02 – Elaborazione- Iterazione 1

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.

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.***

2.2 Analisi Orientata agli Oggetti

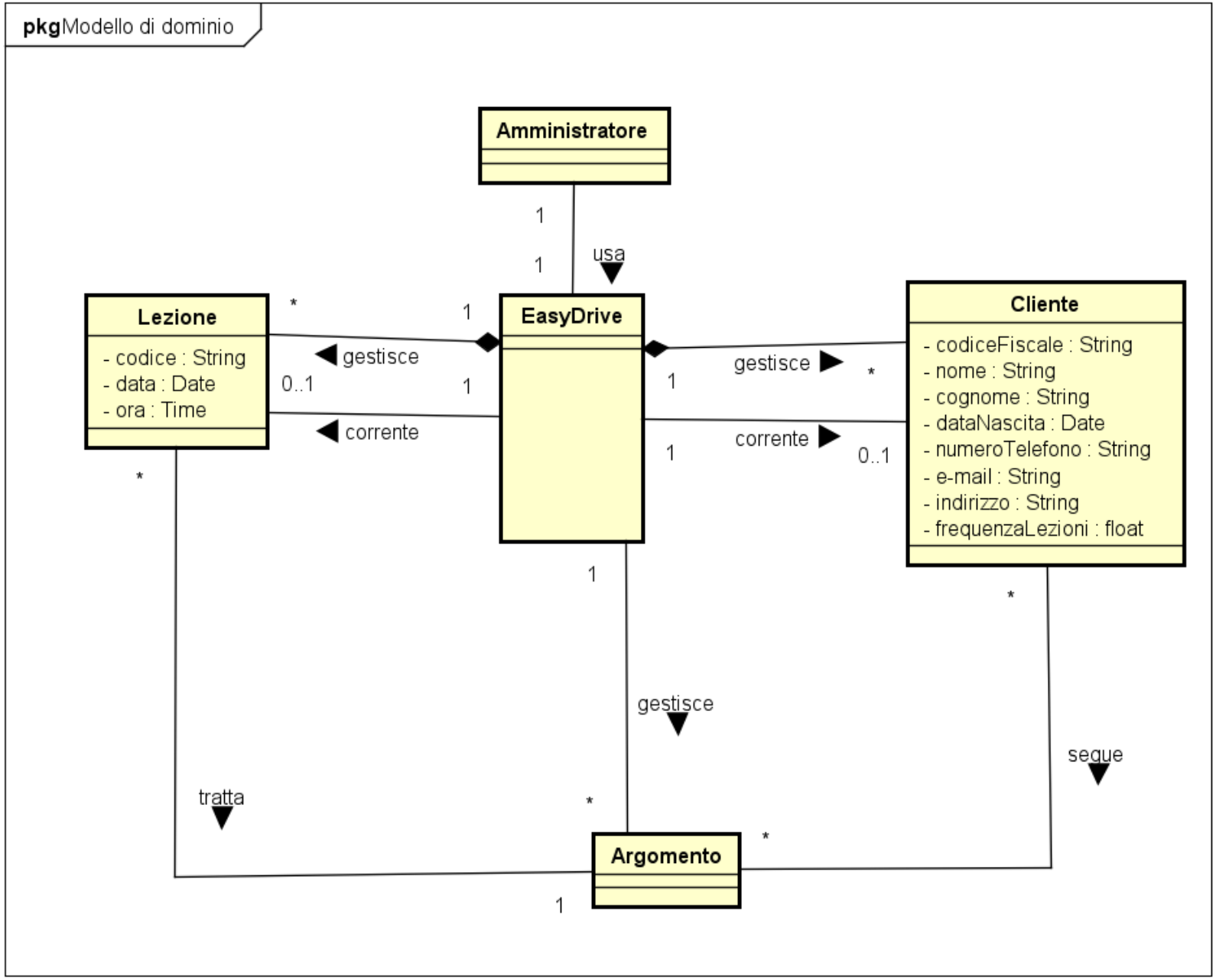
L'analisi orientata agli oggetti si basa sulla creazione di una descrizione del dominio da un punto di vista ad oggetti.

- Verranno utilizzati:
- Modello di Dominio *(paragrafo 2.2.1)*;
- Diagramma di sequenza di sistema [SSD] *(paragrafo 2.2.2)*;
- Contratti delle operazioni *(paragrafo 2.2.3)*;

2.2.1 Modello di Dominio

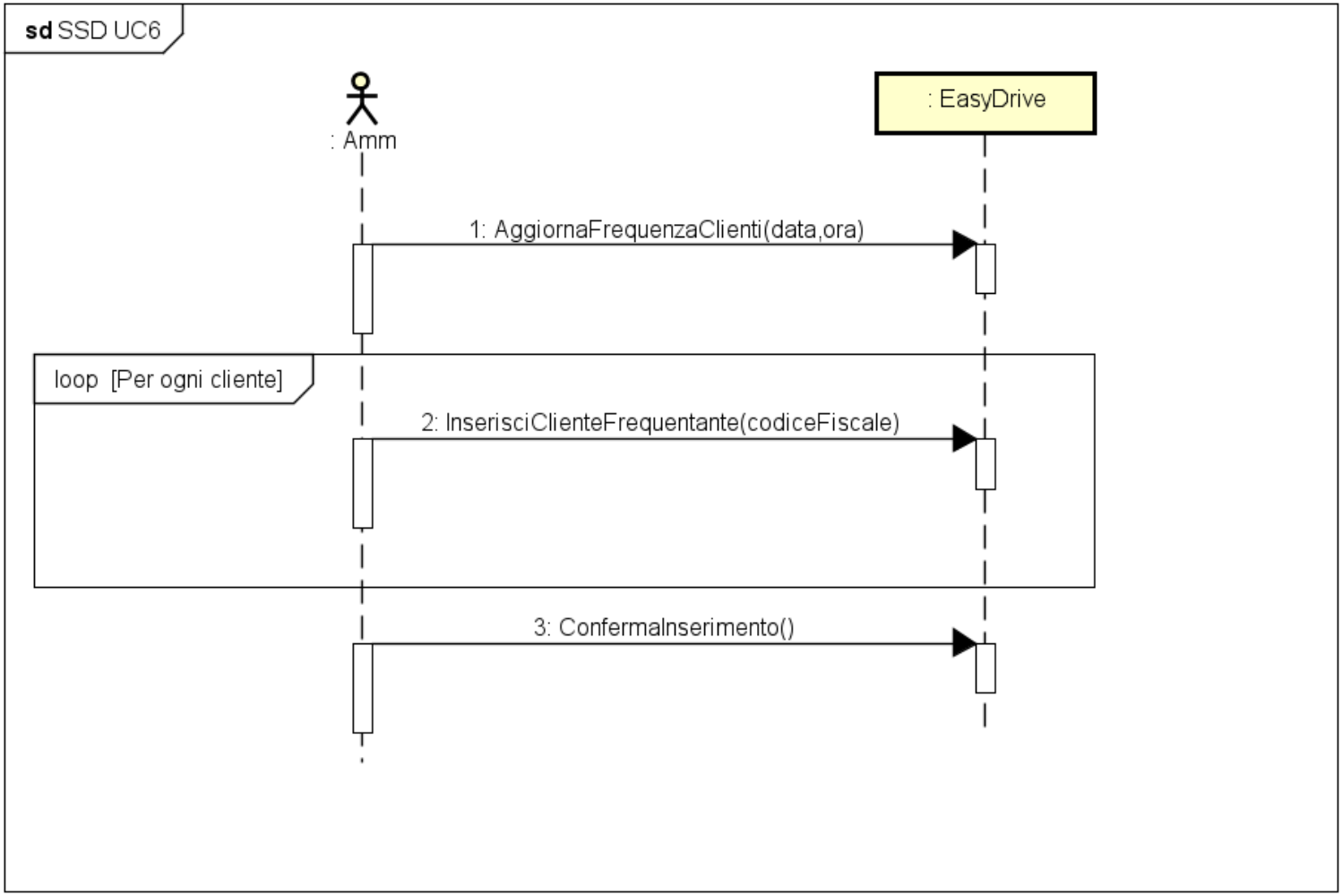
Relativamente ai casi d'uso in esame **(UC1, UC5, UC6)**, dopo la valutazione dello scenario principale di successo è stato possibile identificare le seguenti classi concettuali:

- **Amministratore:** attore primario, interagisce direttamente con il sistema;
- **EasyDrive:** rappresenta il sistema "EasyDrive";
- **Cliente:** cliente della scuola guida che sta seguendo la lezione corrente;
- **Lezione**: lezione corrente in cui si stanno prendendo le frequenze;
- **Argomento:** argomento trattato durante la lezione corrente.


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

2.2.2 Diagramma di sequenza di sistema

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

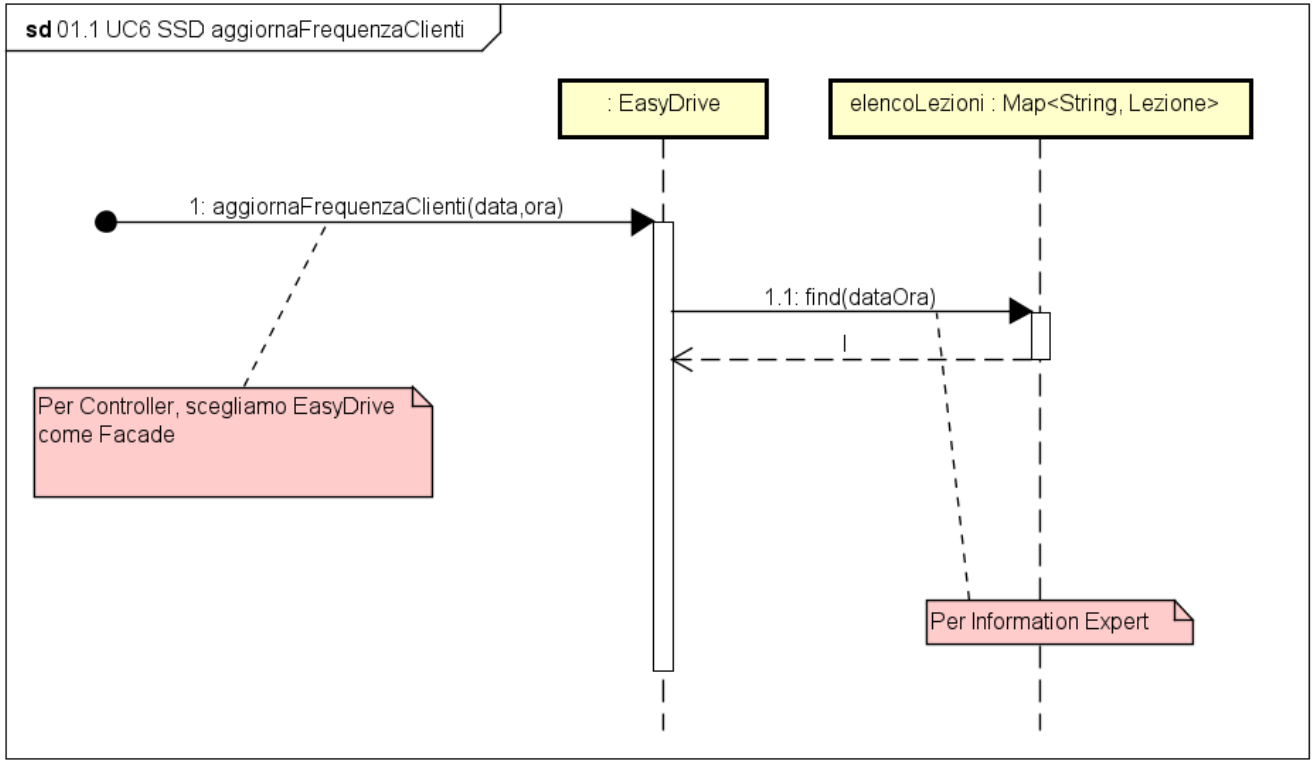
- È stata recuperata l'istanza numArgomentiTotali;
- L'istanza argomento è stata associata all'istanza c tramite l'associazione "segue";
- È stata recuperata l'istanza numArgomentiSeguiti;
- c.frequenzaLezioni è stato aggiornato.

2.3 Progettazione

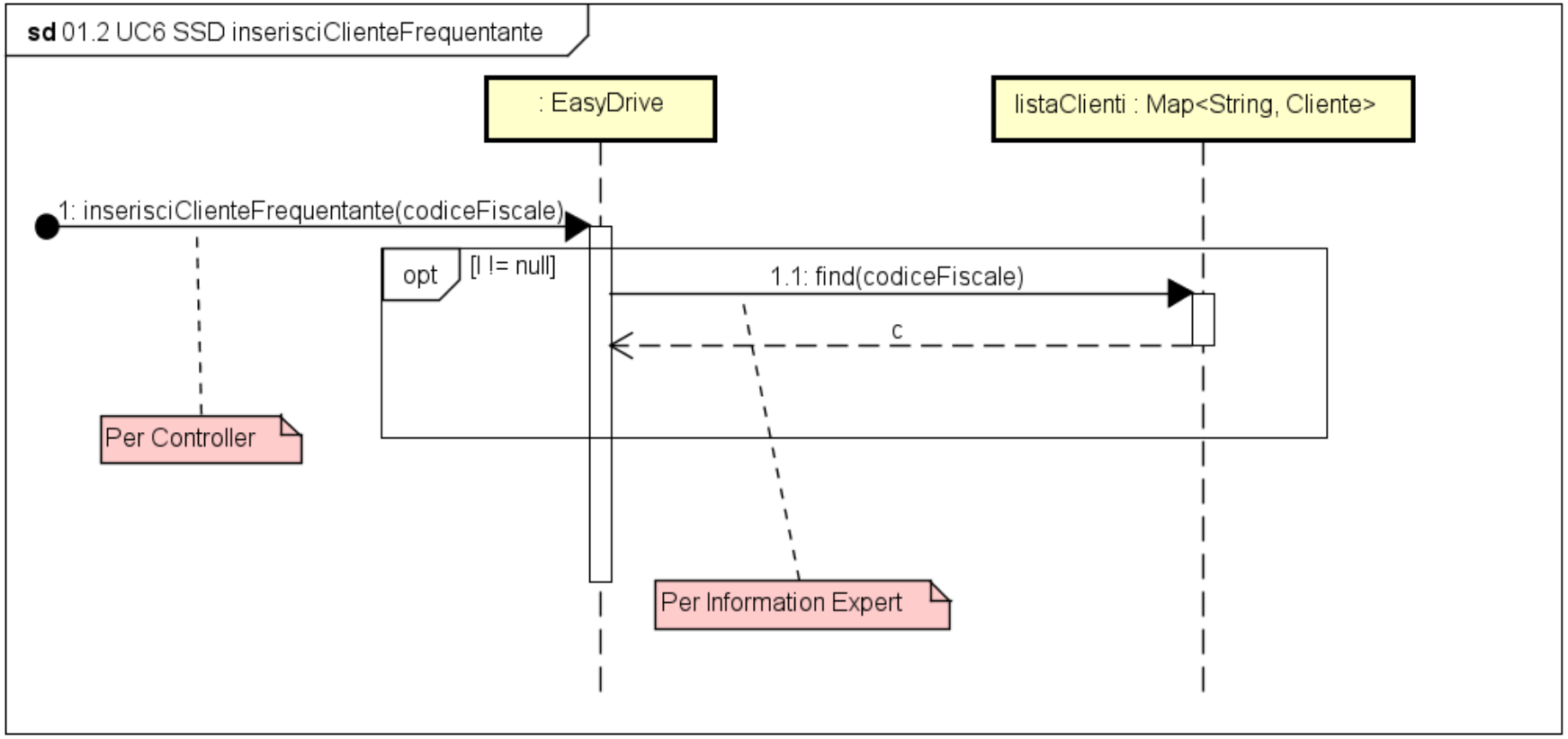
La progettazione orientata agli oggetti è la disciplina di UP interessata alla definizione degli oggetti software, delle loro responsabilità e a come questi collaborano per soddisfare i requisiti individuati nei passi precedenti. Seguono dunque i diagrammi di Interazione più significativi e il diagramma delle Classi relativi al caso d'uso UC6 determinati a seguito di un attento studio degli elaborati scritti in precedenza.

2.3.1 Diagrammi di sequenza

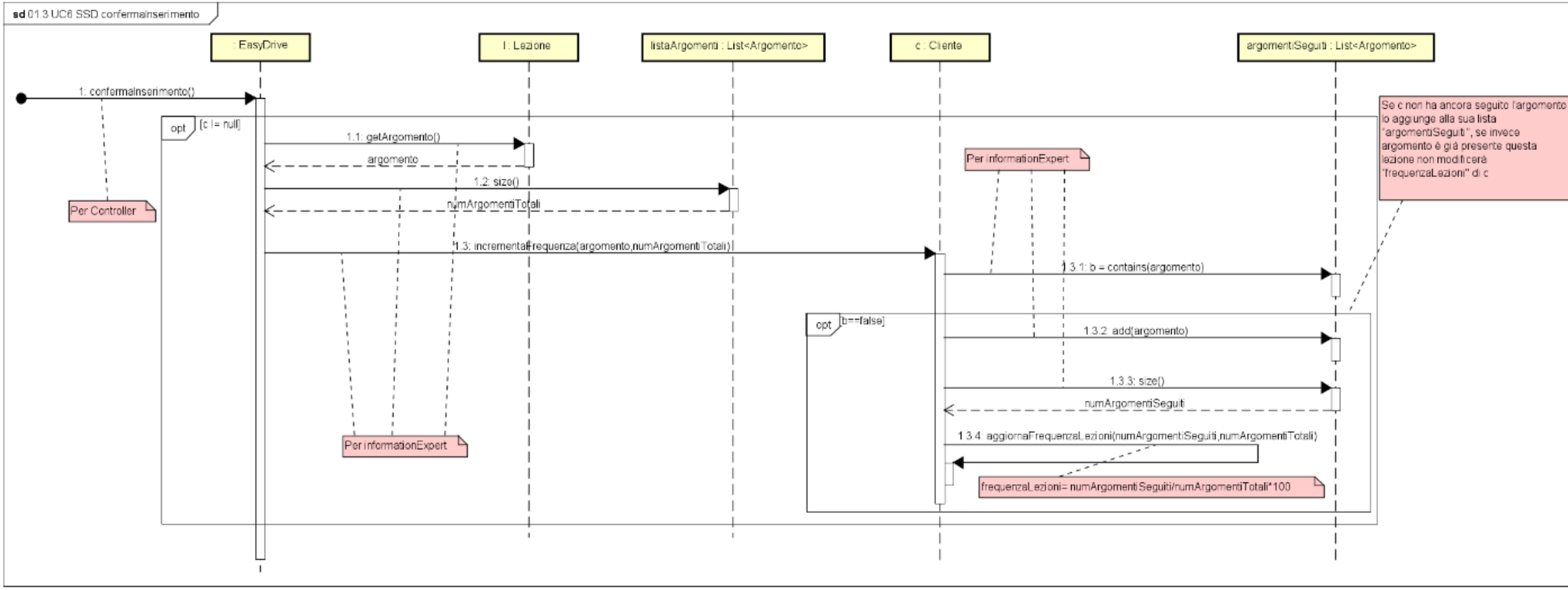
- **Aggiorna frequenza clienti**



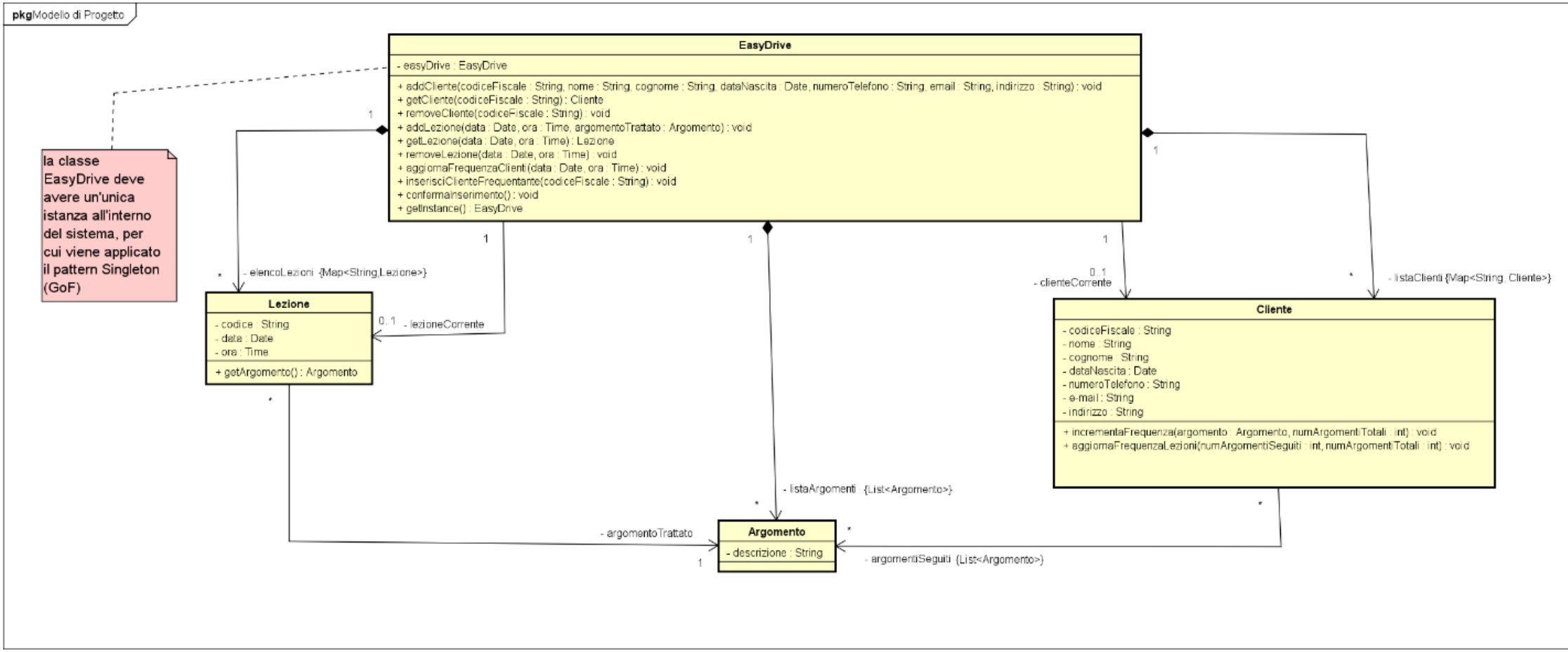
- **Inserisci cliente frequentante**



- **Conferma inserimento**



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.

- **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.

- **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.