04 – Elaborazione- Iterazione 3

2.1 Introduzione

Durante questa terza iterazione ci si concentrerà su:

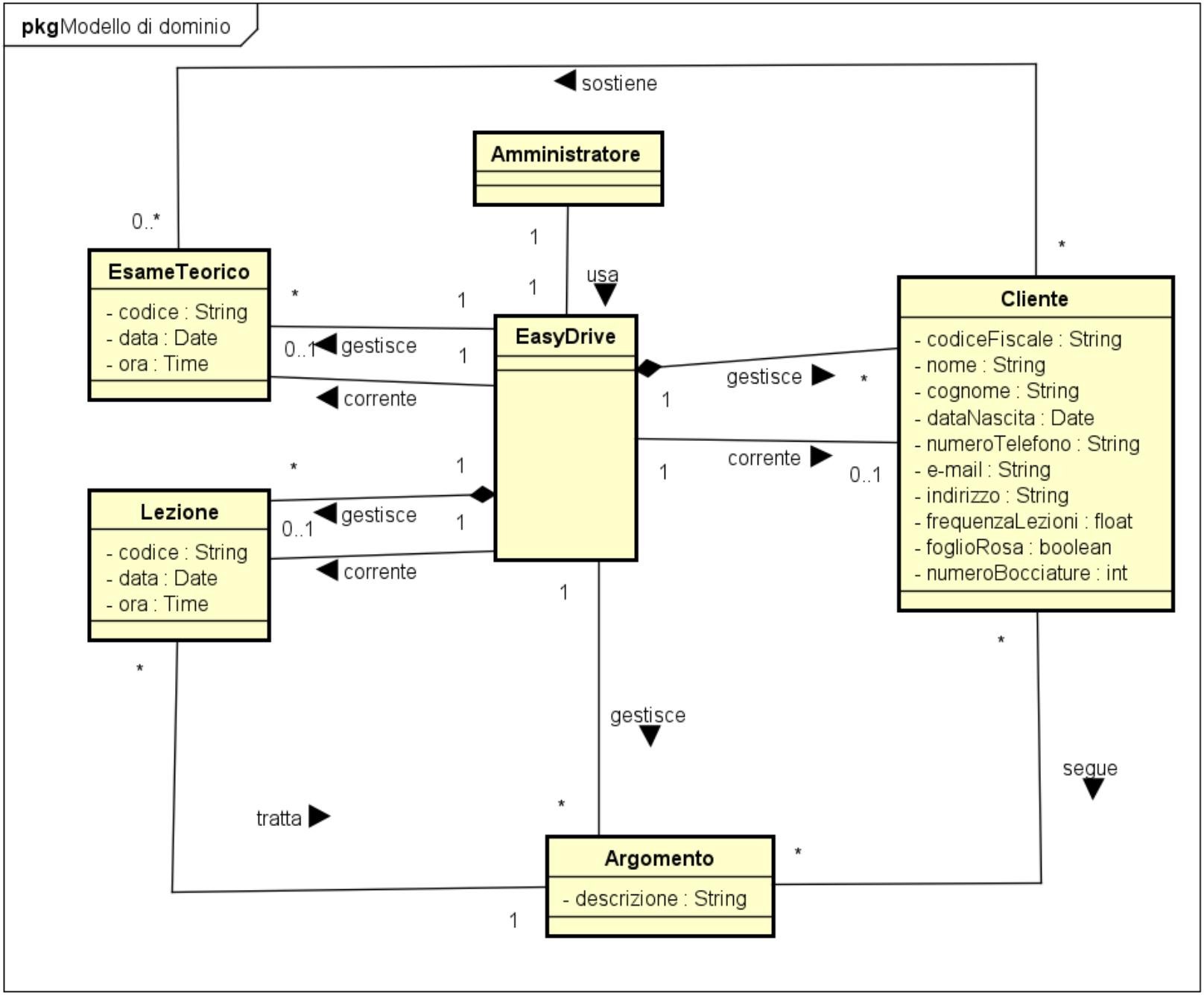
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso **UC10**: ***Pubblica esiti esame teorico***.

2.2 Analisi Orientata agli Oggetti

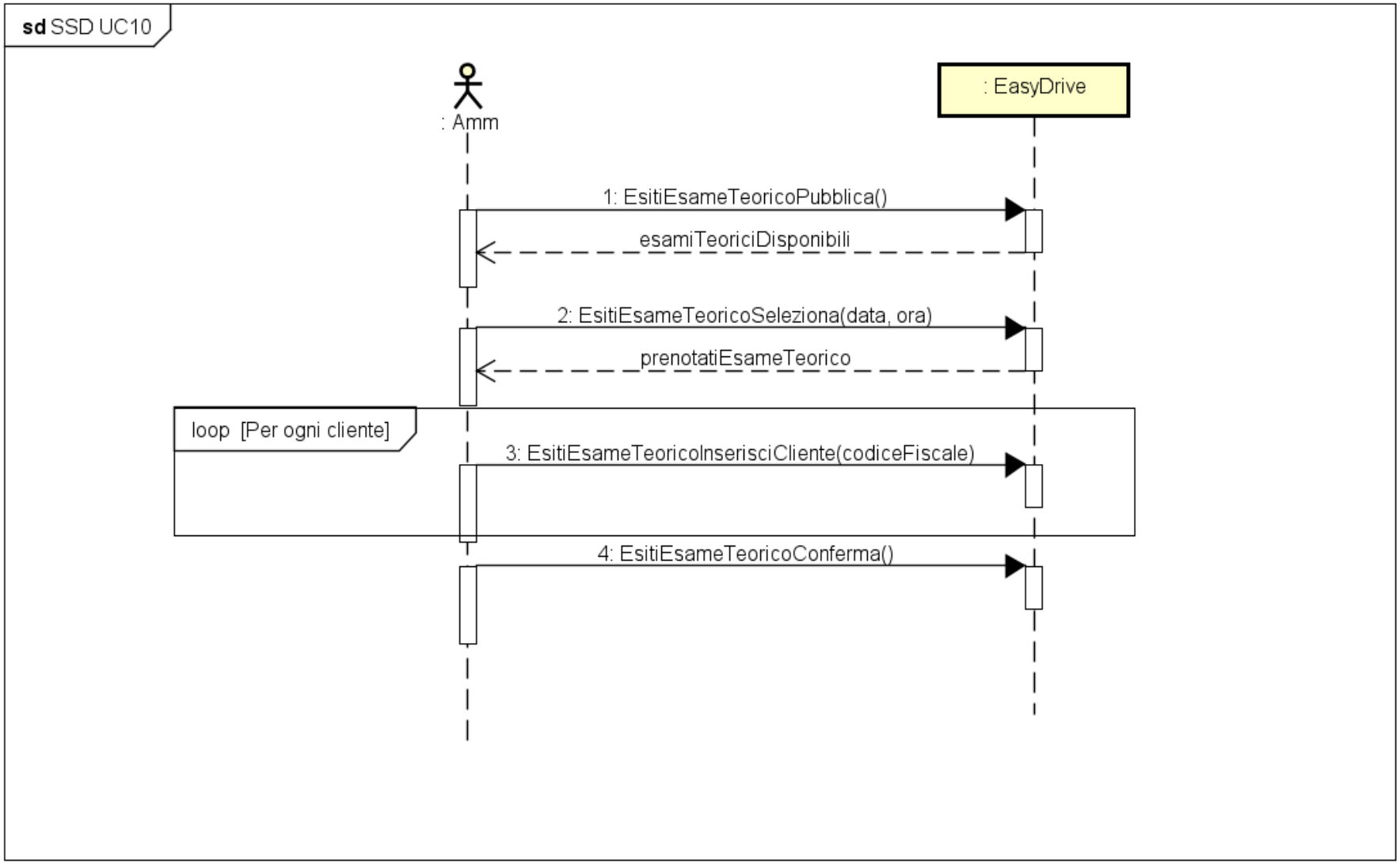
Al fine di descrivere il dominio da un punto di vista ad oggetti e gestire ulteriori requisiti, saranno utilizzati nuovamente gli stessi strumenti dell'iterazione precedente (Modello di Dominio, SSD - Sequence System Diagram e Contratti delle operazioni). In particolare, i paragrafi seguenti permettono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

2.2.1 Modello di Dominio

Relativamente al caso d'uso in esame **(UC10)**, nasce l'esigenza di inserire 2 nuovi attributi nella classe concettuale **Cliente**: *foglioRosa* e *numeroBocciature*. Tale necessità nasce per poter permettere al sistema di evidenziare quali clienti hanno preso il foglio rosa e quanti eventuali bocciature all'esame teorico hanno avuto. Il Modello di Dominio, dunque, tenendo conto di associazioni e attributi, diventa:


pkgModello di dominio
sostiene
Amministratore
0..*
1
1
usa
*
EsameTeorico
- codice : String
- data : Date
- ora : Time
*
1
0..1
gestisce
1
corrente
EasyDrive
*
Cliente
- codiceFiscale : String
- nome : String
- cognome : String
- dataNascita : Date
- numeroTelefono : String
- e-mail : String
- indirizzo : String
- frequenzaLezioni : float
- foglioRosa : boolean
- numeroBocciature : int
gestisce
*
1
corrente
1
0..1
*
1
Lezione
- codice : String
- data : Date
- ora : Time
0..1
gestisce
1
corrente
1
*
*
gestisce
segue
tratta
*
*
Argomento
- descrizione : String
1

2.2.2 Diagramma di sequenza di sistema



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: 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;

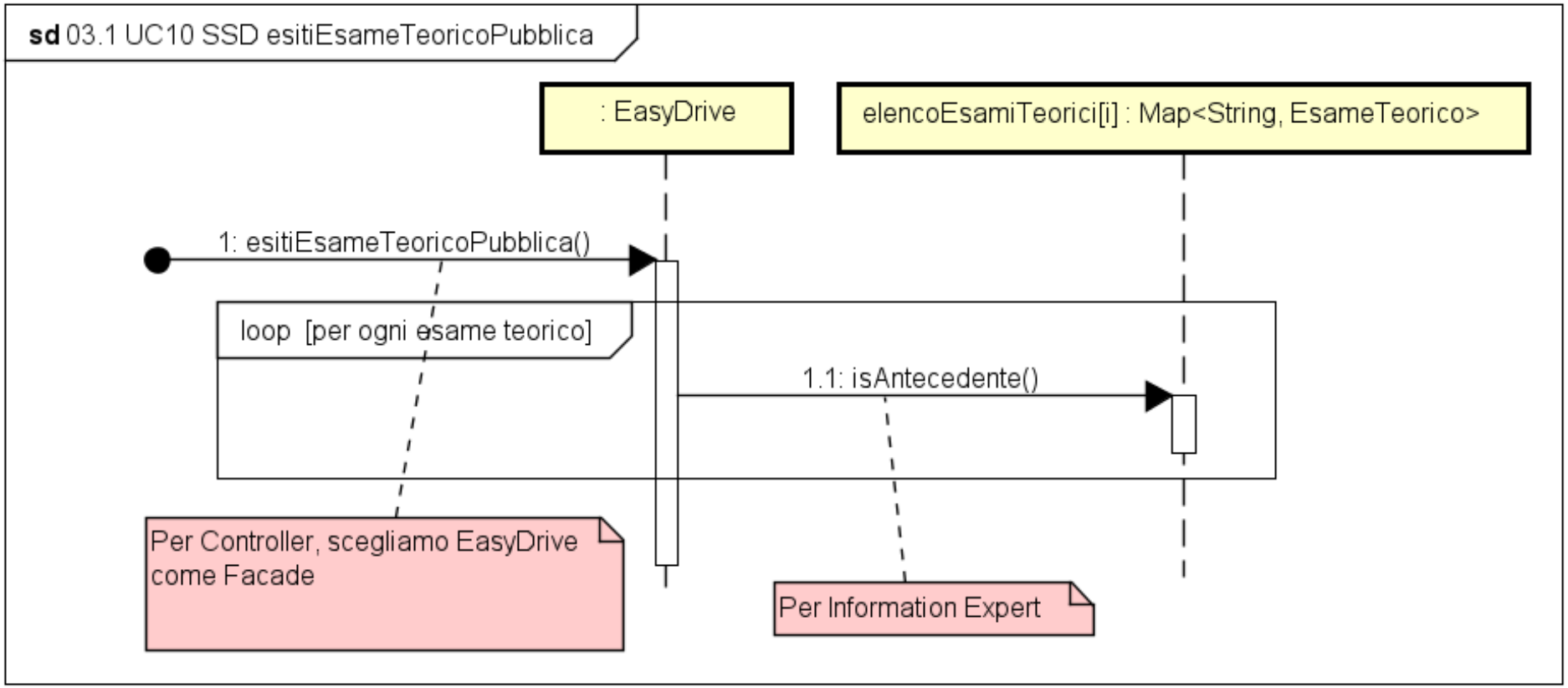
- È stato incrementato l'attributo numeroBocciature nel caso in cui l'attributo foglioRosa sia "false";
- "c" è stata associata ad EasyDrive tramite l'associazione "gestisce" nel caso in cui numeroBocciature sia maggiore o uguale a 2;

2.3 Progettazione

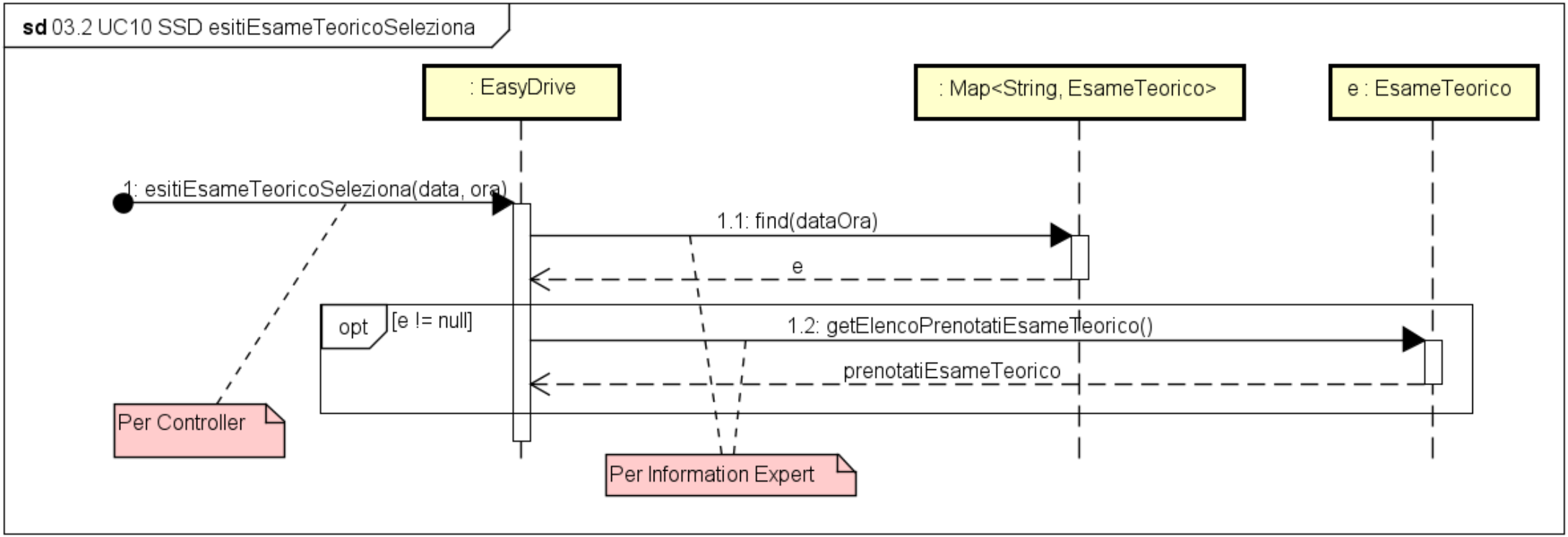
La progettazione orientata agli oggetti è la disciplina di UP interessata alla definizione degli oggetti software, delle loro responsabilità e a come questi collaborano per soddisfare i requisiti individuati nei passi precedenti. Seguono dunque i diagrammi di Interazione più significativi e il diagramma delle Classi relativi al caso d'uso **UC10** determinati a seguito di un attento studio degli elaborati scritti in precedenza.

2.3.1 Diagrammi di sequenza

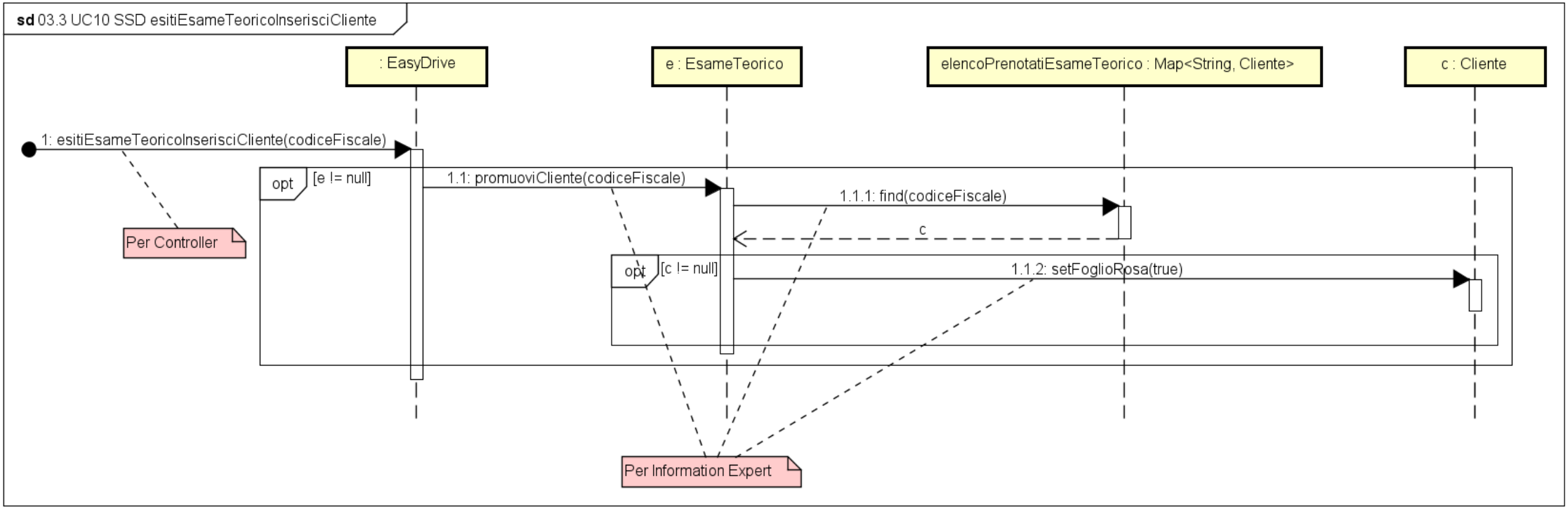
- **Esiti esame teorico pubblica**



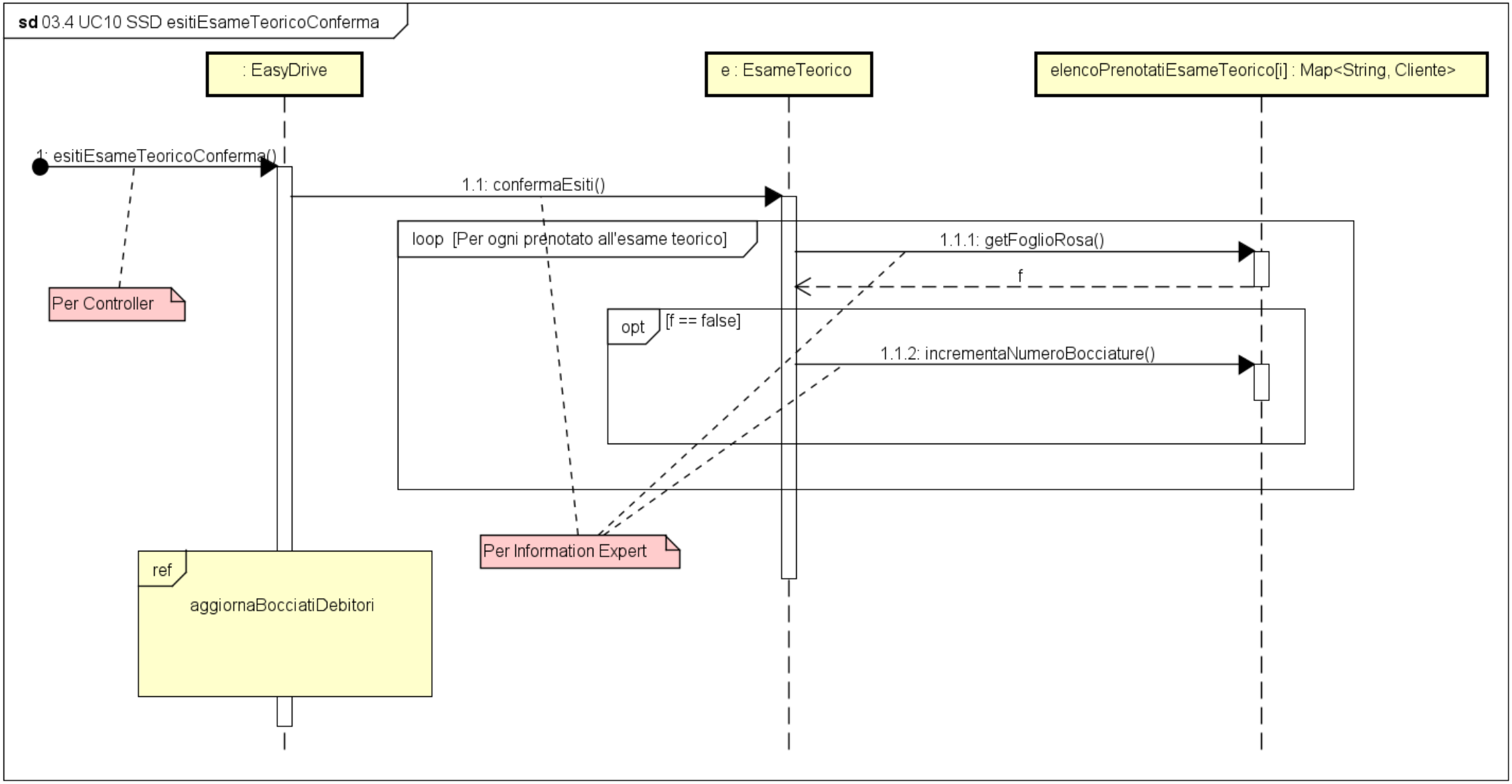
- **Esiti esame teorico seleziona**



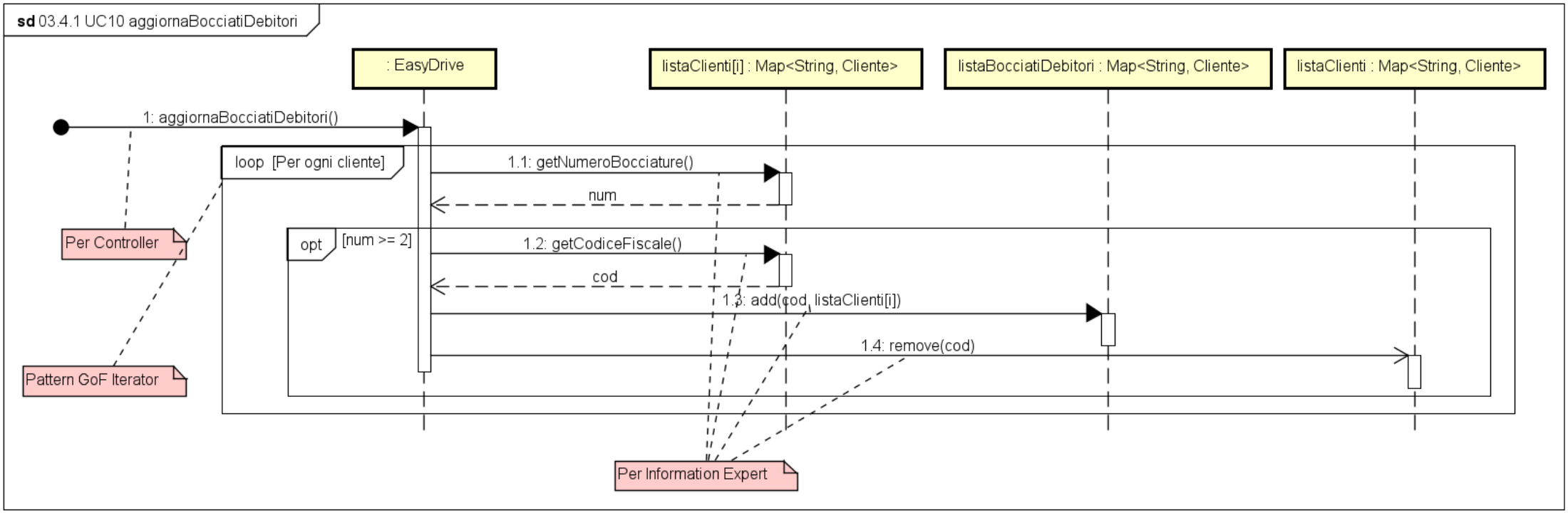
- **Esiti Esame teorico inserisci cliente**



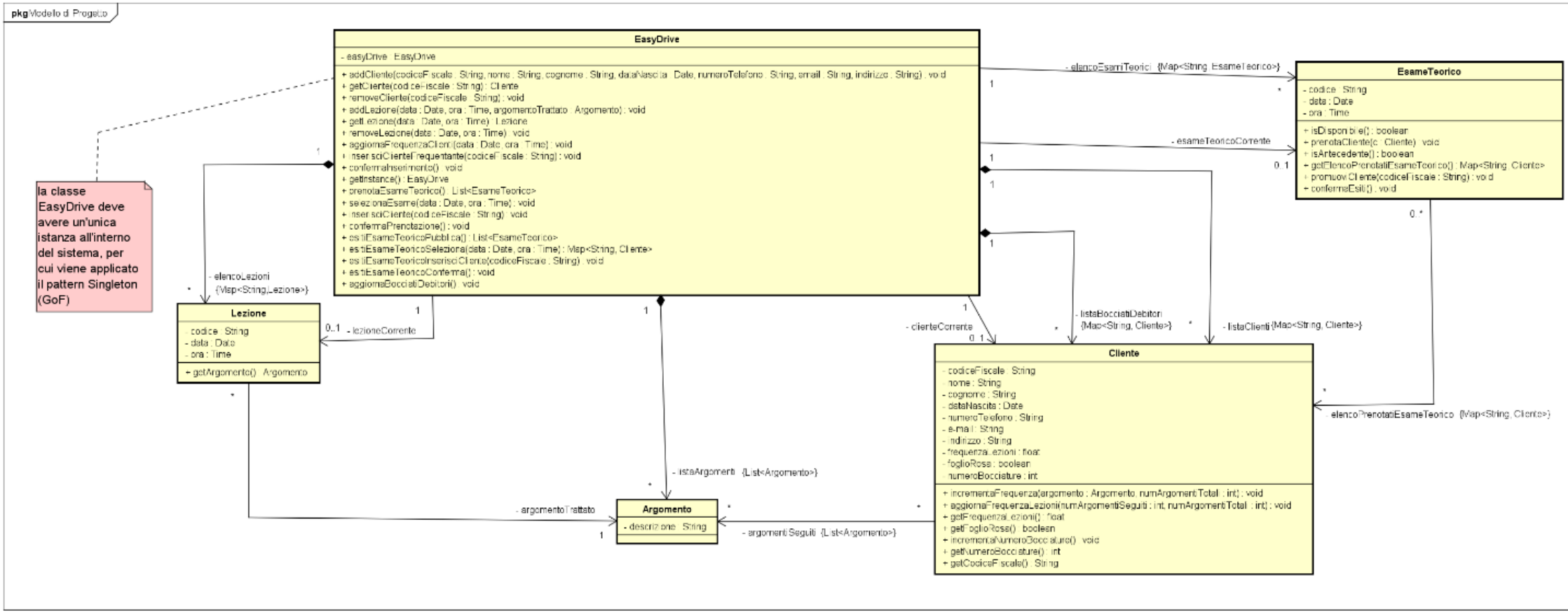
- **Esiti esame teorico conferma**



- **Aggiorna Bocciati Debitori**



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)**:
  - **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.

- **EsameTeorico (TestEsameTeorico.java):**
  - **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.
- **Cliente(TestCliente.java):**
  - **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.