03 – Elaborazione- Iterazione 2

2.1 Introduzione

Durante questa seconda iterazione ci si concentrerà su:

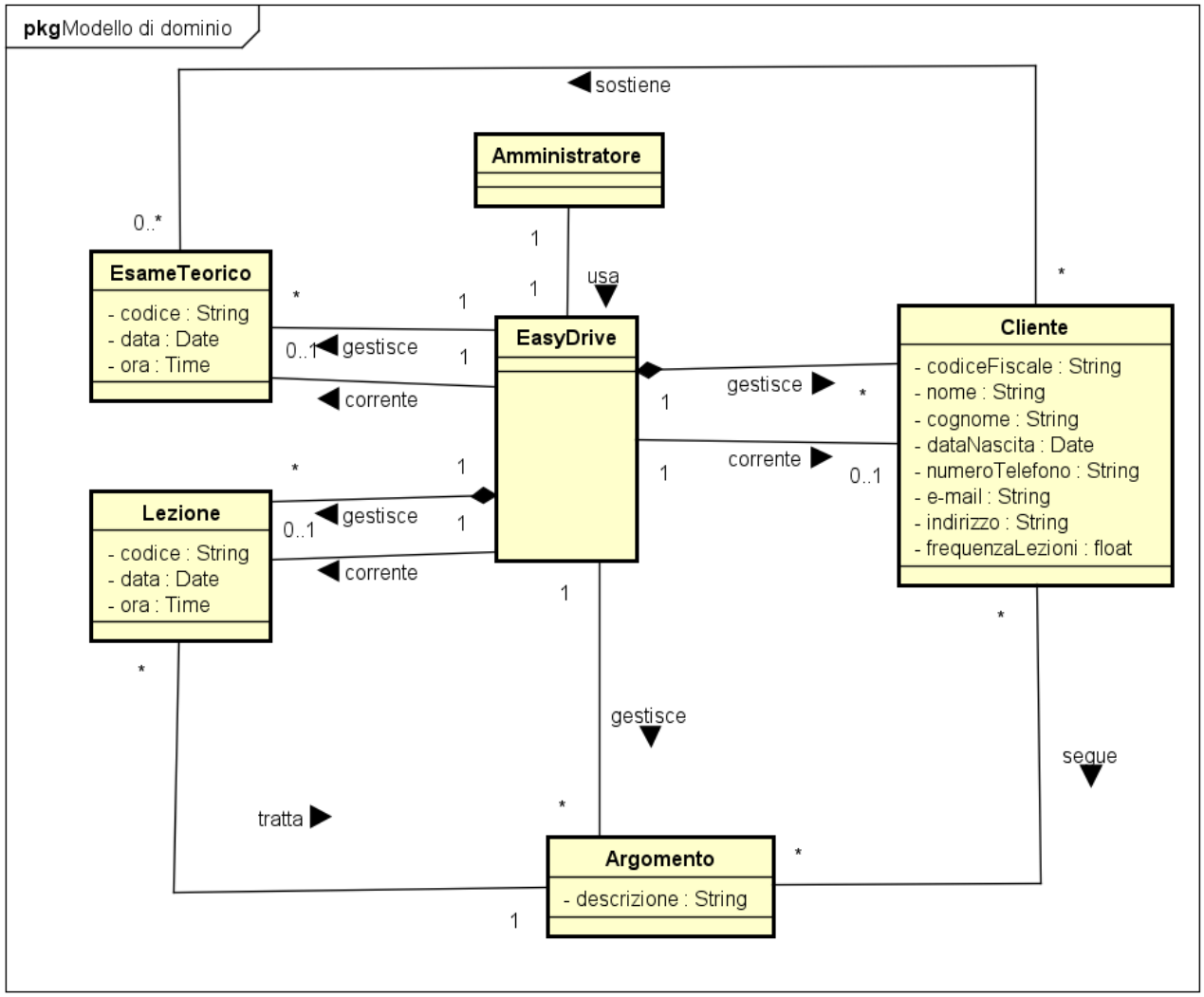
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate del caso d'uso **UC2**: ***Gestisci programmazione esame (CRUD);***
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate del caso d'uso **UC3**: **Prenota esame teorico;**

2.2 Analisi Orientata agli 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.

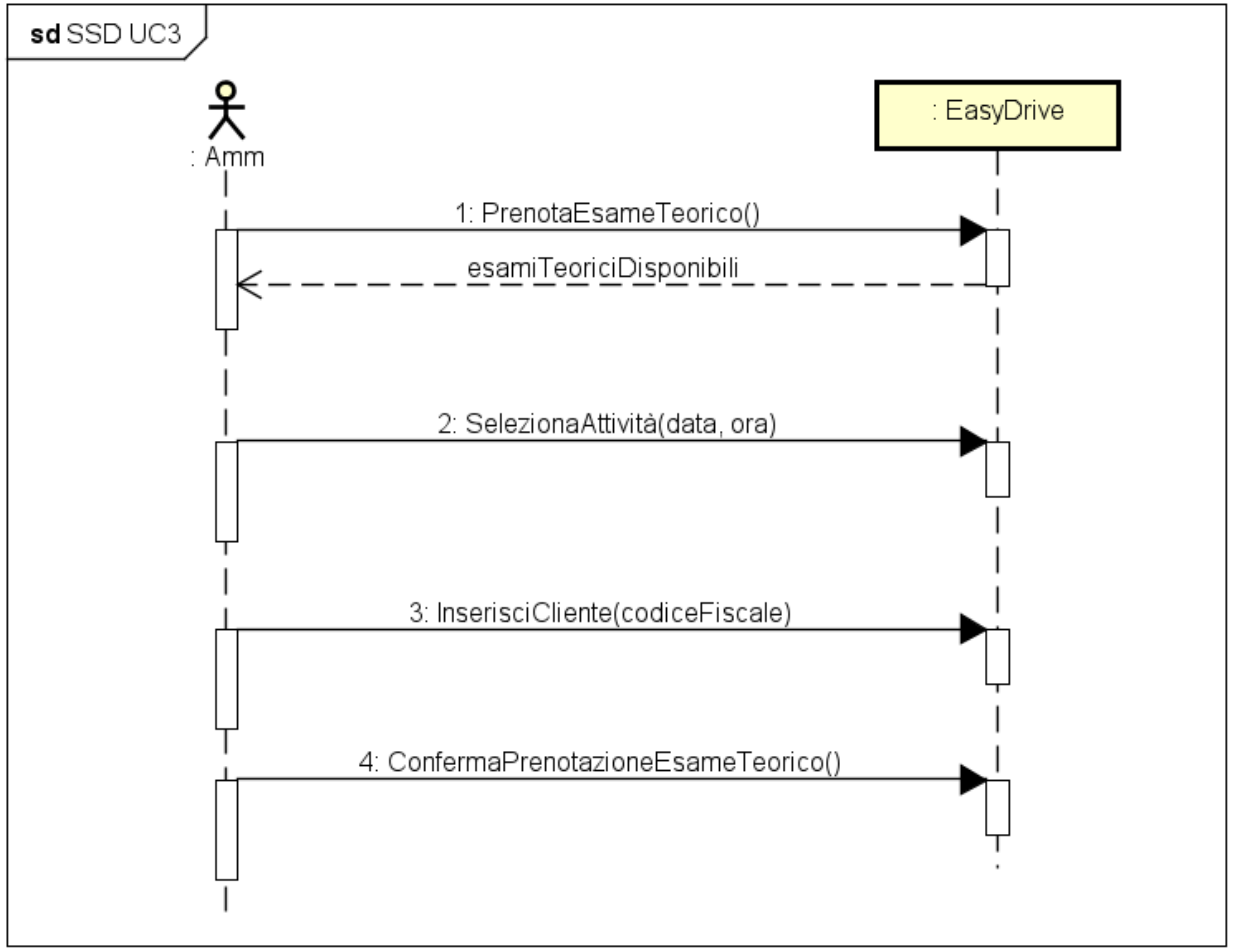
2.2.1 Modello di Dominio

Relativamente ai casi d'uso in esame (**UC2**, **UC3**), nasce l'esigenza di inserire una nuova classe concettuale: **EsameTeorico**. Tale necessità nasce per poter registrare nel sistema gli esami teorici e permettere ai clienti di sostenerli. Il Modello di Dominio, dunque, tenendo conto di associazioni e attributi, diventa:



2.2.2 Diagramma di sequenza di sistema

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



2.2.3 Contratti delle operazioni

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;

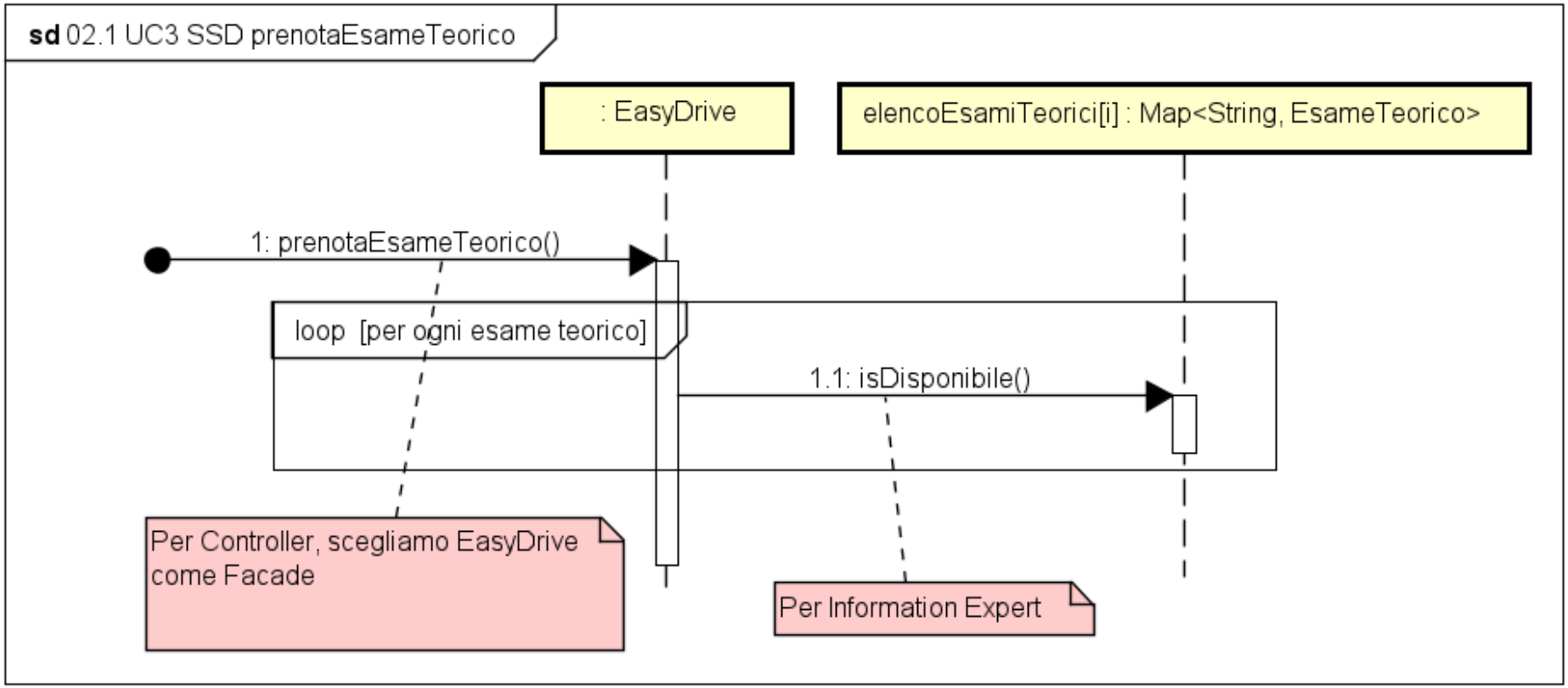
- "c" è stata associata ad EsameTeorico tramite l'associazione "sostiene"

2.3 Progettazione

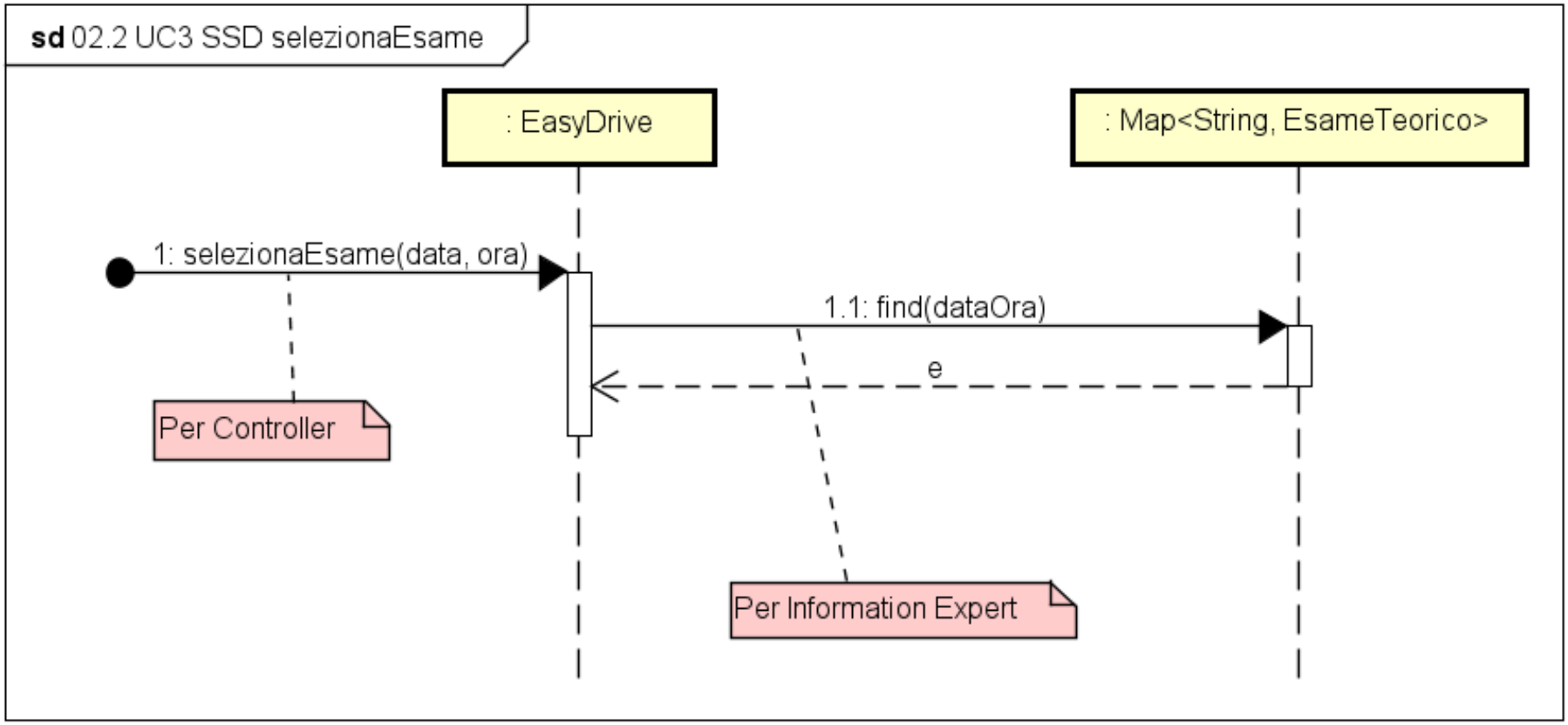
La progettazione orientata agli oggetti è la disciplina di UP interessata alla definizione degli oggetti software, delle loro responsabilità e a come questi collaborano per soddisfare i requisiti individuati nei passi precedenti. Seguono dunque i diagrammi di Interazione più significativi e il diagramma delle Classi relativi al caso d'uso **UC3** determinati a seguito di un attento studio degli elaborati scritti in precedenza.

2.3.1 Diagrammi di sequenza

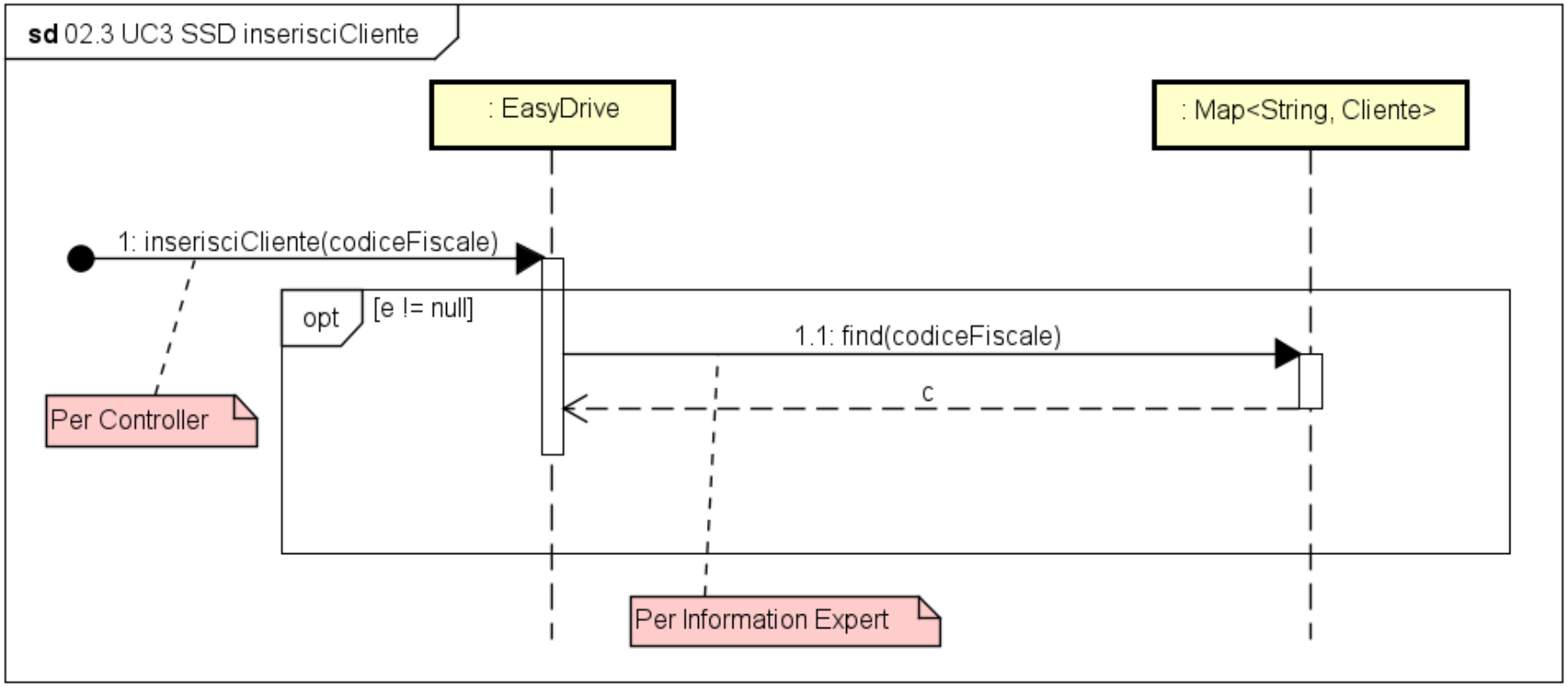
- **Prenota esame teorico**



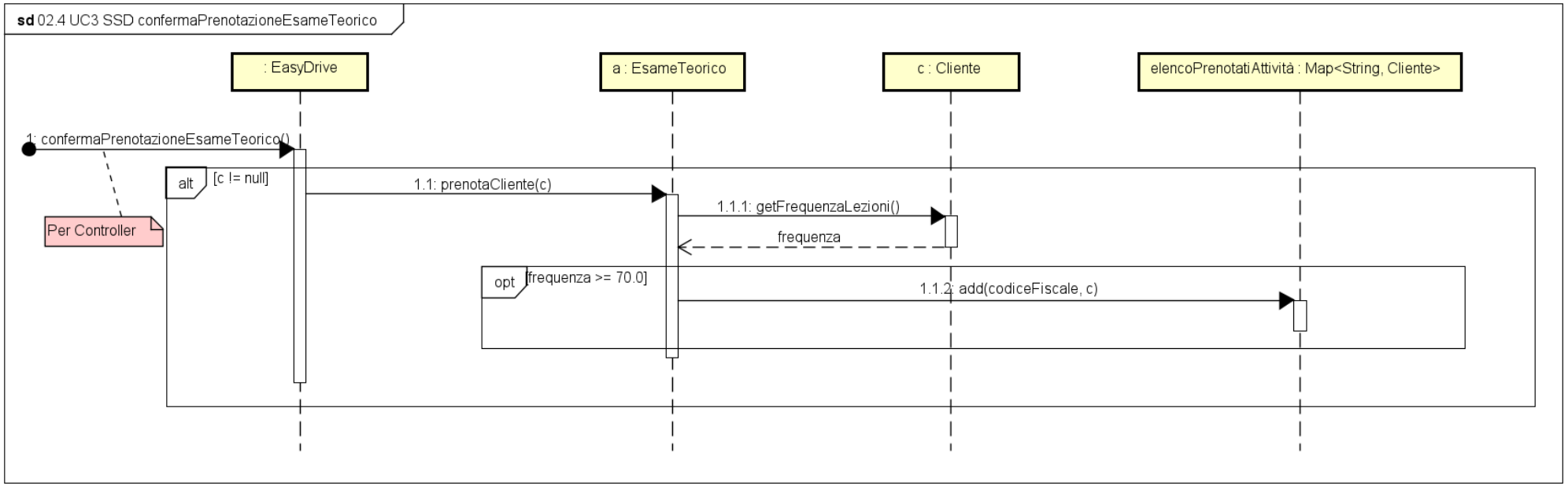
- **Seleziona esame**



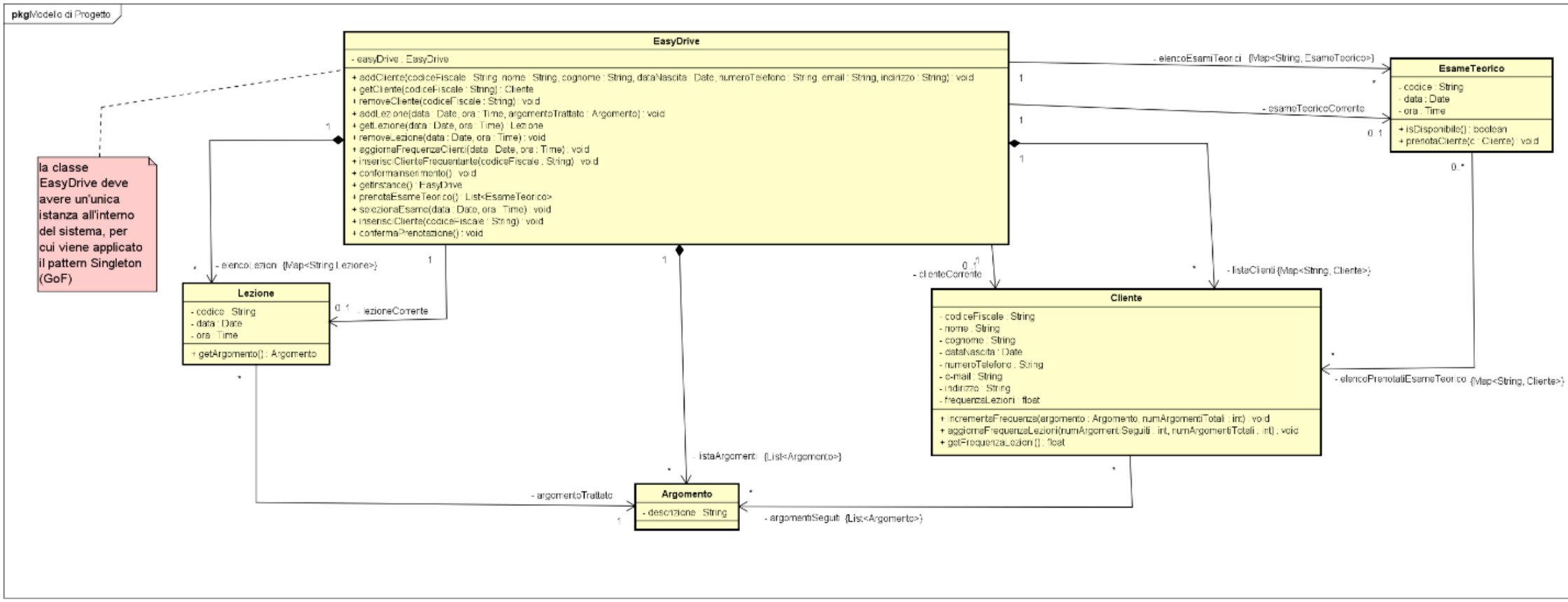
- **Inserisci cliente**



- **Conferma prenotazione Esame Teorico**



2.3.2 Diagramma delle Classi



2.4 Implementazione

Il codice è stato scritto in linguaggio Java utilizzando l'IDE Eclipse ed il framework di testing JUnit.

2.5 Test

Per verificare che i metodi e le classi da noi implementate siano funzionanti abbiamo creato dei test automatizzati. Segue un elenco puntato con la descrizione della metodologia di testing utilizzati per la seconda iterazione:

- **EasyDrive (testEasyDrive.java):**
  - **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*.

- **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.