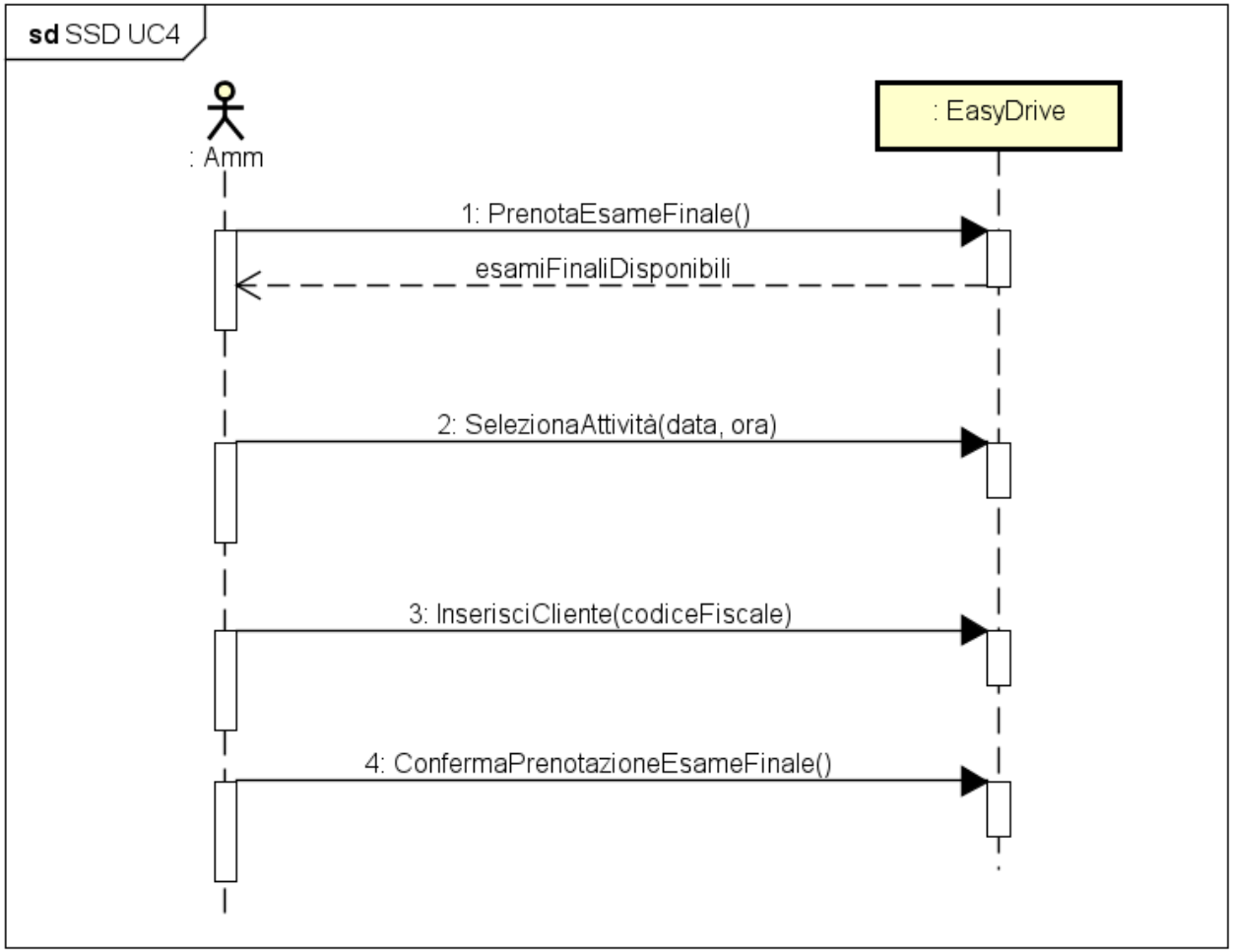
Riferimenti: caso d’uso: Aggiorna Numero Guide Sostenute;

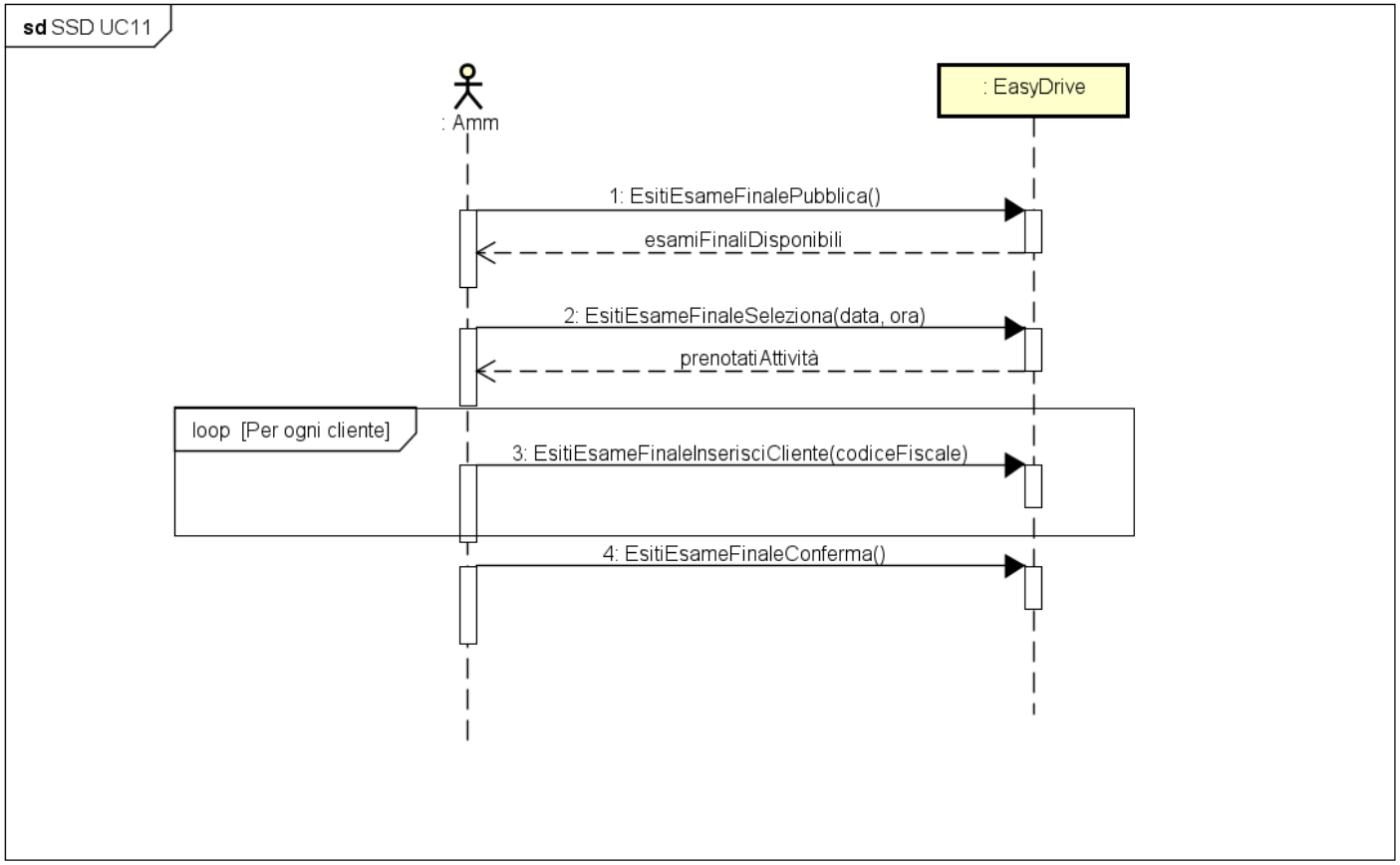
Pre-condizioni: - È in corso l’aggiornamento del numero di guide di un cliente;

PostCondizioni: - È stato incrementato l’attributo numeroGuide del cliente c;

- **Iterazione 5**

Per il caso d'uso UC12 (CRUD) **non è** stato creato l'SSD.





Vengono ora descritte attraverso i Contratti le principali operazioni di sistema che si occupano di gestire gli eventi di sistema individuati nell'SSD.

UC4:

Contratto CO1: prenotaEsameFinale

Operazione: prenotaEsameFinale();

Riferimenti: caso d'uso: Prenota Esame Finale;

Pre-condizioni:

Post-Condizioni: - sono state recuperate le istanze "esame Finale" sulla base della data e ora attuali;

Contratto CO2: selezionaAttività

Operazione: selezionaAttività(data, ora)

Riferimenti: caso d'uso: Prenota Esame Finale;

Pre-condizioni: - È noto l'elenco degli esami finali disponibili;

Post-Condizioni: - - È stata recuperata l'istanza a di EsameFinale sulla base di data e ora;

- "a" è stata associata a EasyDrive tramite l'associazione "corrente";

Contratto CO3: inserisciCliente

Operazione: inserisciCliente(codiceFiscale)

Riferimenti: caso d'uso: Prenota Esame Finale;

Pre-condizioni: - È in corso la prenotazione di un cliente ad un esame Finale

Post-Condizioni: - È stata recuperata l'istanza c di Cliente sulla base di codiceFiscale;

- "c" è stata associata a EasyDrive tramite l'associazione "corrente";

Contratto CO4: confermaPrenotazioneEsameFinale

Operazione: confermaPrenotazioneEsameFinale()

Riferimenti: caso d'uso: Prenota Esame Finale;

Pre-condizioni: - È in corso la prenotazione di un cliente ad un esame Finale;

Post-Condizioni: - È stato recuperato l'attributo numeroGuude di Cliente c;

- "c" è stata associata a EsameFinale tramite l'associazione "sostiene"

UC11:

Contratto CO1: esitiEsameFinalePubblica

Operazione: esitiEsameFinalePubblica();

Riferimenti: caso d'uso: Pubblica Esiti Esame Finale;

Pre-condizioni:

Post-Condizioni: - sono state recuperate le istanze di "esame Finale" svolte in passato sulla base della data e ora attuali;

Contratto CO2: esitiEsameFinaleSeleziona

Operazione: esitiEsameFinaleSeleziona(data, ora);

Riferimenti: caso d'uso: Pubblica Esiti Esame Finale;

Pre-condizioni: - È noto l'elenco degli esami finali svolti in passato;

Post-Condizioni: - - È stata recuperata l'istanza a di EsameFinale sulla base di data e ora;

- “a” è stata associata a EasyDrive tramite l’associazione “corrente”;

- Sono state recuperate le istanze di “Cliente” che hanno partecipato all’esame “a”.

Contratto CO3: esitiEsameFinaleInserisciCliente

Operazione: esitiEsameFinaleInserisciCliente(codiceFiscale);

Riferimenti: caso d’uso: Pubblica Esiti Esame Finale;

Pre-condizioni: - È in corso la promozione di un cliente all’esame Finaleselezionato;

Post-Condizioni: - È stata recuperata l’istanza c di Cliente sulla base di codiceFiscale;

- È stato aggiornato l’attributo “patente” di c a “true”;

Contratto CO4: esitiEsameFinaleConferma

Operazione: esitiEsameFinaleConferma ()

Riferimenti: caso d’uso: Pubblica Esiti Esame Finale;

Pre-condizioni: - È in corso la promozione o bocciatura di un cliente all’esame Finale selezionato;

Post-Condizioni: - È stato recuperato l’attributo patente di Cliente c;

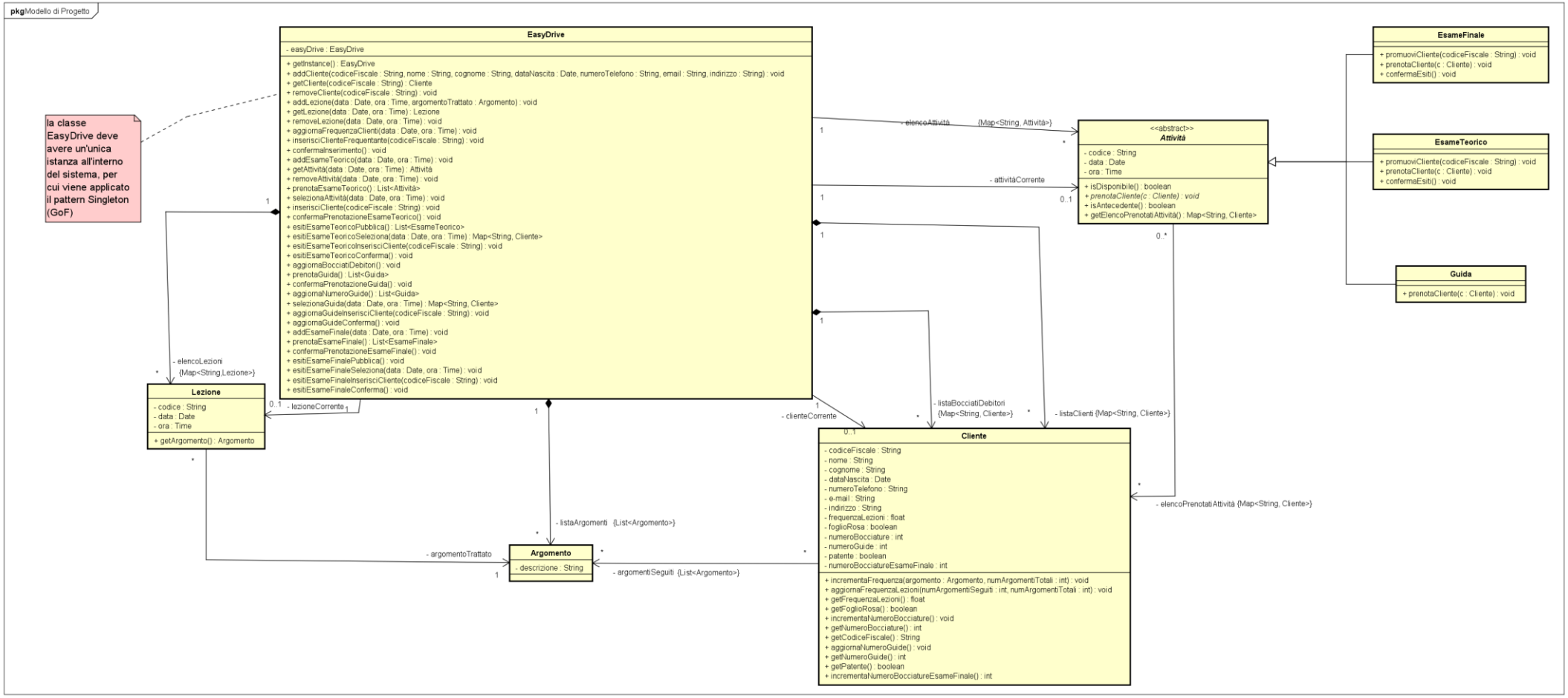
- È stato incrementato l’attributo numeroBocciatureEsameFinale nel caso in cui l’attributo patente sia “false”;

- È stato incrementato modifcato l’attributo foglioRosa a false nel caso in cui l’attributo numeroBocciatureEsameFinale sia >= 2;

3. Progettazione

3.1 Diagramma delle classi

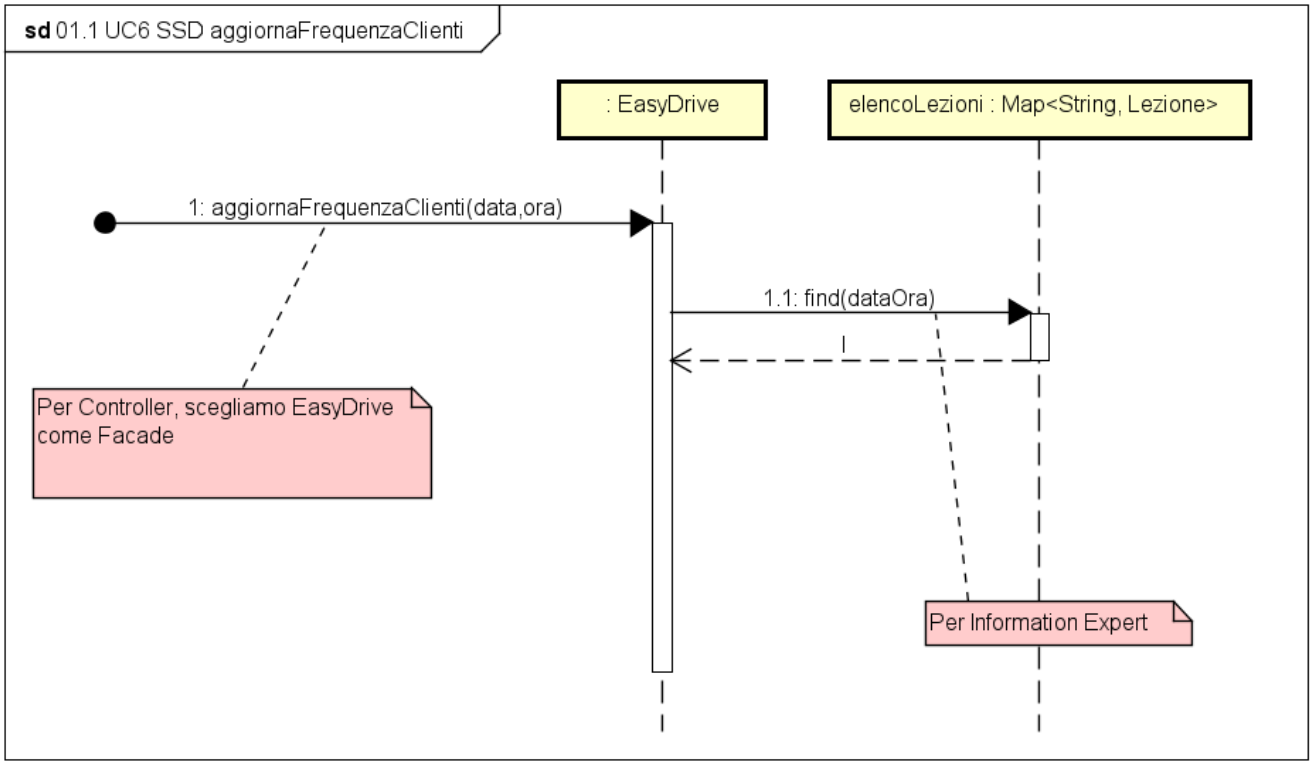
La progettazione orientata agli oggetti è la disciplina di UP interessata alla definizione degli oggetti software, delle loro responsabilità e a come questi collaborano per soddisfare i requisiti individuati nei passi precedenti. L'elaborato principale di questa fase che è stato preso in considerazione è il Modello di Progetto, ovvero l'insieme dei diagrammi che descrivono la progettazione logica sia da un punto di vista dinamico (Diagrammi di Interazione) che da un punto di vista statico (Diagramma delle Classi). Il diagramma delle Classi dunque sarà:



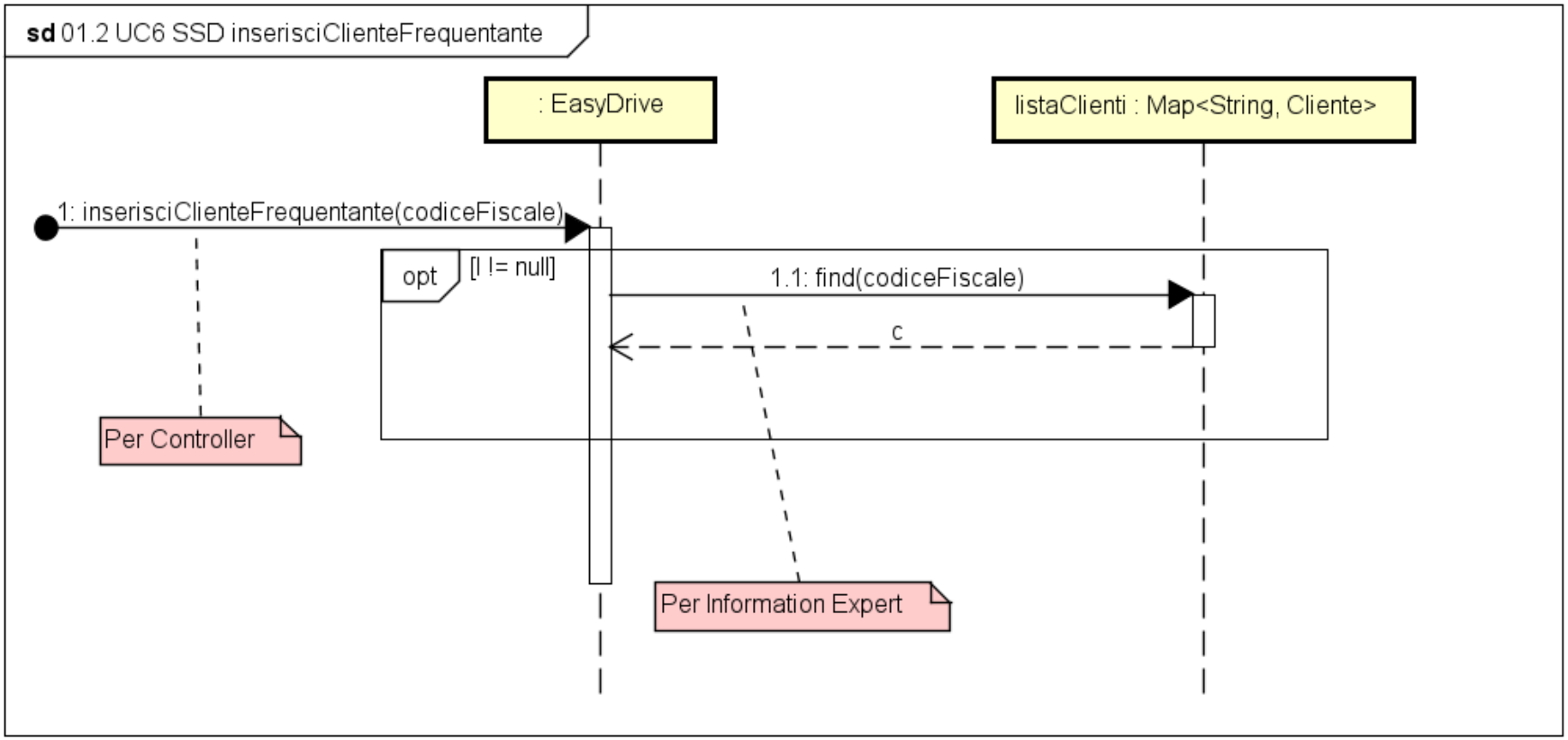
3.2 Diagrammi di sequenza

Vengono ora presentati i più importanti diagrammi di sequenza, per maggiori dettagli e/o per visionarli tutti si veda il documento sorgente creato con il software Astah Professional.

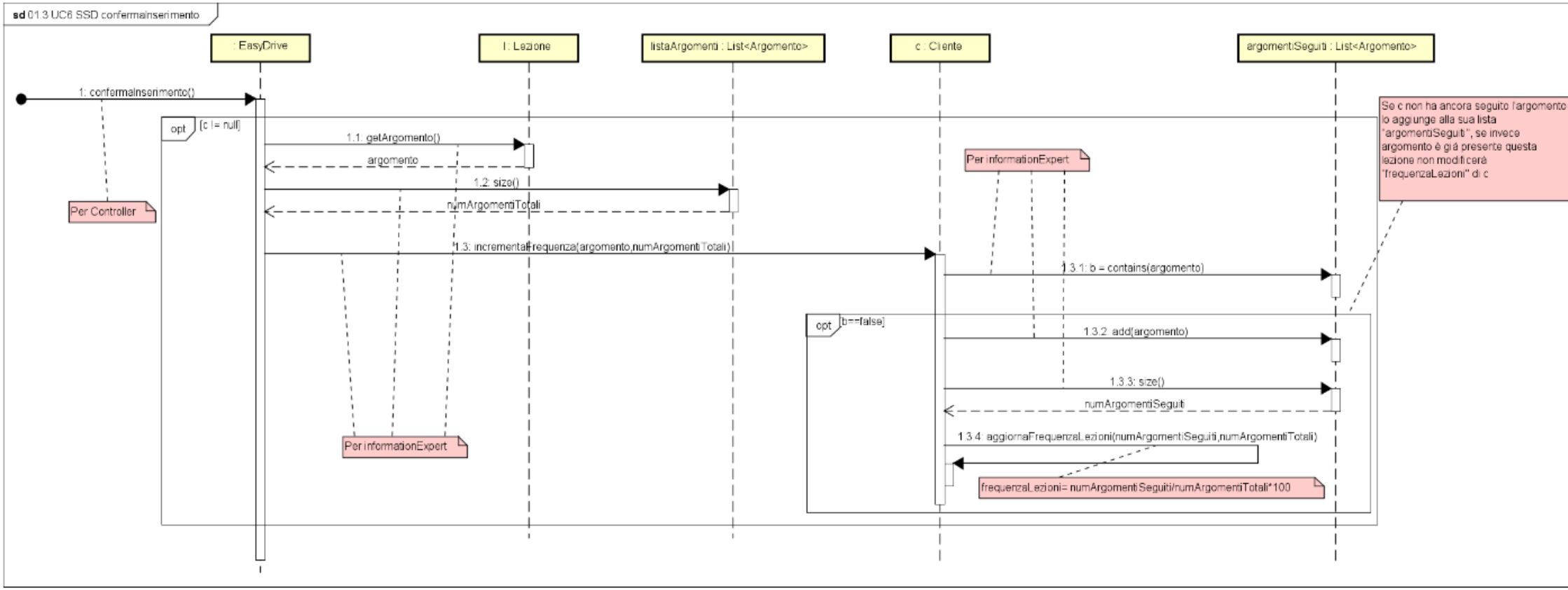
- **Aggiorna frequenza clienti**



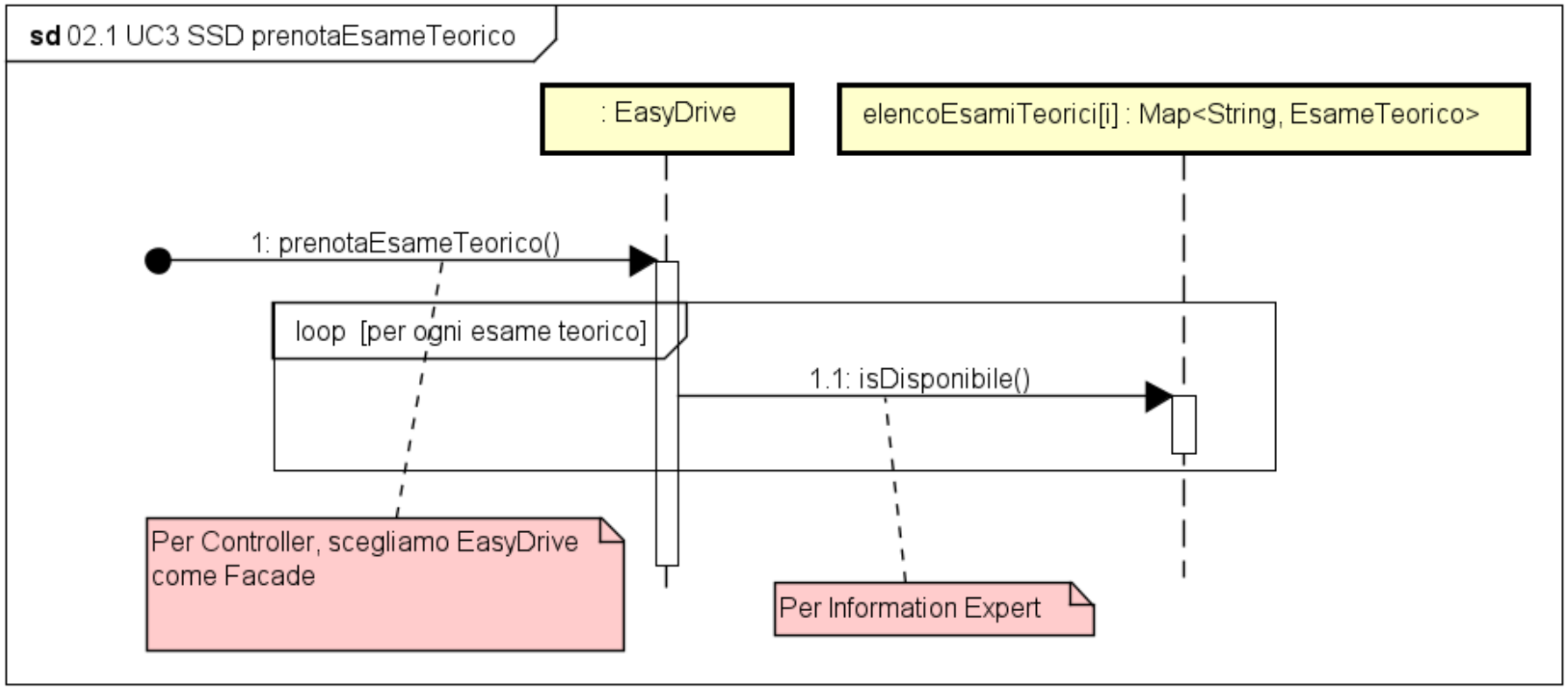
- **Inserisci cliente frequentante**



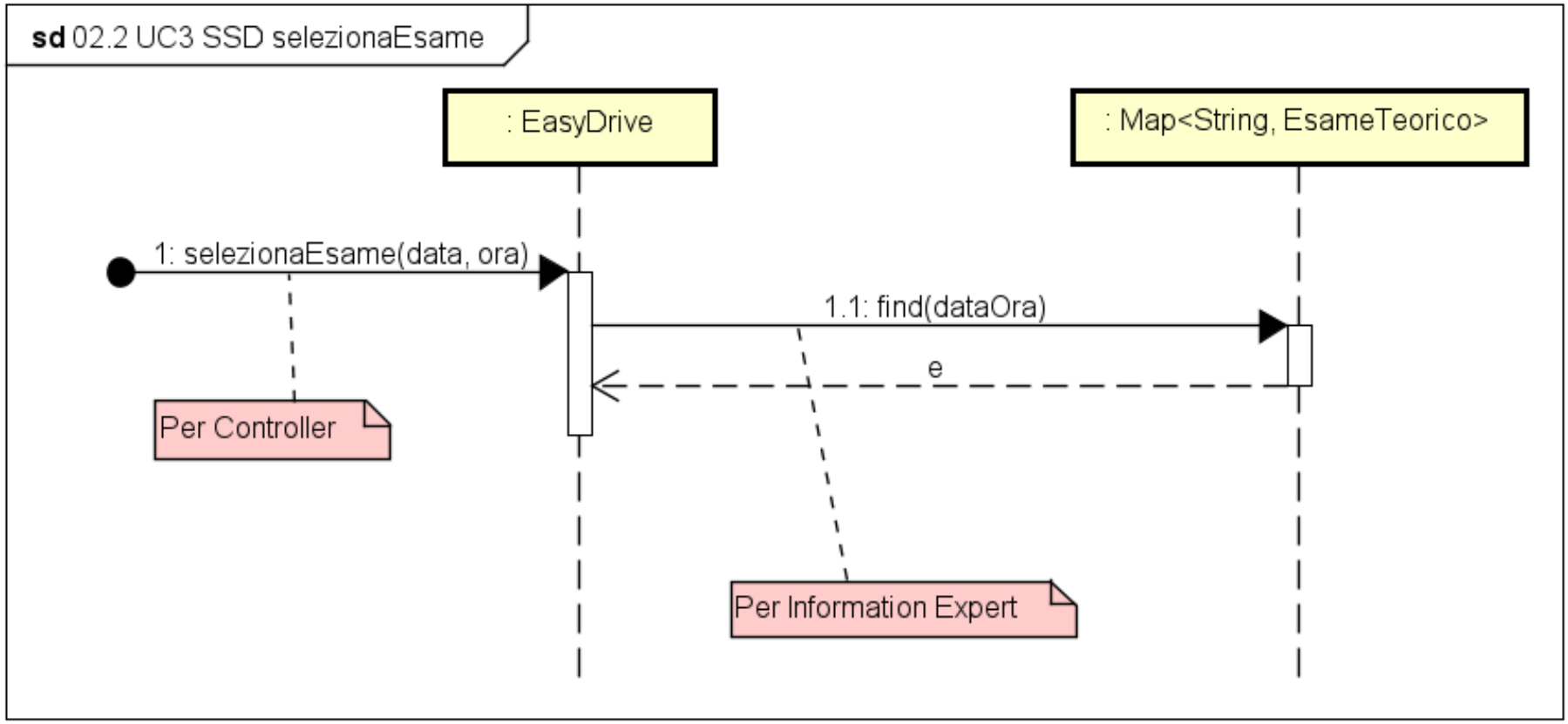
- **Conferma inserimento**



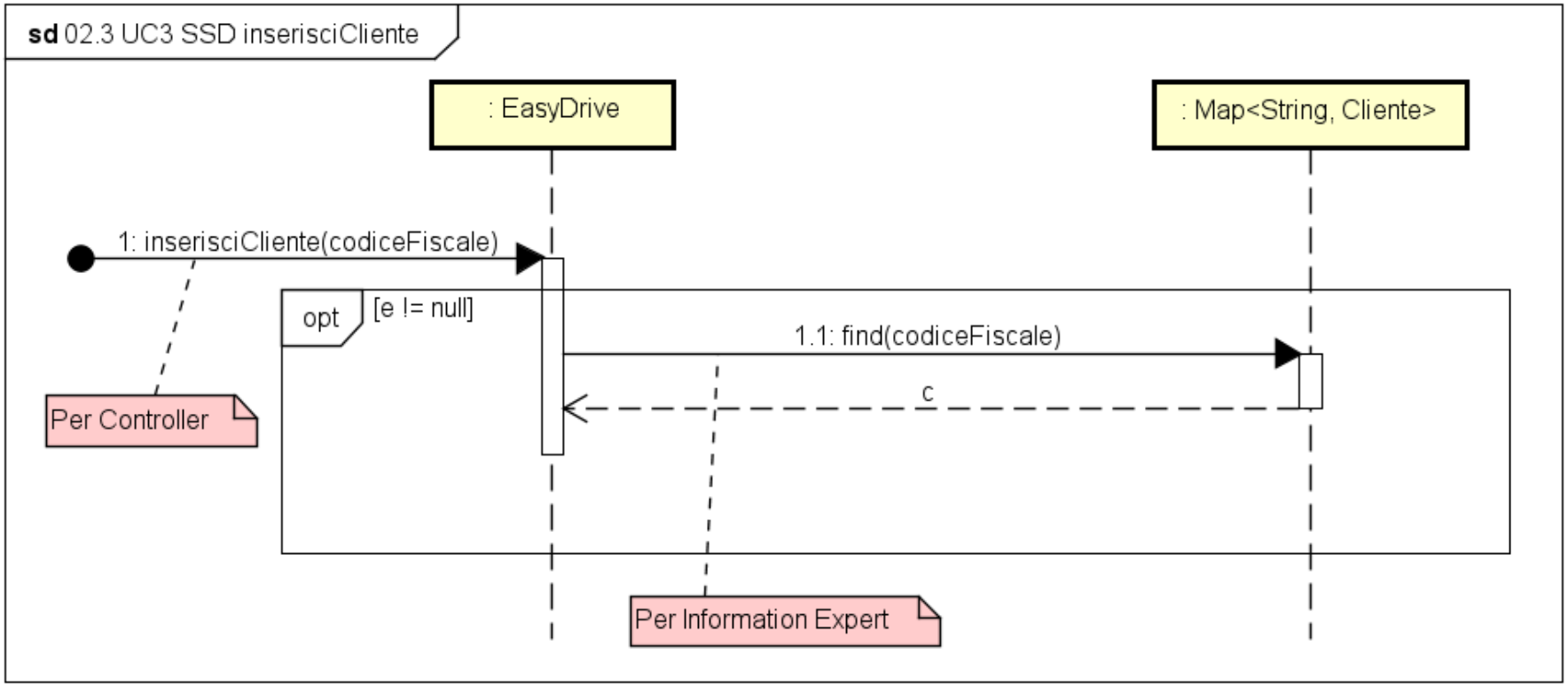
- **Prenota esame teorico**



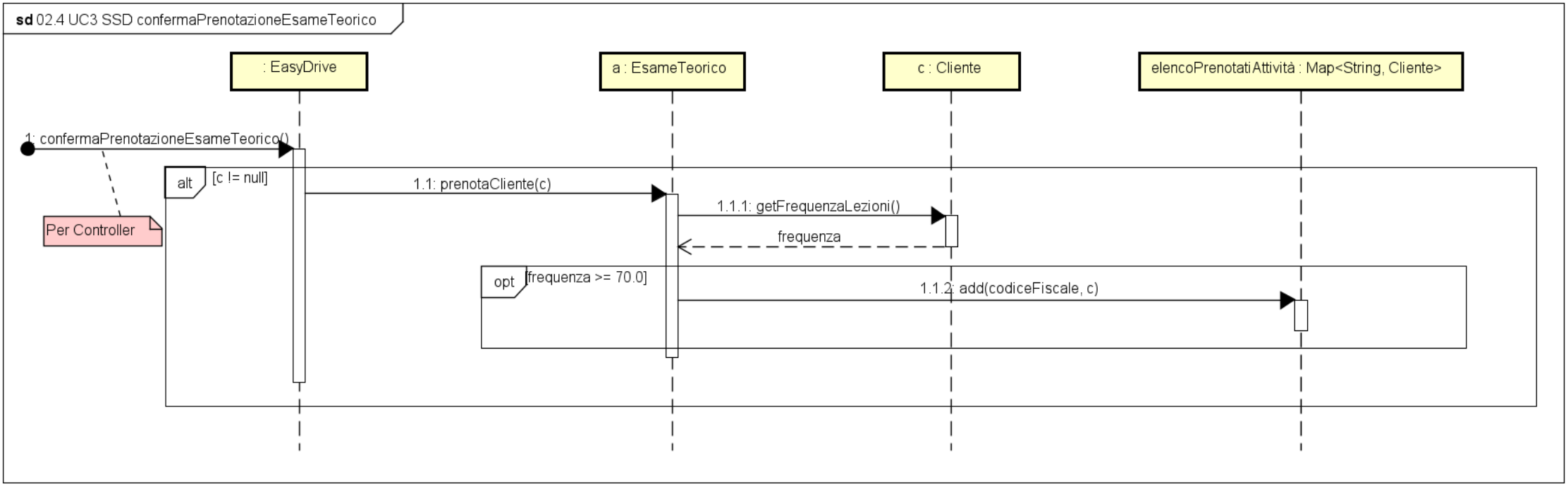
- **Seleziona esame**


sd 02.2 UC3 SSD selezionaEsame
: EasyDrive
: Map<String, EsameTeorico>
1: selezionaEsame(data, ora)
1.1: find(dataOra)
e
Per Controller
Per Information Expert

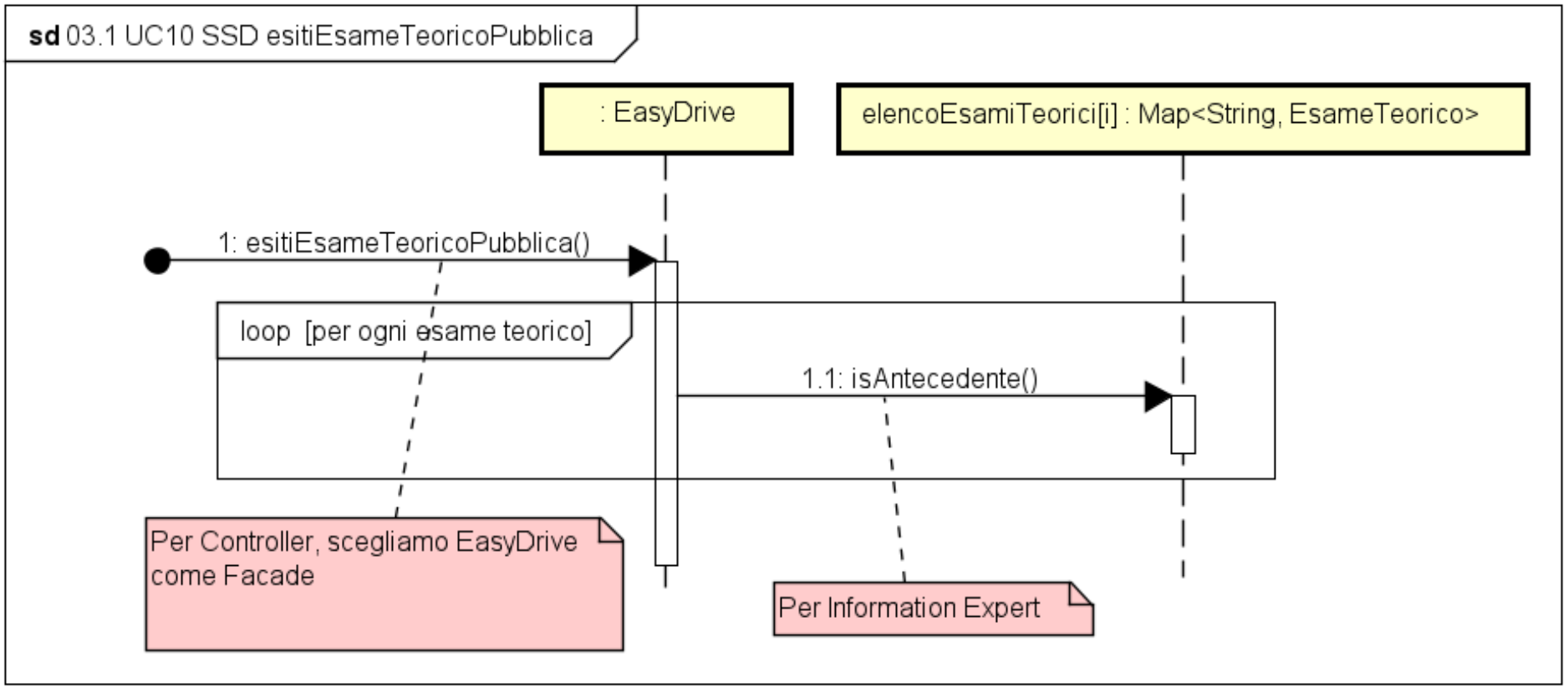
- **Inserisci cliente**



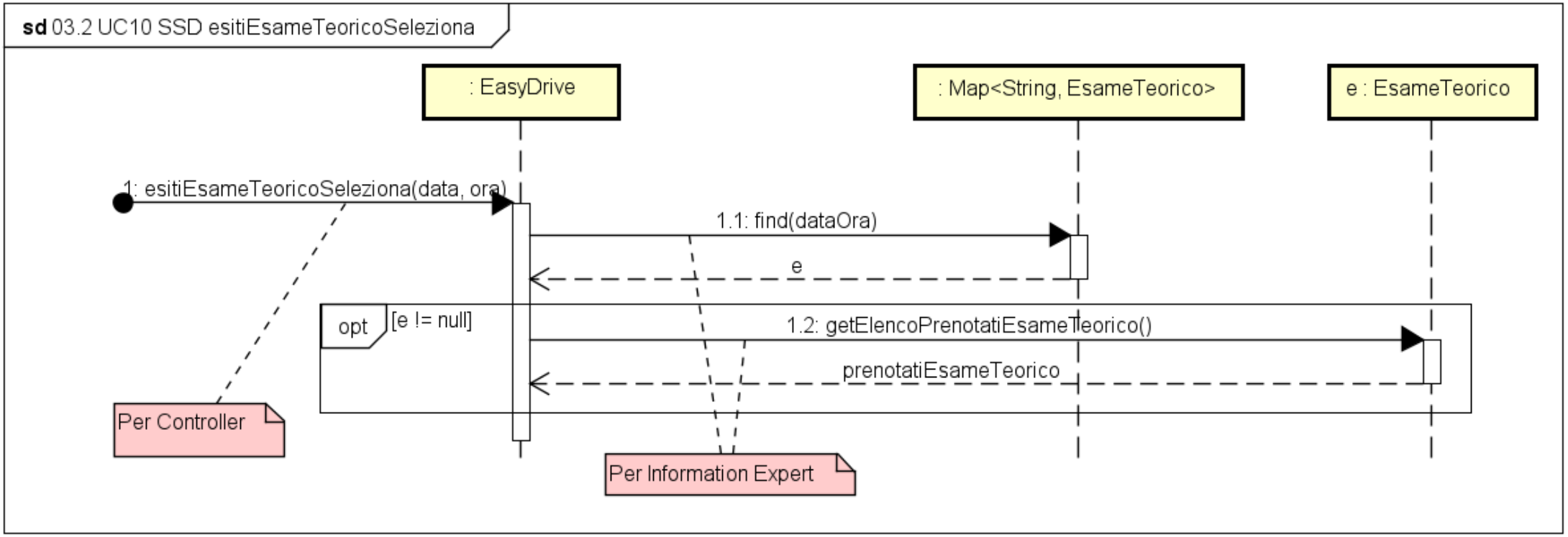
- **Conferma prenotazione Esame Teorico**



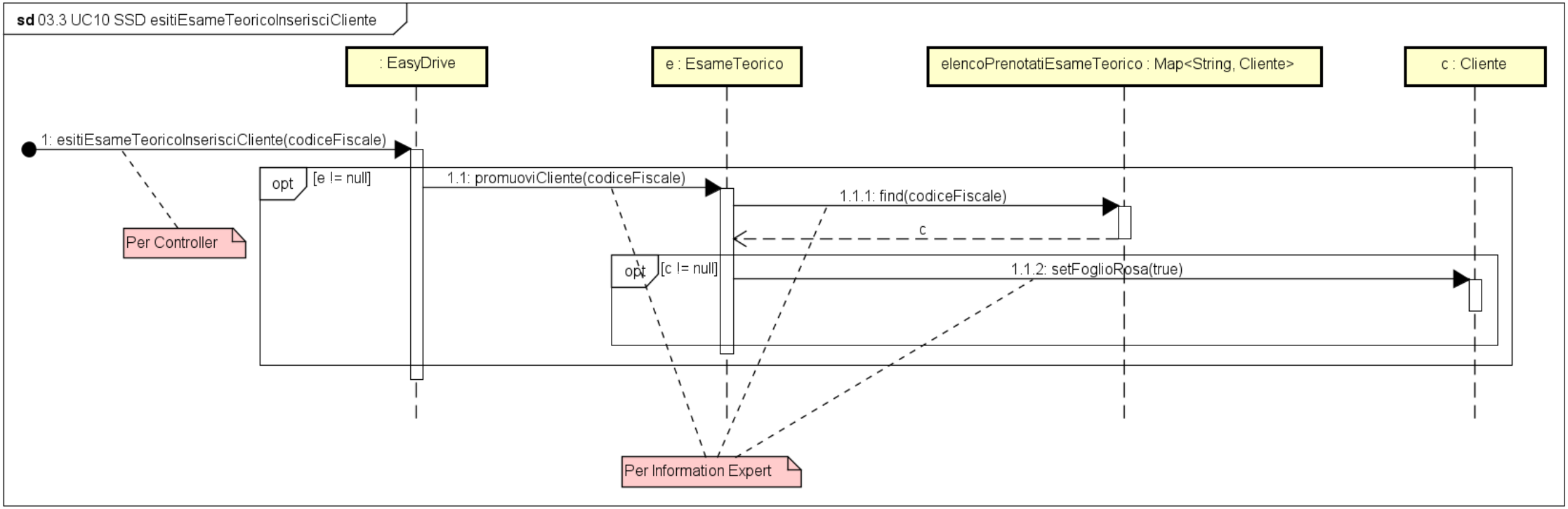
- **Esiti esame teorico pubblica**



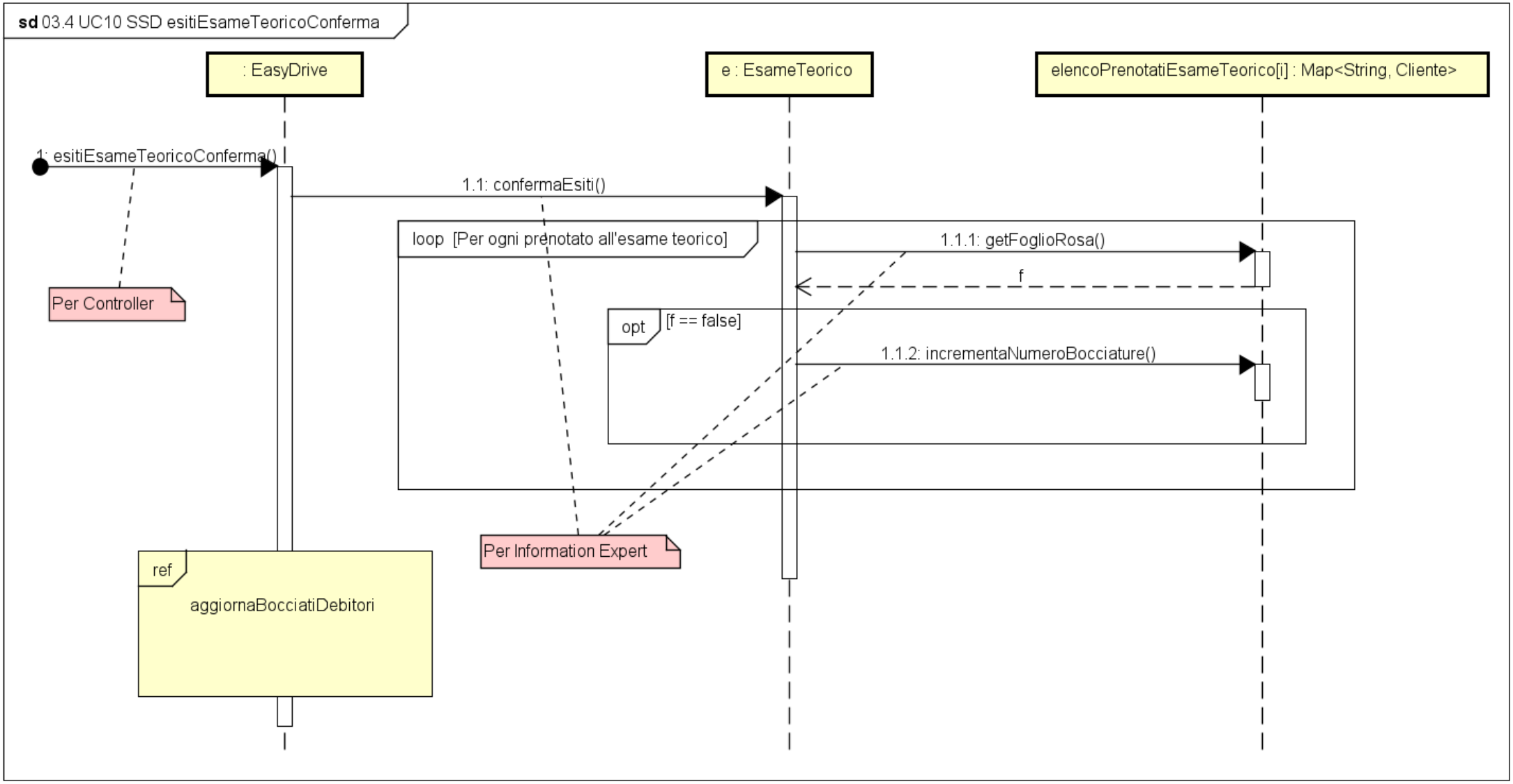
- **Esiti esame teorico seleziona**



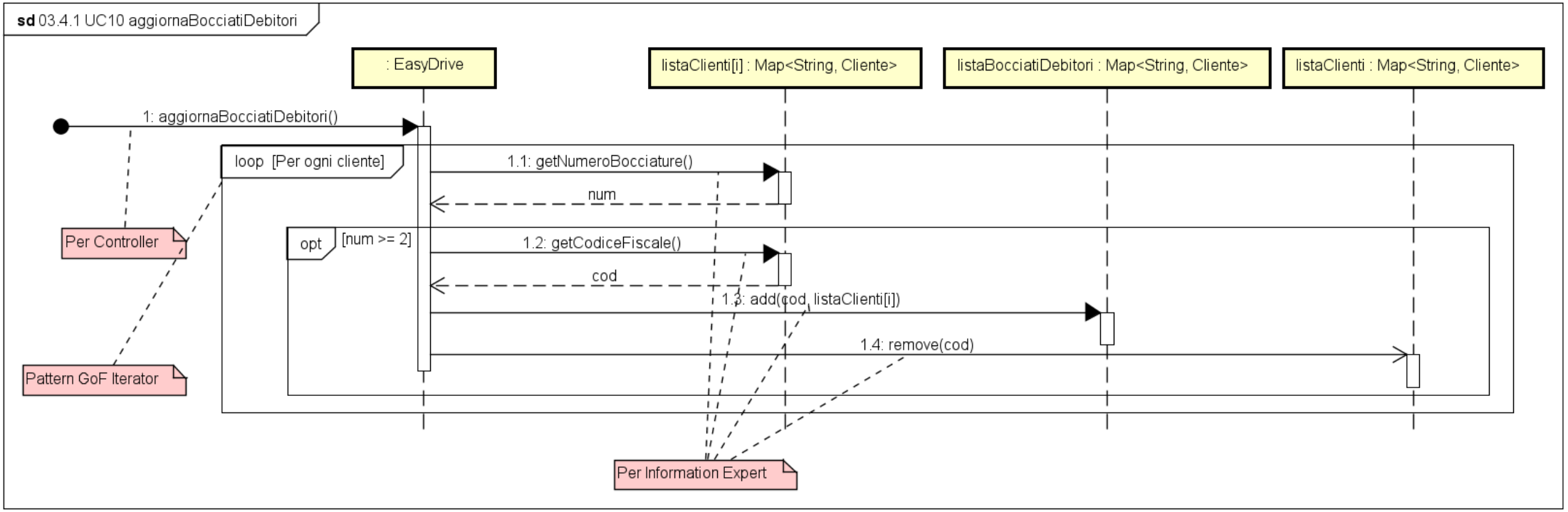
- **Esiti Esame teorico inserisci cliente**



- **Esiti esame teorico conferma**

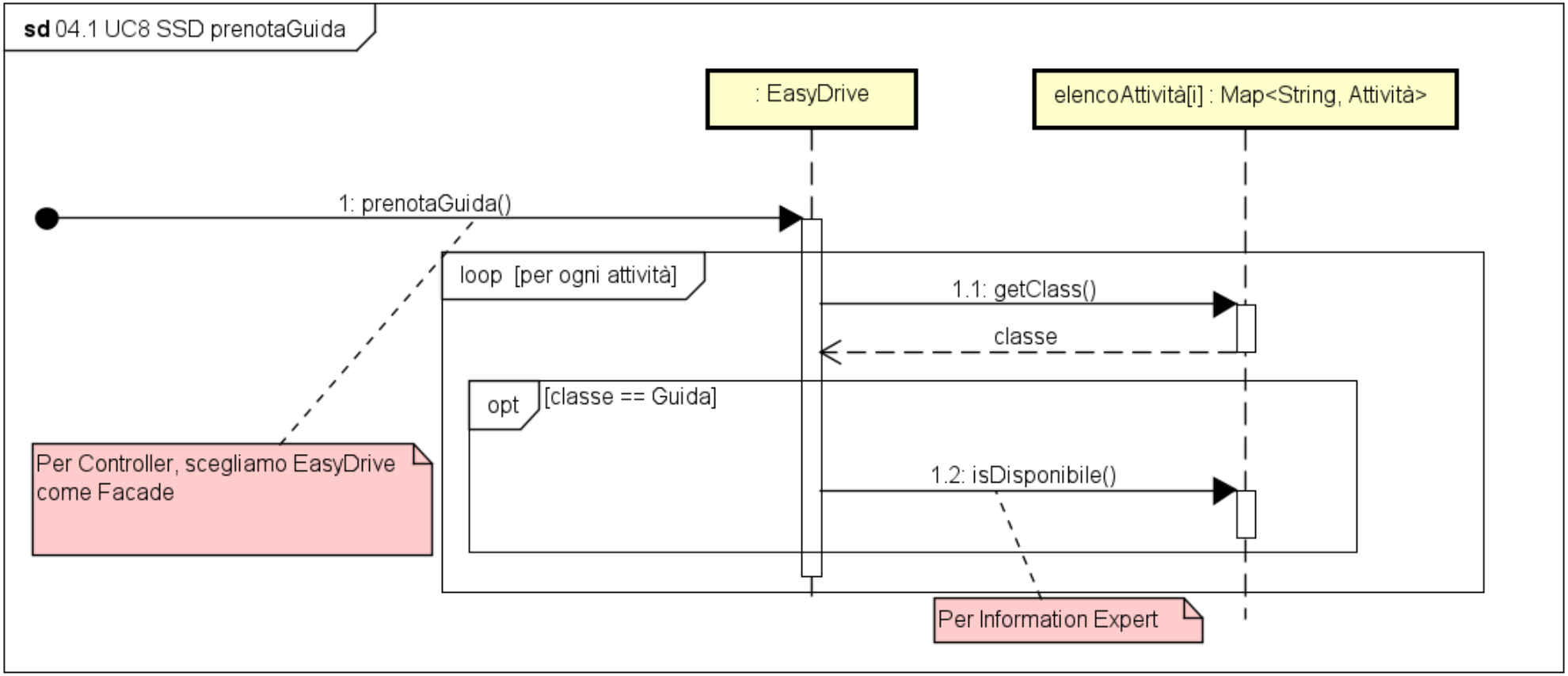


- **Aggiorna Bocciati Debitori**

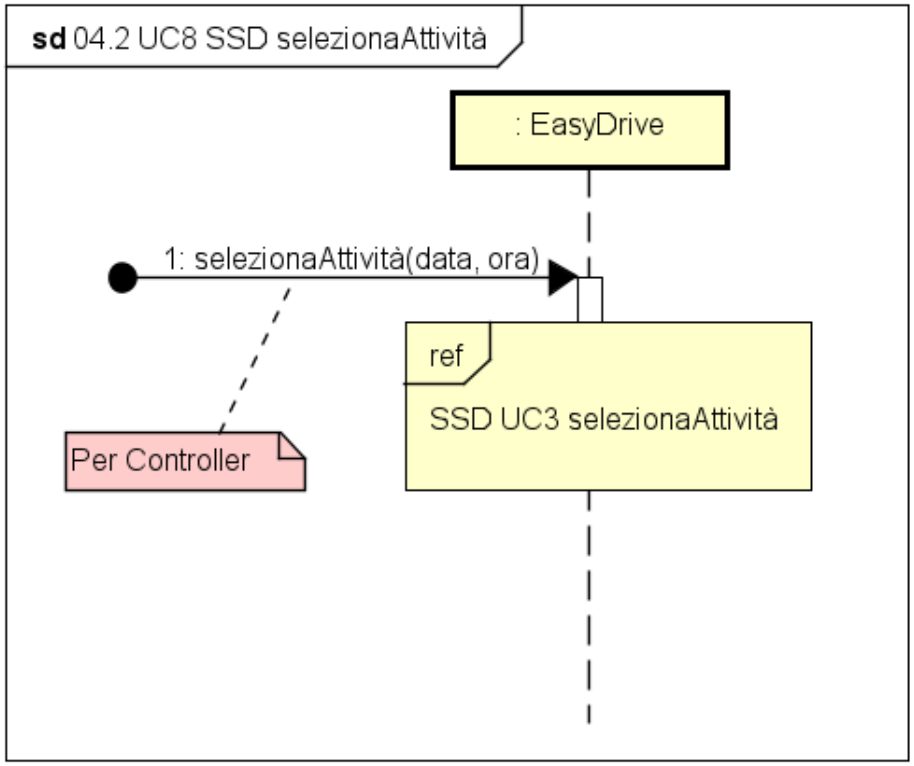


UC8:

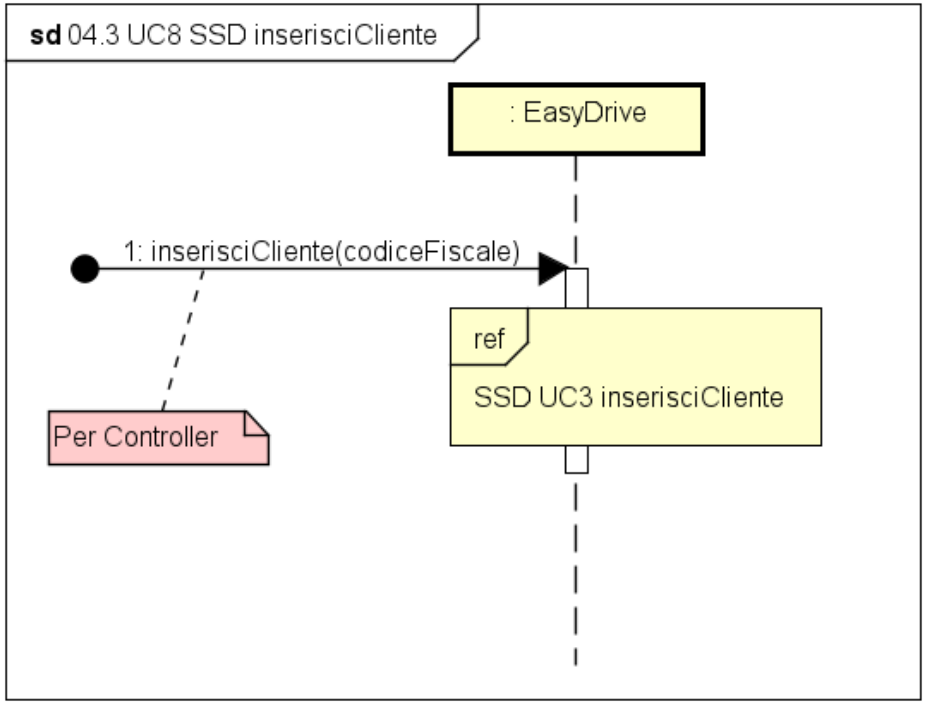
- **Prenota Guida**



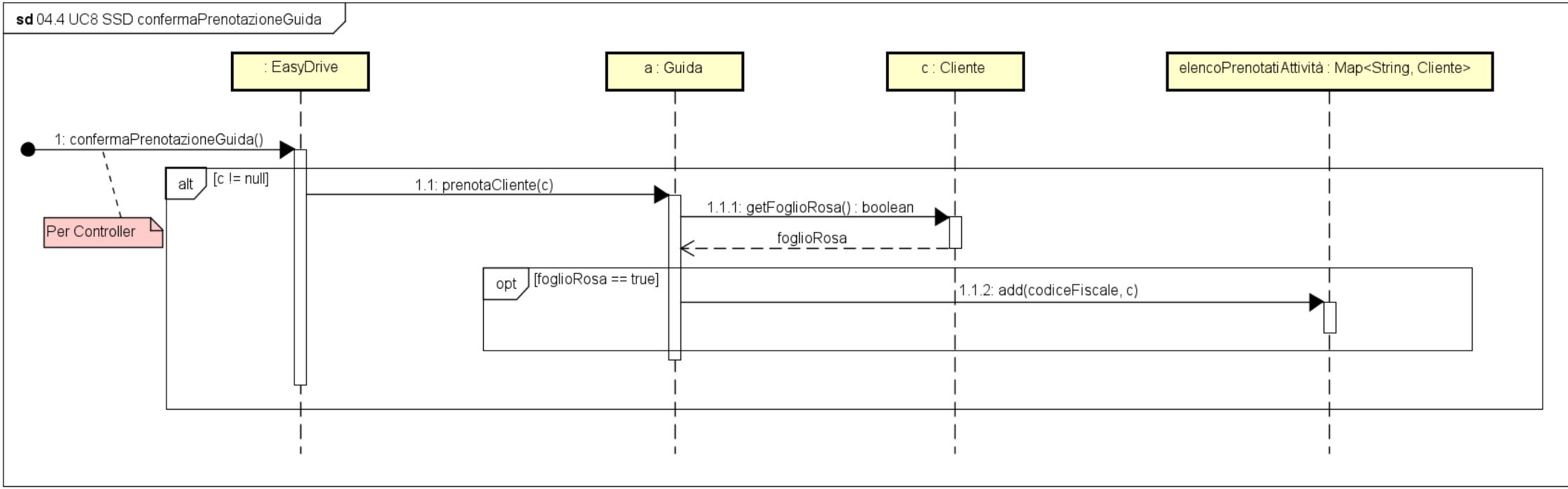
- **Seleziona Attività**



- **Inserisci Cliente**

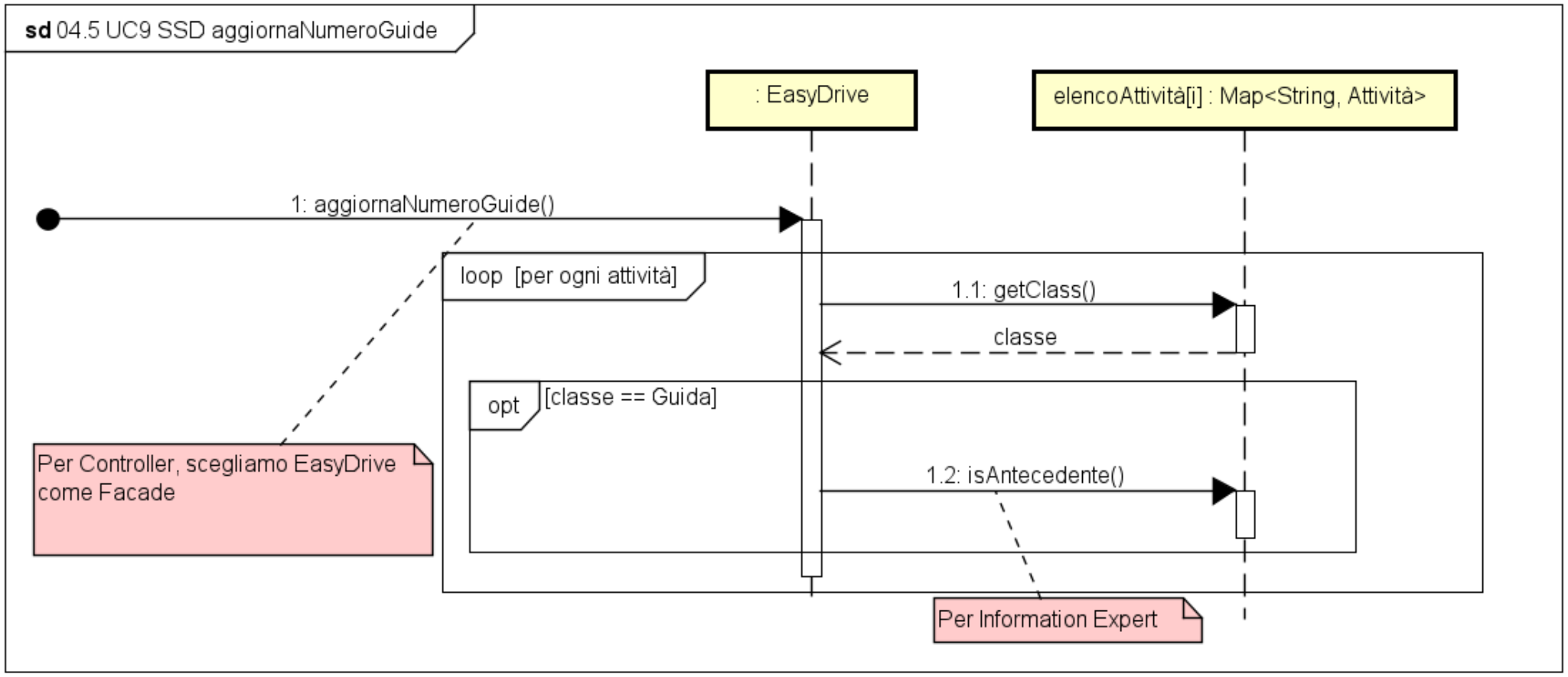


- **Conferma Prenotazione Guide**

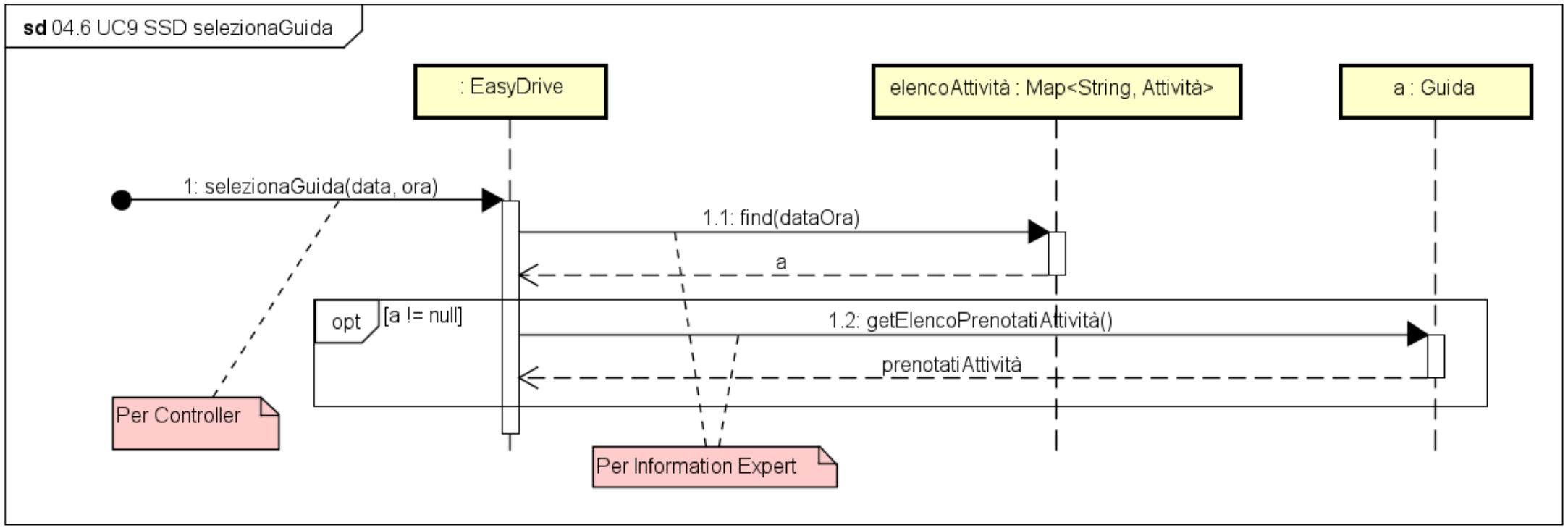


UC9:

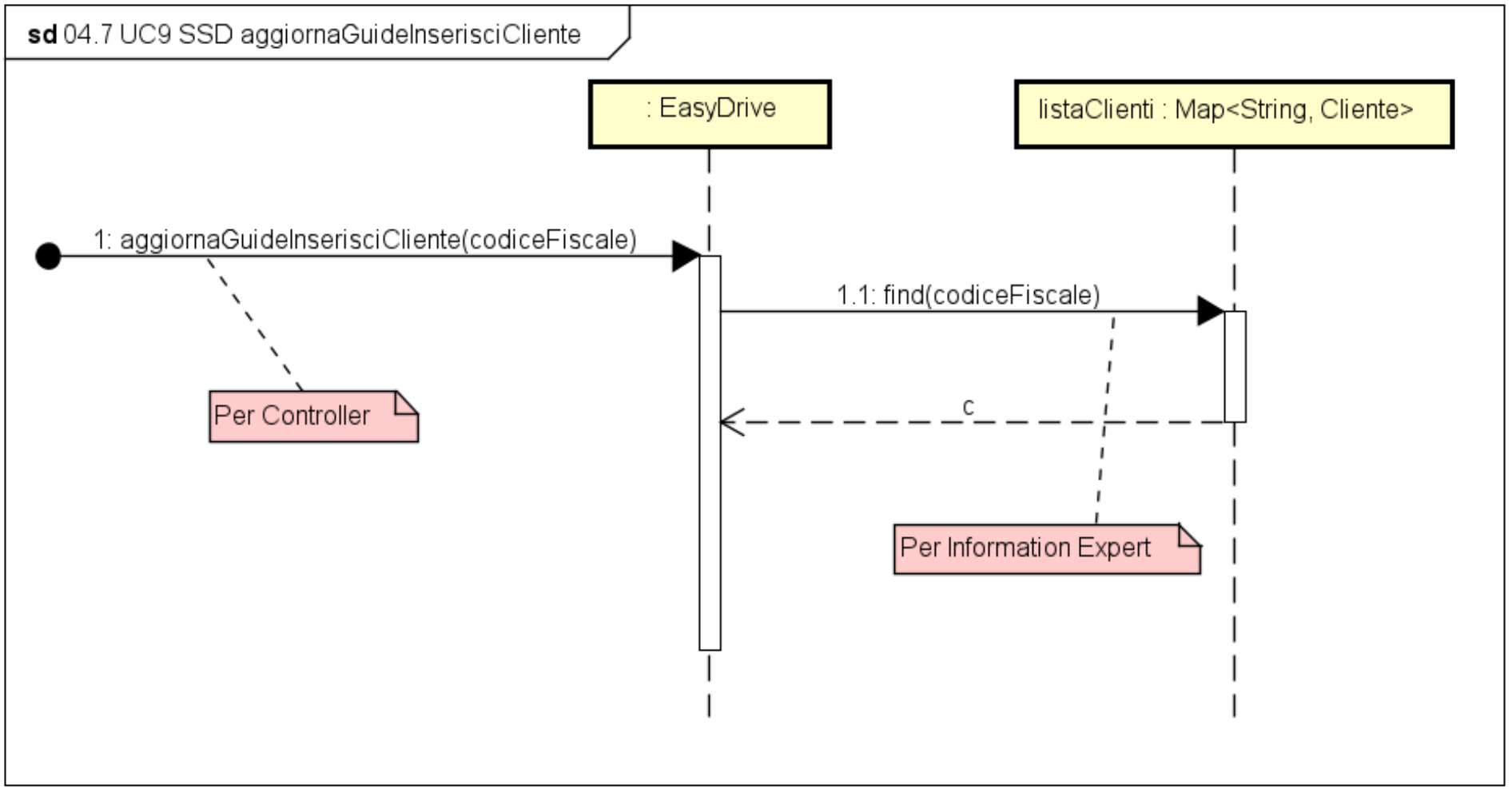
- **Aggiorna Numero Guide**



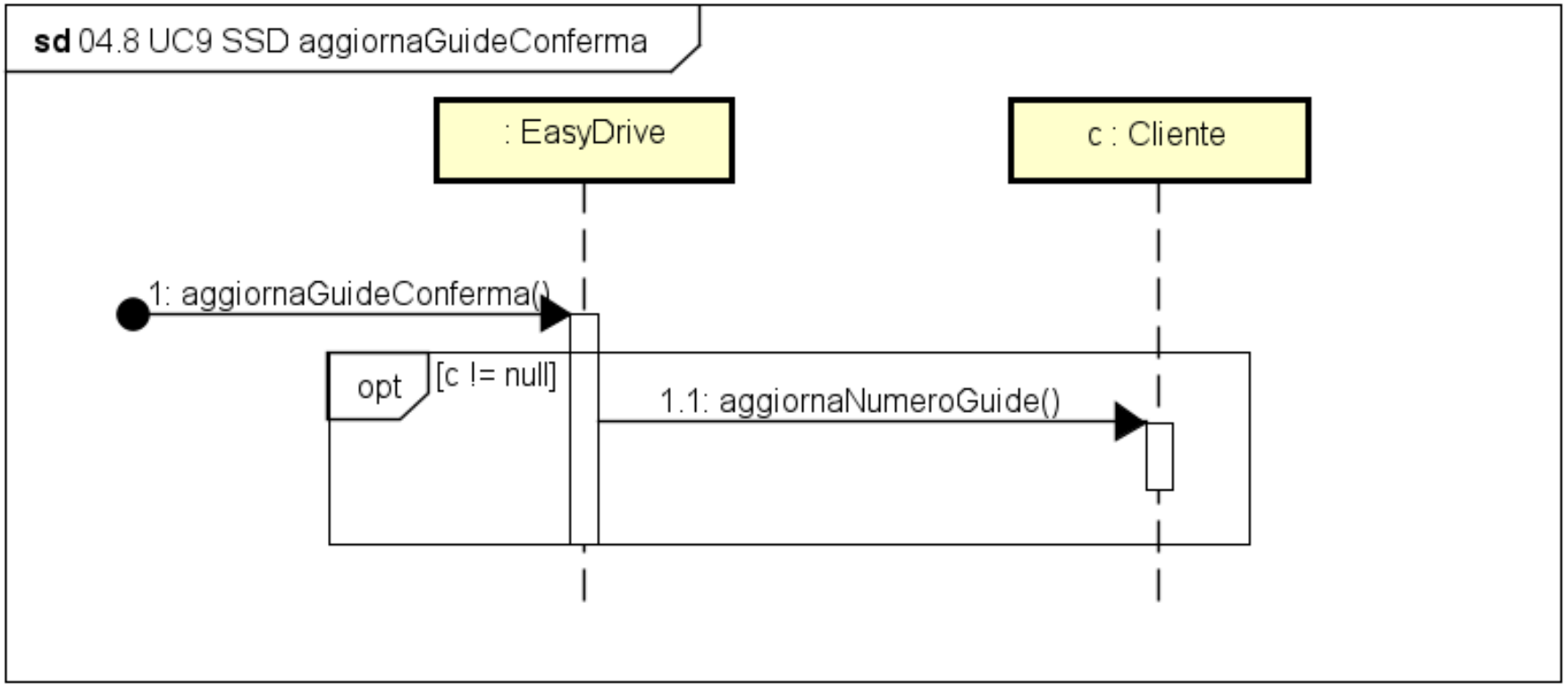
- **Seleziona Guida**



- **Aggiorna Guide Inserisci Cliente**

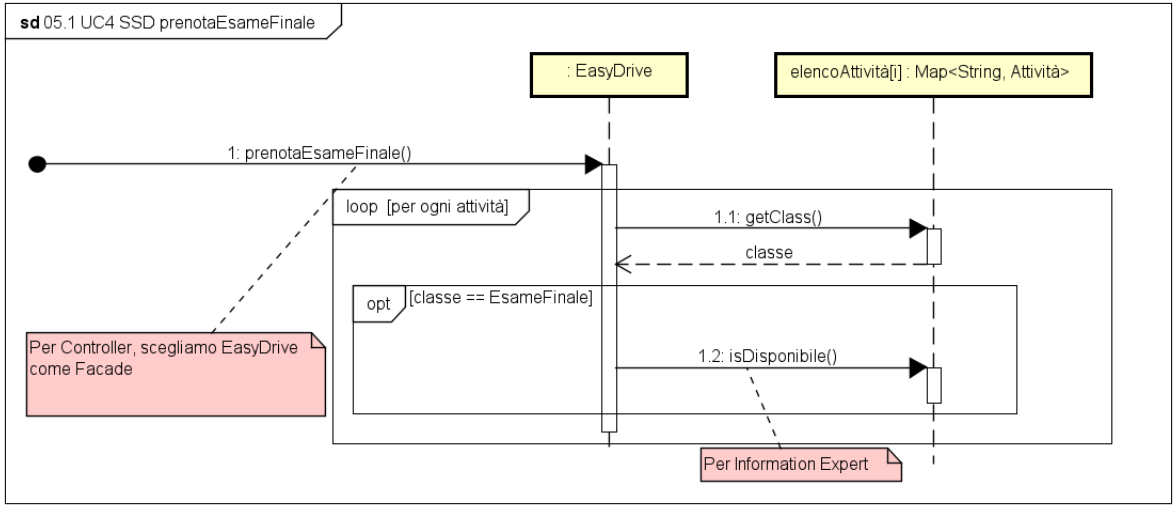


- **Aggiorna Guide Conferma**

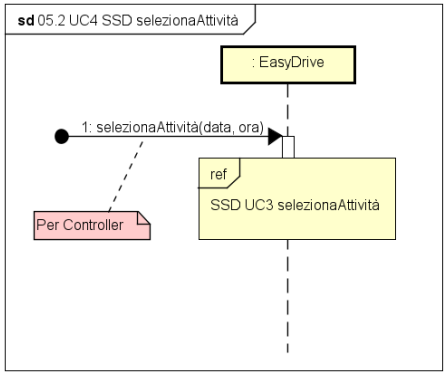


UC4:

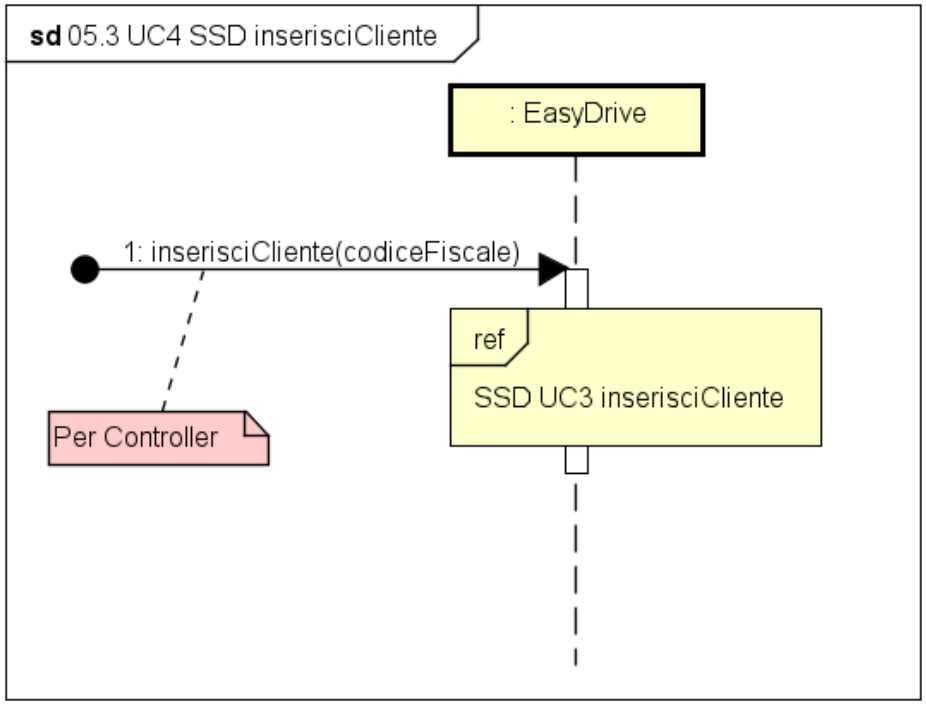
- **Prenota Esame Finale**



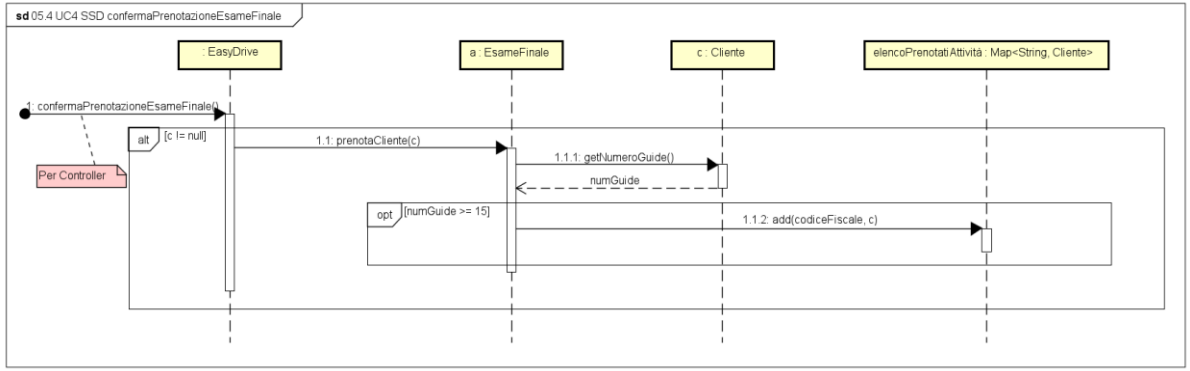
- **Seleziona Attività**



- **Inserisci Cliente**

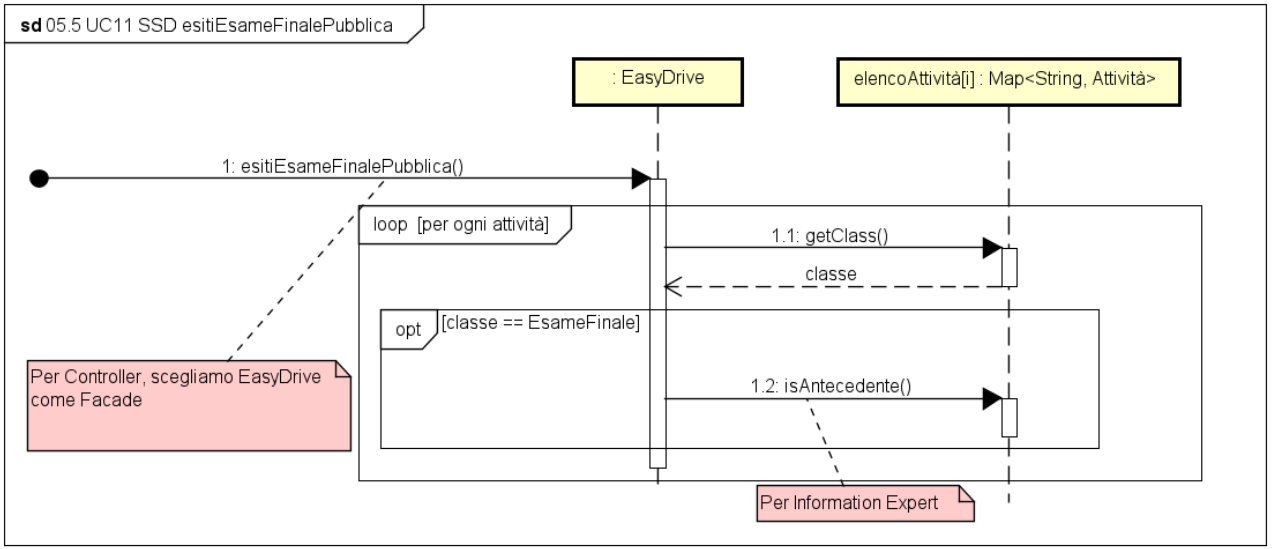


- **Conferma Prenotazione Esame Finale**

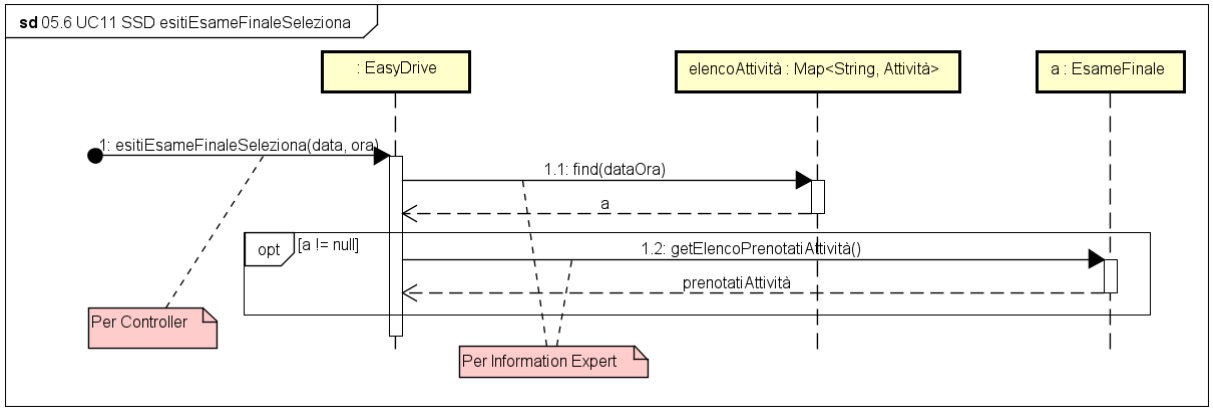


UC11:

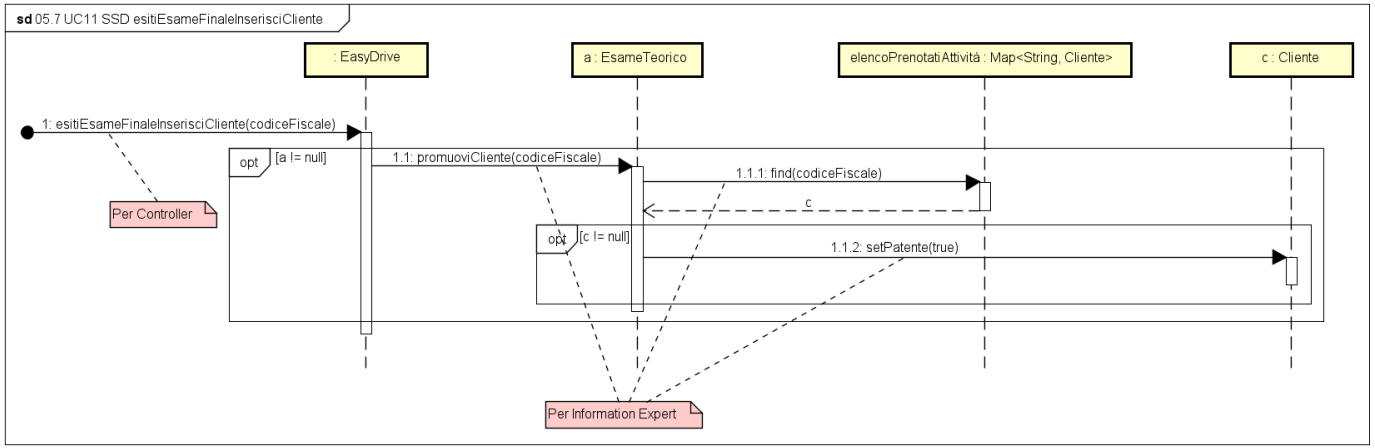
- **Esiti Esame Finale Pubblica**



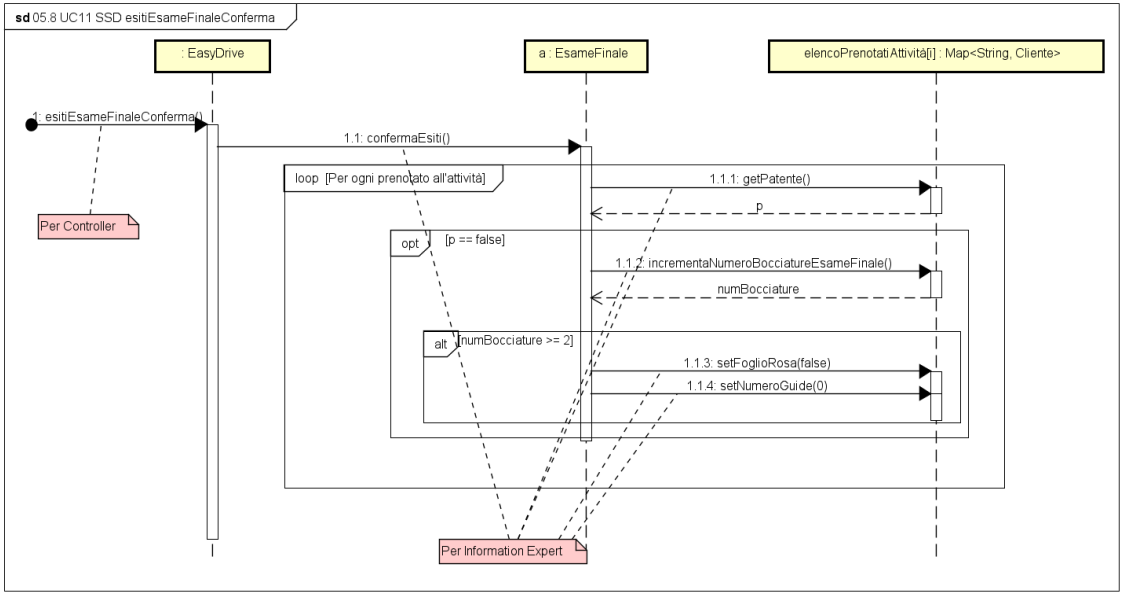
- **Esiti Esame Finale Seleziona**

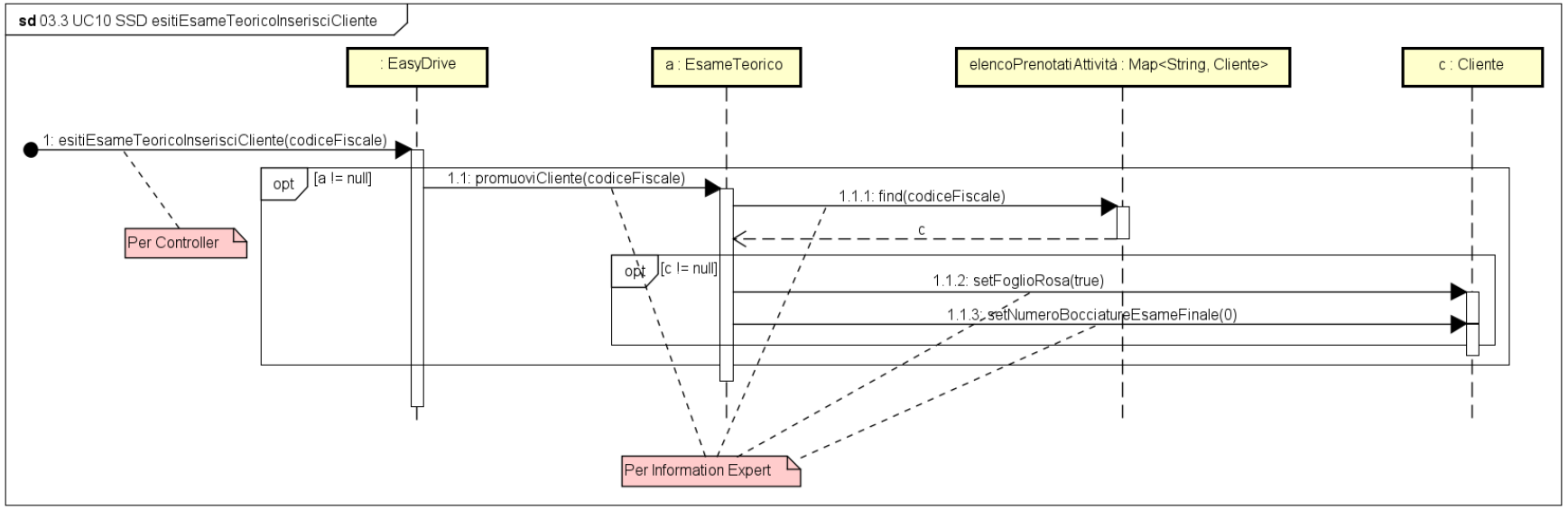


- **Esiti Esame Finale Inserisci Cliente**



- **Esiti Esame Finale Conferma**





(In questo caso è stato modificato anche l'SSD esitiEsameTeoricoInserisciCliente relativo all' UC10 in modo da resettare il numeroBocciatureEsameFinale una volta superato l'esame teorico)

4. Test

Il testing è una fase essenziale del processo di sviluppo di un programma robusto ed efficiente. Permette inoltre di ridurre in modo sostanziale i costi di manutenzione di un applicazione. La probabilità di rilevare malfunzionamenti è proporzionale al numero di test eseguiti, tuttavia è difficile testare un programma completamente. Infatti, le possibili combinazioni di valori di input da prendere in considerazione sono enormi, e non sempre possono essere riprodotte in un tempo ragionevole. Tuttavia, un testing progettato accuratamente può rivelare comportamenti anomali ed indesiderati al fine di rendere il software realizzato funzionante e funzionale secondo le specifiche del committente.

Segue un elenco puntato con la descrizione della metodologia di testing.

- **EasyDrive (testEasyDrive.java)**:
    - **testAddCliente:** Utilizziamo il metodo "*addCliente*" della classe *EasyDrive* per inserire nuovi clienti all'interno della sua mappa *listaClienti.* Se il test va a buon fine, verranno stampati su console tutti i clienti presenti in *listaClienti* della classe *EasyDrive*, altrimenti verrà stampato su console "Nessun cliente in lista".
    - **testRemoveCliente:** Per prima cosa verifichiamo che un cliente inserito in *listaClienti* della classe *EasyDrive* venga rimosso correttamente chiamando il metodo "removeCliente" e passando come parametro il suo codice fiscale. Se tutto va a buon fine il metodo "*getCliente*" di *EasyDrive* con codice fiscale utilizzato in precedenza dovrebbe restituire NULL. Successivamente verifichiamo che, passando il codice fiscale di un utente non inserito in *listaClienti* nel metodo "*removeCliente*" di *EasyDrive,* questo generi il messaggio "Impossibile rimuovere il cliente con il codice fiscale selezionato".
    - **testAddLezione:** Per prima cosa recuperiamo la lista di tutti gli argomenti trattati chiamando in metodo "*getListaArgomenti*" della classe *EasyDrive*, questa ci sarà utile per la creazione delle lezioni. Successivamente utilizziamo il metodo "*addLezione*" di *EasyDrive* per inserire nuove lezioni all'interno della sua mappa *elencoLezioni.* Se il test va a buon fine, verranno stampati su console tutte le lezioni presenti in *elencoLezioni* della classe *EasyDrive*, altrimenti verrà stampato su console "Nessun lezione in lista".
    - **testRemoveLezione:** Per prima cosa verifichiamo che una lezione inserita in *elencoLezioni* della classe *EasyDrive* venga rimossa correttamente chiamando il metodo "removeLezione" e passando come parametro la sua data e ora. Se tutto va a buon fine il metodo "*getLezione*" di *EasyDrive* con data e ora utilizzate in precedenza dovrebbe restituire NULL. Successivamente verifichiamo che, passando la data e ora di una lezione non inserita in *elencoLezioni* nel metodo "*removeLezione*" di *EasyDrive,* questo generi il messaggio "Impossibile rimuovere la lezione con la data e l'ora selezionate".
    - **TestAggiornaFrequenzaClienti:** Una volta inserita una lezione in *elencoLezioni* della classe EasyDrive tramite il metodo "*addLezione*", chiamiamo il metodo "*aggiornaFrequenzaClienti*" passando come parametri la data e l'ora della lezione appena inserita. Il test avrà esito positivo se la lezione selezionata diventerà *lezioneCorrente* in *EasyDrive*, quindi se il metodo "*getLezioneCorrente*" ritorni un valore diverso da NULL.

- **TestInserisciClienteFrequentante:** Una volta inseriti un cliente ed una lezione rispettivamente in *listaClienti* ed *elencoLezioni* della classe *Easydrive*, chiamiamo il metodo "*aggiornFrequenzaClienti*" inserendo data e ora della lezione inserita. Successivamente utilizziamo il metodo "*inserisciClienteFrequentante*" di *EasyDrive* passando come parametro il codice fiscale del cliente inserito in precedenza, il metodo andrà a buon fine se "*getClienteCorrente*" ritorni un valore diverso da NULL. In seguito, richiamiamo il metodo "*aggiornaFrequenzaClienti"* e "*inserisciClienteFrequentante*" di *EasyDrive* passando però come parametro un codice fiscale non presente in *listaClienti*, stavolta ci aspettiamo che il metodo "*getClienteCorrente*" ritorni il valore NULL.
- **TestConfermaInserimento:** Inseriamo un cliente e diverse lezioni rispettivamente in *listaClienti* ed *elencoLezioni* della classe *Easydrive.* Per prima cosa chiamiamo i metodi "*aggiornaFrequenzaClienti*" e "*inserisciClienteFrequentante*" passando come parametri i dati del cliente e di una lezione inseriti in precedenza, dopodiché chiamiamo il metodo "*confermaInserimento*", se tutto è andato a buon fine l'attributo *frequenzaLezioni* del cliente selezionato dovrebbe aggiornarsi poiché non aveva ancora seguito l'argomento trattato nella lezione selezionata. Successivamente richiamiamo i metodi "*aggiornaFrequenzaClienti*" e "*inserisciClienteFrequentante*", passando come parametri i dati del cliente e di una lezione già seguita da esso. A questo punto chiamiamo il metodo "*confermaInserimento*", se tutto è andato a buon fine l'attributo *frequenzaLezioni* del cliente selezionato non dovrebbe aggiornarsi. Se invece in "*inserisciClienteFrequentante*" non viene inserito il codice fiscale di un cliente presente in *listaClienti* il metodo "*confermaInserimento*" non dovrebbe produrre nessun risultato.
- **testAddEsameTeorico:** Utilizziamo il metodo *"addEsameTeorico"* della classe EasyDrive per inserire un nuovo esame Teorico all'interno della sua mappa *elencoEsamiTeorici*. Se il test va a buon fine verrà stampato su console "Nessun esame teorico in lista" nel caso in cui la mappa sia vuota, in caso contrario verranno stampati gli esami teorici presenti in *elencoEsamiTeorici.* Inoltre, successivamente proviamo a passare anche la data di un esame esistente e utilizziamo "assertNotNull" perché ci aspettiamo che ci venga restituito diverso da NULL nel caso in cui il test vada a buon fine, allo stesso modo facciamo passando la data di un esame non presente in *elencoEsamiTeorici* e dovrebbe restituire NULL.
- **testRemoveEsameTeorico:** Per prima cosa verifichiamo che un esame teorico inserito in *elencoEsamiTeorici* della classe EasyDrive venga rimosso correttamente chiamando il metodo *"removeEsameTeorico"* passando come parametri data e ora, il mese e il giorno. Se tutto va a buon fine il metodo "*getEsameTeorico*" nel momento in cui riceve gli stessi parametri dell'esame che dovrebbe essere stato cancellato ci aspettiamo che venga restituito NULL, ed in tal caso se l'esame teorico non viene trovato l'amministratore viene avvertito. Successivamente aggiungiamo un esame tramite il metodo "a*ddEsameTeorico*" e verifichiamo che passando un anno, data e ora e giorno di un esame non inserito in *elencoEsamiTeorici* nel metodo *"removeEsameTeorico"*, questo generi il messaggio "Impossibile rimuovere poiché non è registrato nessun esame per la data e ora selezionati" e verifichiamo che la rimozione di un esame di un'altra data non abbia compromesso quello presente in lista.
- **testPrenotaEsameTeorico:** per prima cosa aggiungiamo all'*elencoEsamiTeorici* due esami teorici tramite il metodo *"addEsameTeorico"* inserendo delle date successive a quella odierna. Chiamando il metodo *prenotaEsameTeorico* ci aspettiamo che esso torni una lista contenente tutti gli oggetti di tipo "*EsameTeorico*" con data posteriore a quella odierna. Se tutto è andato a buon fine prevediamo che la lista ritornata dal metodo "*prenotaEsameTeorico*" contenga i due esami teorici inseriti inizialmente.
- **testSelezionaEsame:** Dopo aver inserito un esame teorico all'interno di elencoEsamiTeorici tramite il metodo "*addEsameTeorico*", chiamiamo il metodo "selezionaEsame" passando come parametri la data e ora dell'esame aggiunto in precedenza. Ci aspettiamo che l'esame selezionato

diventi "*esameTeoricoCorrente*". Successivamente impostiamo "*esameTeoricoCorrente*" a NULL e richiamiamo nuovamente il metodo "*selezionaEsame*" passando stavolta come parametri la data e l'ora di un esame non presente in *elencoEsamiTeorici*. Ci aspettiamo che "*esameTeoricoCorrente*" continui ad essere NULL.

- **testInserisciCliente:** inizialmente aggiungiamo un esame teorico in *elencoEsamiTeorici* e un cliente tramite il metodo *"addCliente"*, viene selezionato l'esame appena aggiunto attraverso la funzione *"selezionaEsame"* e successivamente chiamiamo il metodo "*inserisciCliente*" passando come parametro il codice fiscale del cliente aggiunto in precedenza; se tutto è andato a buon fine ci aspettiamo che il cliente inserito diventi "*clienteCorrente*". Se invece selezioniamo un esame che non esiste all'interno di *elencoEsamiTeorici* ci aspetteremo NULL quando chiamiamo il metodo "*getClienteCorrrente"* perché non è stato possibile associare il cliente selezionato ad un esame esistente nel sistema.
- **testConfermaPrenotazione:** per prima cosa aggiungiamo un esame in *elencoEsamiTeorici* e un cliente all'interno della mappa *listaClienti* rispettivamente tramite i metodi "*addEsameTeorico*" e "*addCliente*". Successivamente creiamo un nuovo argomento della lezione inserendo come descrizione: "Segnali di pericolo" e lo passiamo come parametro nel metodo "*incrementaFrequenzaLezioni*" dell'oggetto di tipo Cliente creato in precedenza, in questo modo il cliente avrà una *frequenzaLezioni* maggiore della soglia minima del 70% richiesta per prenotarsi all'esame. Infine, selezioniamo l'esame ed il cliente inseriti in precedenza e chiamiamo il metodo "*confermaPrenotazione*" che inserirà *clienteCorrente* in *elencoPrenotatiEsameTeorico*.
- **testEsitiEsameTeoricoPubblica:** Utilizziamo il metodo "*addEsameTeorico*" della classe *EasyDrive* per inserire nuovi oggetti di tipo esame teorico all'interno della mappa *"elencoEsamiTeorici"*. In particolare, inseriamo, 2 esami con date antecedenti alla data odierna, e un esame con una data del 2024; in tal modo verifichiamo che solo gli esami antecedenti alla data odierna verranno effettivamente inseriti all'interno della lista *"EsamiTeoriciDisponibili"*, per tale motivo ci aspettiamo che la lista non sia vuota. Infine, stampiamo in console tutti gli elementi aggiunti in tale lista.
- **testEsitiEsameTeoricoSeleziona:** Per prima cosa inseriamo all'interno della mappa *"elencoEsamiTeorici"* un esame con data antecedente a quella odierna, e in *"listaClienti"* aggiungiamo 3 oggetti di tipo cliente. A questo punto inseriamo i 3 clienti nella lista *"elencoPrenotatiEsamiTeorico"* dell'oggetto *"EsameTeorico"* inserito in precedenza e infine chiamiamo il metodo *"EsitiEsameTeoricoSeleziona"* passando come parametri la data e l'ora dell'esame inserito. Poiché non è possibile prenotare i clienti all'esame se non si ha più del 70% della frequenza delle lezioni, facciamo in modo che abbiano una frequenza lezioni adeguata per la prenotazione. Ci aspettiamo che l'esame teorico inserito diventi *"esameTeoricoCorrente"* e che l'HashMap ritornato da questo metodo *(esitiEsameTeoricoSeleziona)* contenga i 3 clienti inseriti in precedenza, e li stampiamo in console.
- **testEsameTeoricoInserisciCliente:** Per prima cosa inseriamo all'interno della mappa *"elencoEsamiTeorici"* un esame con data antecedente a quella odierna, e in *"listaClienti"* aggiungiamo 3 oggetti di tipo cliente. A questo punto inseriamo i 3 clienti nella lista *"elencoPrenotatiEsamiTeorico"* dell'oggetto *"EsameTeorico"* inserito in precedenza. Poiché non è possibile prenotare i clienti all'esame se non si ha più del 70% della frequenza delle lezioni, facciamo in modo che abbiano una frequenza lezioni adeguata per la prenotazione. Dopo di che selezioniamo l'esame teorico inserito tramite la funzione *"EsitiEsameTeoricoSeleziona"* e successivamente promuoviamo i clienti che hanno

partecipato all'esame selezionato settando il loro attributo "foglioRosa" a true, mentre non faremo lo stesso con i prenotati che non l'hanno passato. Infine, per verificare, stampiamo in console tutti i clienti (promossi e non) che hanno effettuato l'esame.

- **testEsitiEsameTeoricoConferma:** Per prima cosa inseriamo all'interno della mappa *"elencoEsamiTeorici"* un esame con data antecedente a quella odierna, e in *"listaClienti"* aggiungiamo 3 oggetti di tipo cliente. A questo punto inseriamo i 3 clienti nella lista *"elencoPrenotatiEsamiTeorico"* dell'oggetto *"EsameTeorico"* inserito in precedenza. Poiché non è possibile prenotare i clienti all'esame se non si ha più del 70% della frequenza delle lezioni, facciamo in modo che abbiano una frequenza lezioni adeguata per la prenotazione. Dopo di che selezioniamo l'esame teorico inserito tramite la funzione *"EsitiEsameTeoricoSeleziona"* e successivamente promuoviamo i clienti che hanno partecipato all'esame selezionato settando il loro attributo *"foglioRosa"* a true, mentre non faremo lo stesso con i prenotati che non l'hanno passato. A questo punto chiamiamo il metodo *"EsitiEsameTeoricoConferma"* il quale incrementerà l'attributo *"numeroBocciature"* dei clienti che non hanno superato l'esame e verifichiamo la corretta esecuzione del metodo attraverso degli assert. Infine, per verificare, stampiamo in console tutti i clienti (promossi e non) che hanno effettuato l'esame.
- **TestAggiornaBocciatoDebitore:** Per prima cosa inseriamo all'interno della mappa *"listaClienti"* 3 oggetti di tipo cliente e incrementiamo l'attributo *"numeroBocciature"* di due clienti al fine di avere un numero di bocciature >= 2. Successivamente chiamiamo il metodo *"aggiornaBocciatiDebitori"*, e ci aspettiamo che i due clienti, il quale numero di bocciature è stato incrementato in precedenza, vengano eliminati dalla mappa *"listaClient*i" ed inseriti nella mappa *"listaBocciatiDebitori"*, ovvero la lista che contiene i clienti che devono ripagare l'iscrizione alla scuola guida poiché sono stati bocciati almeno due volte.
- **testAddGuida:** Per prima cosa aggiungiamo tramite il metodo "*addGuida*" 3 guide specificando la loro data e ora. Successivamente ci facciamo tornare l'elenco di attività e se tale elenco è vuoto vuol dire che non vi è nessuna guida inserita in lista, in caso contrario vengono stampate le guide inserite in lista (in questo caso 3). Verifichiamo, inoltre, la correttezza anche tramite l'utilizzo di "*AssertNotNull*" e "*AssertNull*", passando rispettivamente prima una data di un esame esistente e poi la data di un esame non esistente.
- **testPrenotaGuida:** Inizialmente aggiungiamo 3 guide tramite il metodo "*addGuida*", successivamente creiamo un ArrayList di tipo guida (nominato "*guideDisponibili*") e chiameremo il metodo "*prenotaGuida*" per riempire proprio quest'ultimo. Vedremo che una volta chiamato questo metodo, "*guideDisponibili*" avrà all'interno solo la data successiva alla data odierna e al suo interno non vi saranno quelle con date antecedenti.
- **testConfermaPrenotazioneGuida:** In questo test prima di tutto aggiungiamo una guida, un cliente e un *esameTeorico* utilizzando rispettivamente il metodo "*addGuida*", "*addCliente*" e "*addEsameTeorico*". Faremo, inoltre, in modo che la frequenza delle lezioni sia tale da poter permettere la prenotazione all'esame teorico (>70%). Subito dopo faremo una prova per verificare che se viene selezionata un'attività con data non registrata nel sistema verremo avvisati. Invece, nel momento in cui selezioniamo un'attività con data presente nel sistema saremo in grado di poter prenotare il cliente passandogli il codice fiscale del cliente che vogliamo prenotare tramite il metodo "*inserisciCliente*"; tuttavia, viene dimostrato come il cliente non può essere prenotato se non ha ancora superato l'esame teorico, invece, se facciamo la stessa procedura e il cliente ha superato l'esame teorico è possibile prenotarlo. In particolare, viene mostrato come in quest'ultimo caso se viene inserito erroneamente un cliente non registrato nel sistema ci verrà restituito un messaggio di avviso e non potremo procedere con la prenotazione, mentre se eseguiamo

tutta la procedura per bene, inserendo anche il codice fiscale di un cliente davvero registrato nel sistema e che rispetti tutti i vincoli, sarà possibile prenotarlo alla guida e verrà stampato l'elenco dei prenotati.

- **testAggiornaNumeroGuide:** Innanzitutto aggiungiamo 3 guide nel sistema, tra cui una con una data successiva a quella odierna, dunque non verrà inserita all'interno dell'ArrayList "*guideDisponibili*". Nota bene, in questo caso "*guideDisponibili*" ha esattamente il significato opposto a quello che aveva prima, infatti verrà riempito dalle guide con data ANTECEDENTE a quella odierna. Verrà chiamato il metodo "*aggiornaNumeroGuide*" e in questo modo verrà riempito l'ArrayList. Successivamente verranno stampate solo le guide disponibili, che per l'appunto sono antecedenti, dunque ne verranno stampate solo 2.
- **testSelezionaGuida:** Prima di tutto aggiungiamo una guida, un cliente e un esameTeorico utilizzando rispettivamente il metodo "*addGuida*", "*addCliente*" e "*addEsameTeorico*". Faremo, inoltre, in modo che la frequenza delle lezioni sia tale da poter permettere la prenotazione all'esame teorico (>70%). A questo punto prenotiamo e facciamo superare l'esame teorico al cliente in modo da poter prenotare una guida. Dopo di che viene creata un HashMap di nome "prenotati" e verrà verificato prima di tutto che se inseriamo una data di una guida non registrata nel sistema verremo avvertiti, mentre se inseriamo una data corretta allora potremo ricavare l'elenco dei prenotati a tale guida. Inoltre, ci aspettiamo che la guida selezionata diventi *attivitàCorrente* e che l'hashmap "*prenotati*" non sia vuoto verificando entrambe le cose tramite l'utilizzo degli *assertNotNull*.
- **testAggiornaGuideInserisciCliente:** Prima di tutto aggiungiamo una guida, un cliente e un *esameTeorico* utilizzando rispettivamente il metodo "*addGuida*", "*addCliente*" e "*addEsameTeorico*". Faremo, inoltre, in modo che la frequenza delle lezioni sia tale da poter permettere la prenotazione all'esame teorico (>70%). A questo punto prenotiamo e facciamo superare l'esame teorico al cliente in modo da poterlo prenotare ad una guida. Una volta fatto ciò selezioniamo una guida alla quale far partecipare il cliente e chiamiamo il metodo "*aggiornaNumeroGuideInserisciCliente*" e, nel caso in cui venga selezionato un cliente non registrato nel sistema verremo avvisati, altrimenti verrà selezionato il relativo cliente che ci aspettiamo diventi *clienteCorrente*.
- **testAggiornaGuideConferma:** Anche in quest'ultimo caso, prima di tutto aggiungiamo una guida, un cliente e un *esameTeorico* utilizzando rispettivamente il metodo "*addGuida*", "*addCliente*" e "*addEsameTeorico*". Faremo, inoltre, in modo che la frequenza delle lezioni sia tale da poter permettere la prenotazione all'esame teorico (>70%). A questo punto prenotiamo e facciamo superare l'esame teorico al cliente e, una volta fatto ciò, lo prenotiamo ad una guida. Successivamente procediamo con l'aggiornamento del numero di guide effettuate, selezioniamo la guida, chiamiamo il metodo "*aggiornaGuideInserisciCliente*" al quale passiamo il codice fiscale e ci aspettiamo che il cliente selezionato diventi *clienteCorrente*. In tal modo chiamando il metodo "*aggiornaGuideConferma*" saremo in grado di incrementare il numero di guide del cliente inserito. Per verificare ciò effettuiamo un *assert* e stampiamo tutti i dati del cliente selezionato.
- **testAddEsameFinale:** Utilizziamo il metodo *"addEsameFinale"* della classe EasyDrive per inserire un nuovo esame Finale all'interno della sua mappa *elencoEsamiFinali*. Se il test va a buon fine verrà stampato su console "Nessun esame finale in lista" nel caso in cui la mappa sia vuota, in caso contrario verranno stampati gli esami finali presenti in *elencoEsamiFinali.* Inoltre, successivamente proviamo a passare anche la data di un esame esistente e utilizziamo "assertNotNull" perché ci aspettiamo che ci venga restituito un valore diverso da NULL nel caso in cui il test vada a buon fine, allo stesso modo facciamo passando la data di un esame non presente in *elencoEsamiFinali* e dovrebbe restituire NULL.

- **testPrenotaEsameFinale:** per prima cosa aggiungiamo all'*elencoEsamiFinali* due esami finali tramite il metodo *"addEsameFinale"* inserendo delle date successive a quella odierna. Chiamando il metodo *prenotaEsameFinale* ci aspettiamo che esso torni una lista contenente tutti gli oggetti di tipo "*EsameFinale*" con data posteriore a quella odierna. Se tutto è andato a buon fine prevediamo che la lista ritornata dal metodo "*prenotaEsameFinale*" contenga i due esami finali inseriti inizialmente.
- **testConfermaPrenotazioneEsameFinale:** per prima cosa aggiungiamo un esame finale, un esame teorico, una guida e un cliente, rispettivamente tramite i metodi "addEsameFinale", "addEsameTeorico", "addGuida" e "addCliente". Successivamente creiamo un nuovo argomento della lezione inserendo come descrizione: "Segnali di pericolo" e lo passiamo come parametro nel metodo "*incrementaFrequenzaLezioni*" dell'oggetto di tipo Cliente creato in precedenza, in questo modo il cliente avrà una *frequenzaLezioni* maggiore della soglia minima del 70% richiesta per prenotarsi all'esame. Tuttavia, verifichiamo che se il cliente non ha un numero di guide sufficienti (< 15) e nemmeno il foglio rosa pari a true, non potrà essere prenotato per l'esame finale. Una volta fallito il tentativo di prenotazione precedente, prenotiamo il cliente per l'esame teorico, lo promuoviamo ovvero settiamo il suo foglioRosa a true e incrementiamo il numero di guide pari a 15 così che possa essere effettivamente prenotato per l'esame finale. Verifichiamo, infine, l'effettiva prenotazione tramite un assertNotNull sull'elenco dei prenotati a tali attività.
- **testEsitiEsameFinalePubblica:** Utilizziamo il metodo *"addEsameFinale"* per aggiungere 3 oggetti di tipo esame Finale, tra cui un esame finale sarà con data non antecendente e dunque non verrà inserito nella lista *"esamiFinaliDisponibili"*, infatti nel momento in cui la stamperemo, tale lista avrà solo 2 elementi.
- **testEsitiEsameFinaleSeleziona:** Prima di tutto aggiungiamo un esame finale e 3 clienti rispettivamente tramite i metodi *"addEsameFinale"* e *"addCliente"*. Dopo di che prenotiamo i 3 clienti all'esame finale e li inseriamo nell'esame finale settando prima i loro fogli rosa a true e il loro numero di guide pari a 15 così da permettere la conferma della prenotazione per l'esame finale, tramite appunto il metodo "confermaPrenotazioneEsameFinale". Verifichiamo, inoltre, che se inseriamo un esame finale non registrato nel sistema verremo avvertiti. Infine, stampiamo tutti i prenotati all'esame finale.
- **testEsameFinaleInserisciCliente:** Prima di tutto aggiungiamo un esame finale e 3 clienti rispettivamente tramite i metodi *"addEsameFinale"* e *"addCliente"*. Dopo di che prenotiamo i 3 clienti all'esame finale e li inseriamo nell'esame finale settando prima i loro fogli rosa a true e il loro numero di guide pari a 15 così da permettere la conferma della prenotazione per l'esame finale, tramite appunto il metodo "confermaPrenotazioneEsameFinale". Successivamente chiamiamo il metodo "esitoEsameFinaleInserisciCliente" in maniera tale da promuovere solo i clienti c1 e c2, infatti, quando stamperemo la lista avremo che solo i clienti c1 e c2 avranno l'attributo patente pari a true, mentre c3 lo avrà pari a false.
- **testEsitiEsameFinaleConferma:** Prima di tutto aggiungiamo un esame finale e 3 clienti rispettivamente tramite i metodi *"addEsameFinale"* e *"addCliente"*. Dopo di che prenotiamo i 3 clienti all'esame finale e li inseriamo nell'esame finale settando prima i loro fogli rosa a true e il loro numero di guide pari a 15 così da permettere la conferma della prenotazione per l'esame finale, tramite appunto il metodo "confermaPrenotazioneEsameFinale". Successivamente chiamiamo i metodi "esitoEsameFinaleInserisciCliente" ed "esitiEsameFinaleConferma" in maniera tale da promuovere solo i clienti c1 e c2, infatti, quando stamperemo la lista avremo che solo i clienti

c1 e c2 saranno avranno l'attributo patente pari a true, mentre c3 lo avrà pari a false, per tale motivo verificheremo tramite 3 assert che c3, che è stato bocciato, avrà l'attributo NumeroBocciatureEsameFinale > 0. Ripetendo nuovamente la stessa operazione, c3 avrà un numero di bocciature pari a 2 e per tale motivo ci aspettiamo che l'attributo foglioRosa sia pari a false e che l'attributo numeroGuide sia pari a 0 (pociché bocciato 2 volte all'esame finale dovrà ripetere l'esame teorico).

- **Cliente (TestCliente.java):**
  - **testIncrementaFrequenza:** Per prima cosa creiamo un nuovo oggetto di tipo *Argomento* e lo passiamo come parametro nella funzione "*incrementaFrequenzaLezioni*" della classe *Cliente,* se tutto è andato a buon fine ci aspettiamo che l'attributo *frequenzaLezioni* di *Cliente* venga aggiornato poiché non ha ancora seguito l'argomento selezionato. Successivamente richiamiamo il metodo "*incrementaFrequenzaLezioni"* e passiamo come parametro un argomento già seguito dal cliente, ci aspettiamo che l'attributo *frequenzaLezioni* stavolta non venga aggiornato.
  - **testAggiornaFrequenzaLezioni:** Chiamiamo il metodo "*aggiornaFrequenzaLezioni*" della classe *Cliente e* passiamo come parametri 2 numeri interi, il test avrà esito positivo se l'attributo *frequenzaLezioni* di *Cliente* verrà aggiornato con un valore pari al rapporto dei 2 numeri inseriti.
  - **testIncrementaNumeroBocciature:** Inizialmente stampiamo in console l'attributo "numeroBocciature" dell'oggetto Cliente c, ci aspettiamo un valore pari a zero. Successivamente chiamiamo il metodo *"incrementaNumeroBocciature"* della classe Cliente e stampiamo nuovamente in console l'attributo. Ci accorgeremo che ogni volta che verrà chiamato tale metodo, il numero di bocciature sarà incrementato.
  - **testAggiornaNumeroGuide:** Inizialmente stampiamo in console l'attributo "*numeroGuide*" dell'oggetto Cliente c, ci aspettiamo un valore pari a zero. Successivamente chiamiamo il metodo *"aggiornaNumeroGuide"* della classe Cliente e stampiamo nuovamente in console l'attributo. Ci accorgeremo che ogni volta che verrà chiamato tale metodo, il numero di bocciature sarà incrementato.
  - **testIncrementaNumeroBocciatureEsameFinale:** Inizialmente stampiamo in console l'attributo "numeroBocciatureEsameFinale" dell'oggetto Cliente c, ci aspettiamo un valore pari a zero. Successivamente chiamiamo il metodo *"incrementaNumeroBocciatureEsameFinale"* della classe Cliente e stampiamo nuovamente in console l'attributo. Ci accorgeremo che ogni volta che verrà chiamato tale metodo, il numero di bocciature all'esame finale sarà incrementato.
- **EsameTeorico (TestEsameTeorico.java):**
  - **testIsDisponibile:** Chiamiamo il metodo *"isDisponibile"* della classe *EsameTeorico*, se tutto è andato a buon fine ci aspettiamo che il metodo ritoni *false* poiché la data dell'oggetto di tipo *EsameTeorico* chiamante è antecedente a quella odierna.
  - **testPrenotaCliente:** Per prima cosa creiamo un oggetto di tipo *Cliente*, il quale avrà quindi una frequenza lezioni nulla, e lo passiamo come parametro nella funzione *"prenotaCliente"* della classe *EsameTeorico*. Il test avrà esito positivo se l'elenco dei prenotati all'esame teorico sarà ancora vuoto. Successivamente incrementiamo l'attributo *frequenzaLezioni* dell'oggetto di tipo *Cliente* creato in precedenza attraverso il metodo *"incrementaFrequenzaLezioni"* e richiamiamo il metodo *"prenotaCliente"* dell'oggetto di tipo *EsameTeorico*. A questo punto ci aspettiamo che il cliente venga aggiunto alla mappa *elencoPrenotatiEsameTeorico* in quanto abbia un attributo *frequenzaLezioni* di dimensione sufficiente.
  - **testPromuoviCliente:** Per prima cosa creiamo 2 oggetti di tipo Cliente, aggiorniamo la loro frequenza lezioni così che possano essere prenotati all'esame attraverso la funzione *"prenotaCliente".* Successivamente chiamiamo il metodo *"promuoviCliente"* della classe esameTeorico passando come parametro il codice fiscale del primo cliente inserito (c1), in tal modo noteremo che quest'ultimo avrà l'attributo "foglioRosa" settato a

true, mentre il cliente c2 avrà questo attributo che rimarrà a *false.* Infine, stampiamo la lista dei prenotati all'esame per vedere l'intero elenco dei prenotati con i relativi esiti.

- **testConfermaEsiti:** Per prima cosa creiamo 4 oggetti di tipo Cliente, aggiorniamo la loro frequenza lezioni così che possano essere prenotati all'esame attraverso la funzione *"prenotaCliente".* Successivamente chiamiamo il metodo *"promuoviCliente"* della classe esameTeorico passando come parametro il codice fiscale dei clienti che vogliamo promuovere (c1 e c3). Dopo di che chiamiamo la funzione "confermaEsiti", la quale incrementa l'attributo "numeroBocciature" per i clienti che hanno l'attributo "foglioRosa" pari a false. Infine, stampiamo la lista dei prenotati all'esame per vedere l'intero elenco dei prenotati con i relativi esiti e numero bocciature.

- **Guida(TestGuida.java):**
  - **testIsDisponibile:** Chiamiamo il metodo *"isDisponibile"* della classe *Guida*, se tutto è andato a buon fine ci aspettiamo che il metodo ritoni *false* poiché la data dell'oggetto di tipo *Guida* chiamante è antecedente a quella odierna.
  - **testPrenotaCliente:** Per prima cosa creiamo un oggetto di tipo *Cliente*, il quale non avrà ancora ottenuto il foglio rosa, e lo passiamo come parametro nella funzione *"prenotaCliente"* della classe *Guida*. Poiché è necessario il possesso del foglio rosa per poter effettuare le guide, ci aspettiamo che la prenotazione non vada a buon fine e che quindi l'elenco dei prenotati alla guida rimanga vuoto. In un secondo momento settiamo l'attributo *foglioRosa* del cliente a *true* e richiamiamo il metodo "*penotaCliente*" della classe Guida, in questo caso ci aspettiamo che la prenotazione vada a buon fine e che l'elenco dei prenotati alla guida non sia vuoto.
  - **TestIsAntecedente:** Chiamiamo il metodo *"isAntecedente"* della classe *Guida*, se tutto è andato a buon fine ci aspettiamo che il metodo ritoni *true* poiché la guida ha una data antecedente al momento della chiamata della funzione.
- **EsameFinale(TestEsameFinale.java):**
  - **testIsDisponibile:** Chiamiamo il metodo *"isDisponibile"* della classe *Attività*, se tutto è andato a buon fine ci aspettiamo che il metodo ritoni *false* poiché la data dell'oggetto di tipo *EsameFinale* chiamante è antecedente a quella odierna.
  - **testPrenotaCliente:** Per prima cosa creiamo un oggetto di tipo *Cliente*, il quale non avrà ancora un numero di guide sufficienti (< 15), e lo passiamo come parametro nella funzione *"prenotaCliente"* della classe *EsameFinale*. Poiché è necessario che il numero di guide sia >= 15 per poter effettuare la prenotazione all'esame finale, ci aspettiamo che la prenotazione non vada a buon fine e che quindi l'elenco dei prenotati all'esame finale rimanga vuoto. In un secondo momento settiamo l'attributo foglioRosa del cliente a true e l'attributo numeroGuide pari a 15, richiamiamo il metodo "penotaCliente" e in questo caso ci aspettiamo che la prenotazione vada a buon fine e che l'elenco dei prenotati all'esame finale non sia vuoto.
  - **testPromuoviCliente:** Prima di tutto inseriamo due clienti c1 e c2. Poiché il cliente potrà prenotarsi solo se avrà superato il numero di guide richieste e se avrà il foglio Rosa pari a true, settiamo l'attributo numeroGuide pari a 15 e l'attributo fogioRosa pari a true. Successivamente promuoviamo solo il cliente c1 e ci accorgeremo che solo quest'ultimo avrà l'attributo patente pari a true, mentre c2 che non è stato promosso avrà l'attributo patente settato a false.
  - **testIsAntecedente:** Chiamiamo il metodo *"isAntecedente"* della classe *EsameFinale*, se tutto è andato a buon fine ci aspettiamo che il metodo ritorni *true* poiché l'esame finale ha una data antecedente al momento della chiamata della funzione.

- **testConfermaEsiti:** Per prima cosa creiamo 4 clienti, e per ognuno di essi settiamo il foglioRosa a true e il numero di guide pari a 15 così da poterli prenotare per l'esame attraverso la funzione "prenotaCliente" e promuoverli attraverso la funzione "promuoviCliente". In particolare, promuoviamo solo i clienti c1 e c3, dunque attraverso un assert verifichiamo che c1 e c3 abbiano l'attributo NumeroBocciatureEsameFinale pari a 0 mentre che c2 e c4 lo abbiano maggiore di 0. infine, stampiamo tutti i clienti.