05 – Elaborazione- Iterazione 5

2.1 Introduzione

Durante questa quarta iterazione ci si concentrerà su:

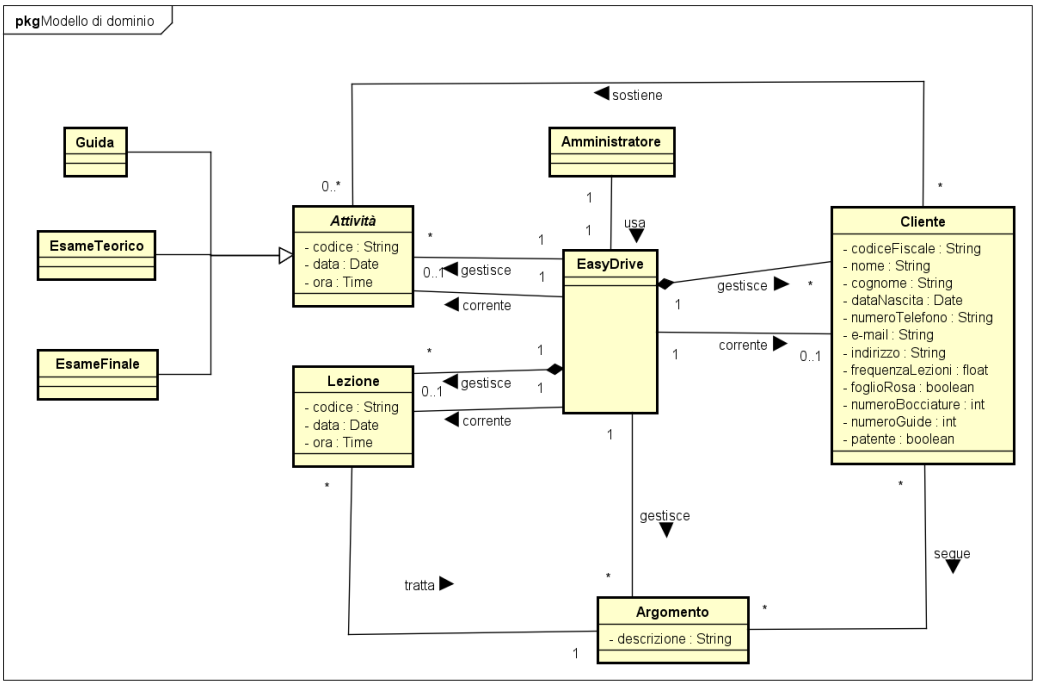
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso UC12: Gestisci programmazione esame finale (CRUD);
- Implementare lo scenario principale di successo e tutte le estensioni finora individuate riguardante il caso d'uso UC4: Prenota esame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

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

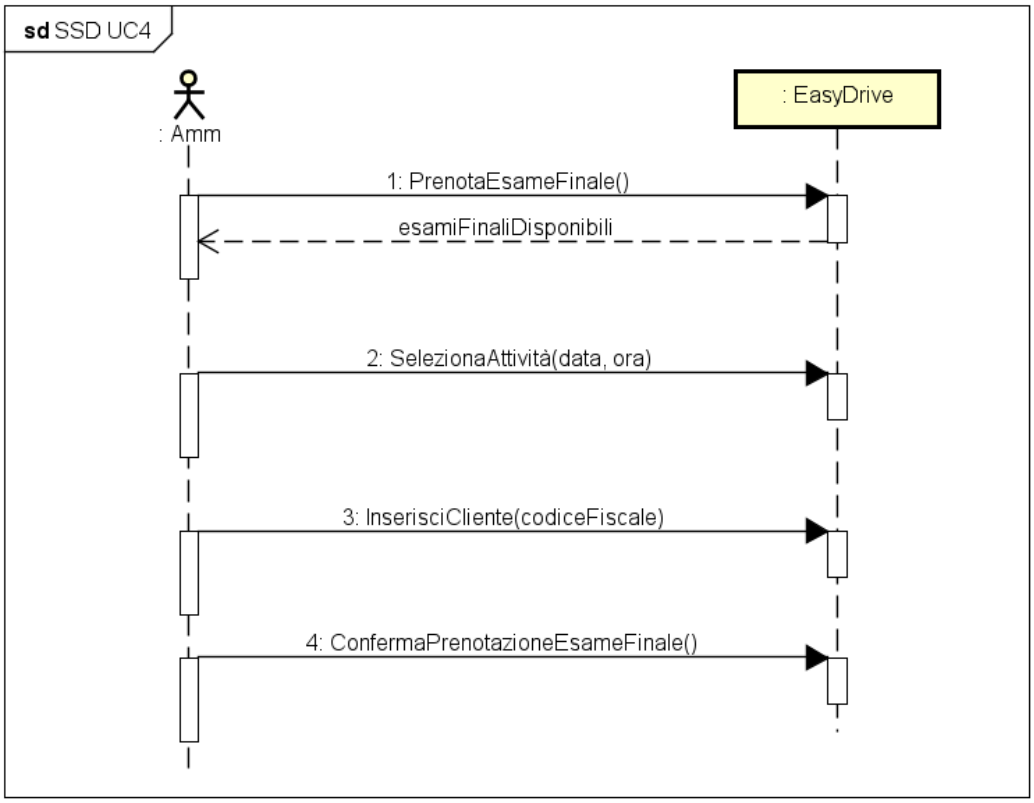
Relativamente ai casi d'uso in esame (UC12, UC4, UC11), nasce l'esigenza di creare una nuova classe "EsameFinale" e, poiché questa avrà caratteristiche comuni a "Guida" ed "EsameTeorico", abbiamo deciso di utilizzare questa classe come un'ulteriore generalizzazione di "Attività". Inoltre, è stato aggiunto anche l'attributo *patente* e *numeroBocciatureEsameFinale* in **Cliente.** Quest'ultimo attributo è stato aggiunto poiché si è considerata una nuova regola di dominio secondo la quale se l'esame finale non viene superato più di due volte si perderà la validità del foglio rosa poiché il cliente dovrà risostenere l'esame teorico.



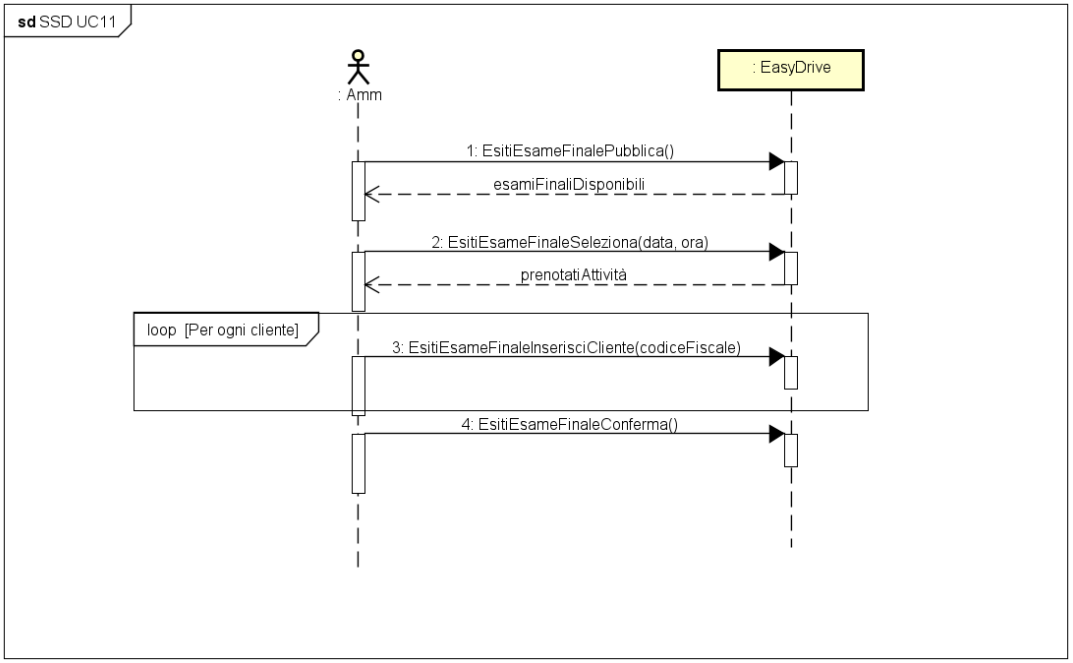
2.2.2 Diagramma di sequenza di sistema

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

SSD UC4:



SSD UC11:



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.

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 Finale selezionato;

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;

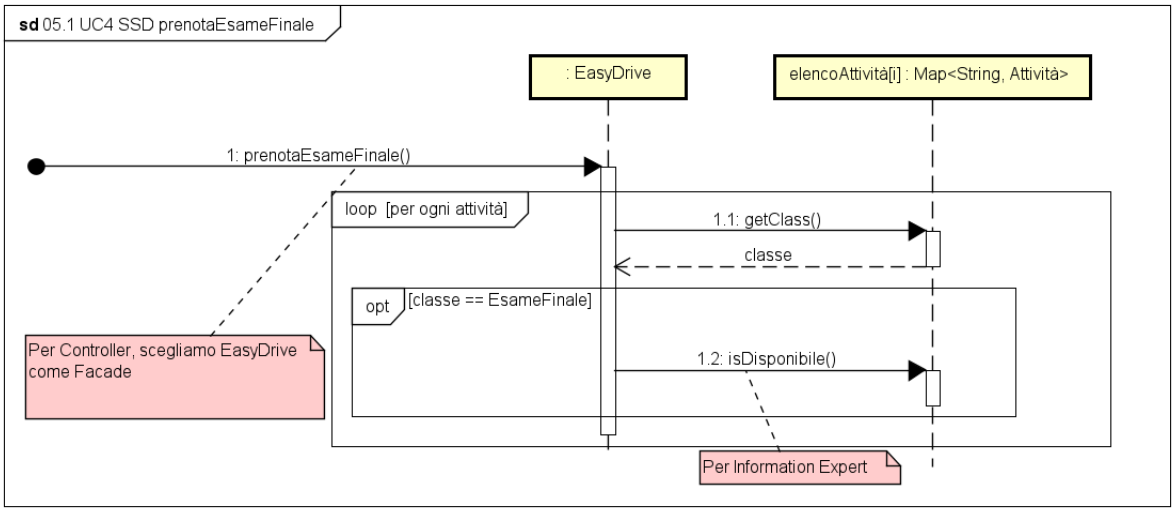
2.3 Progettazione

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 **UC4** e **UC11**, determinati a seguito di un attento studio degli elaborati scritti in precedenza.

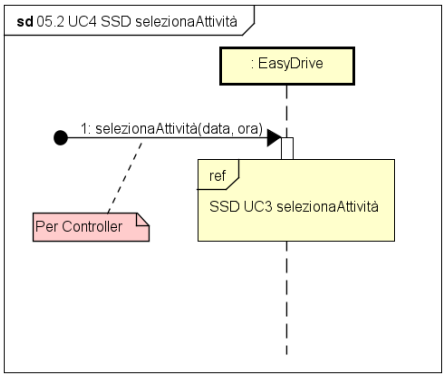
2.3.1 Diagrammi di sequenza

UC4:

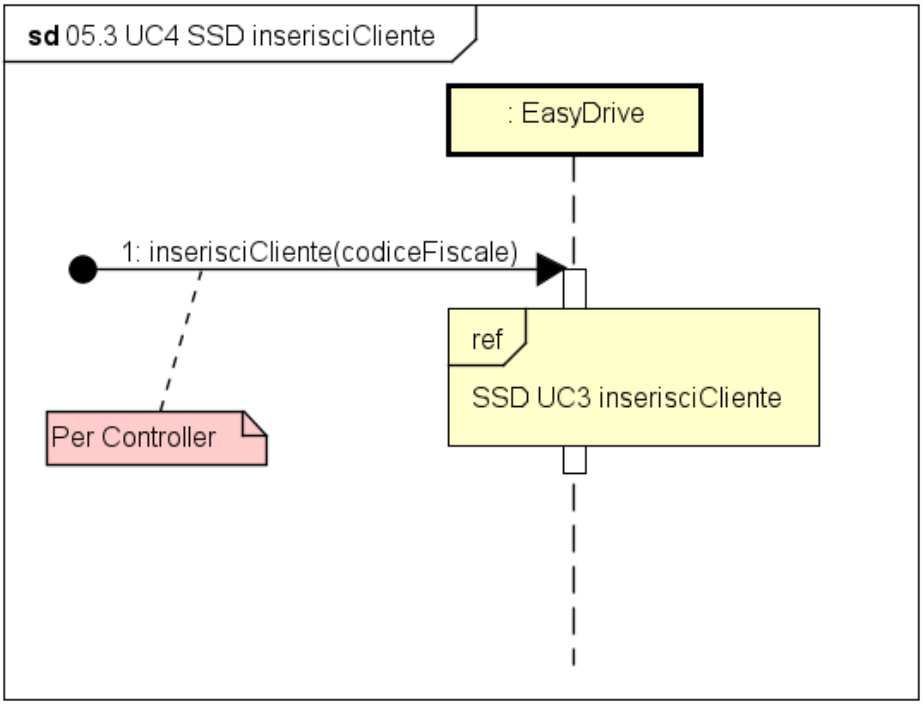
- **Prenota Esame Finale**



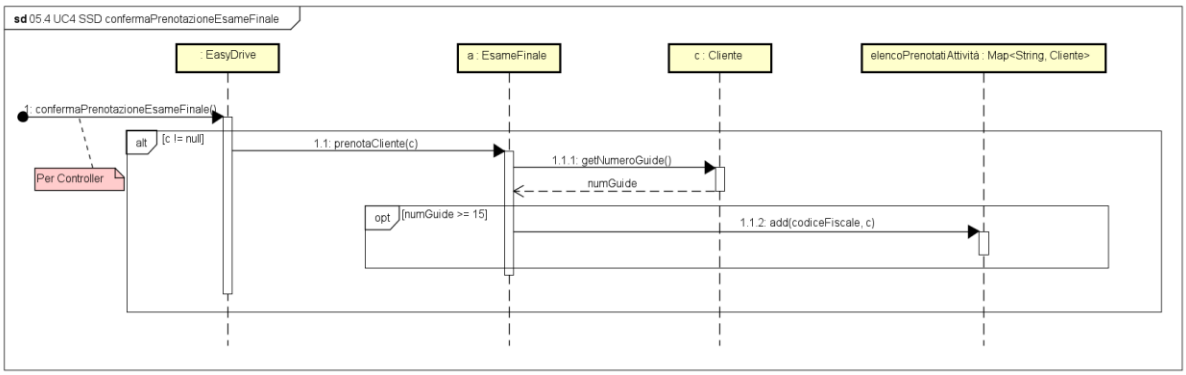
- **Seleziona Attività**



- **Inserisci Cliente**

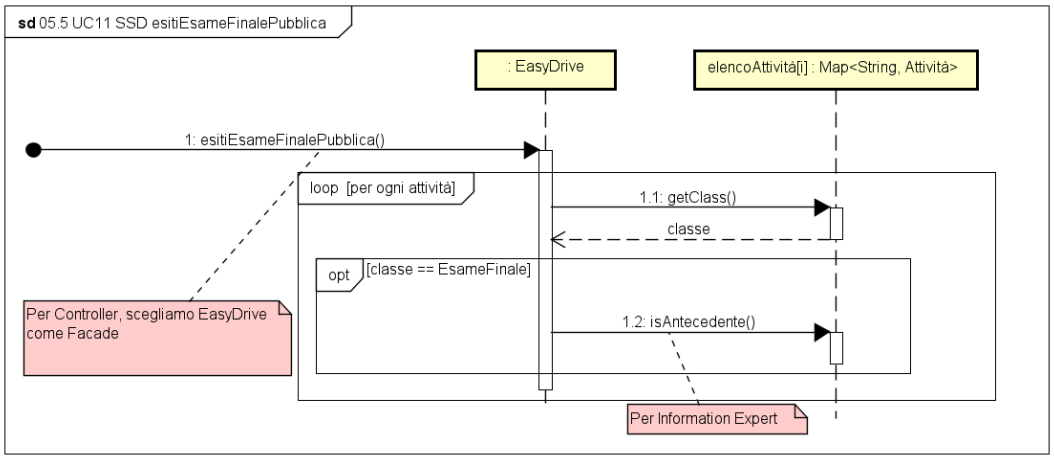


- **Conferma Prenotazione Esame Finale**

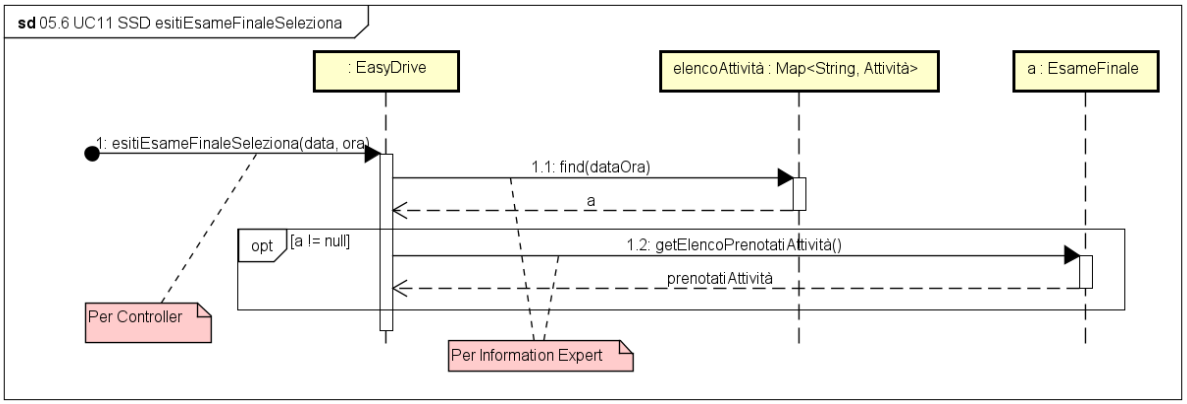


UC11:

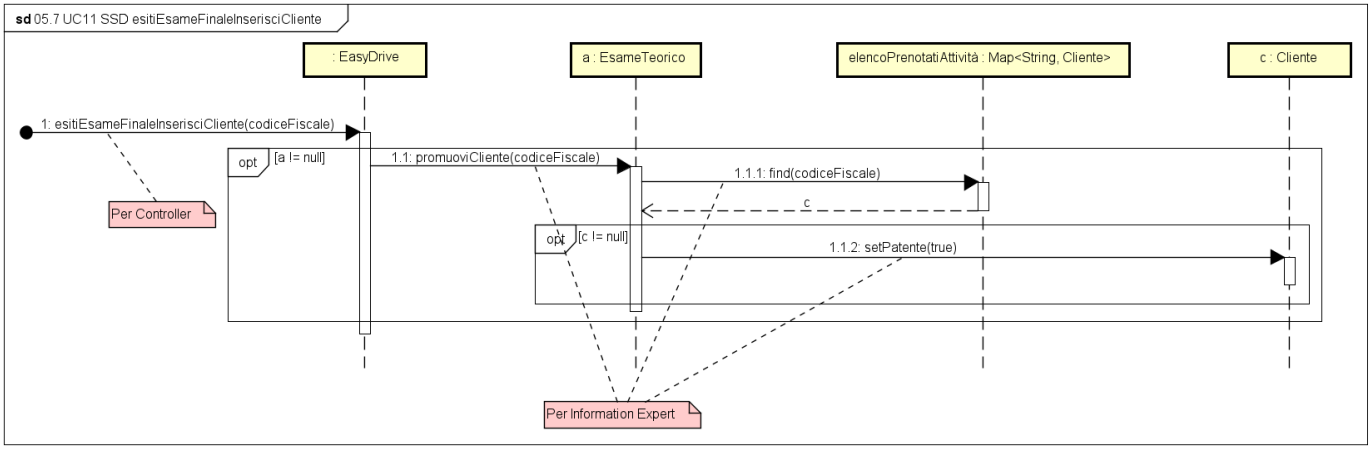
- **Esiti Esame Finale Pubblica**



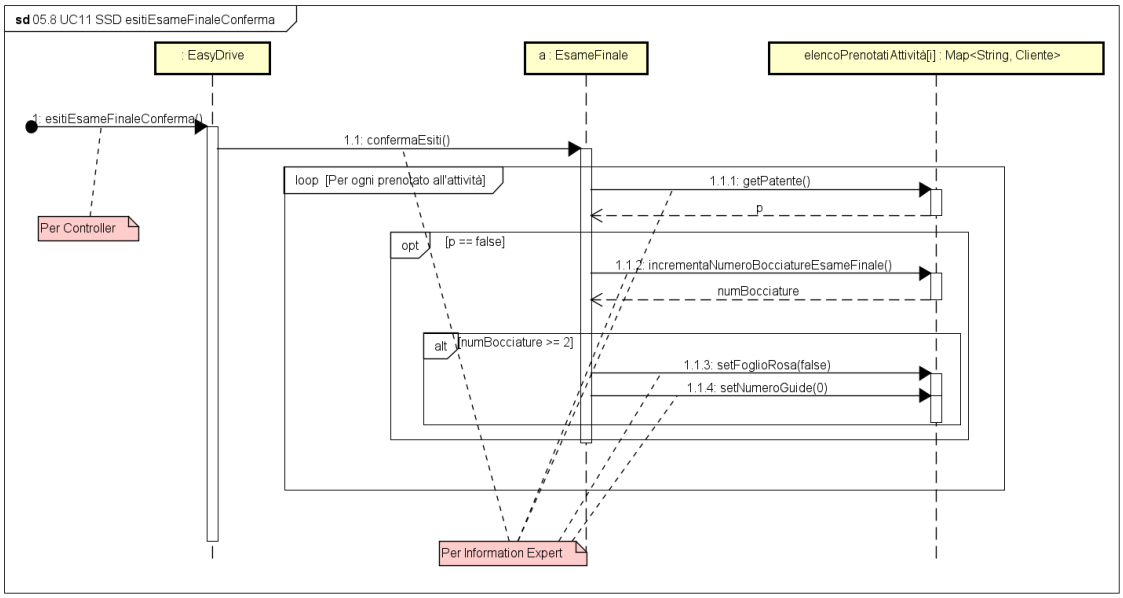
- **Esiti Esame Finale Seleziona**

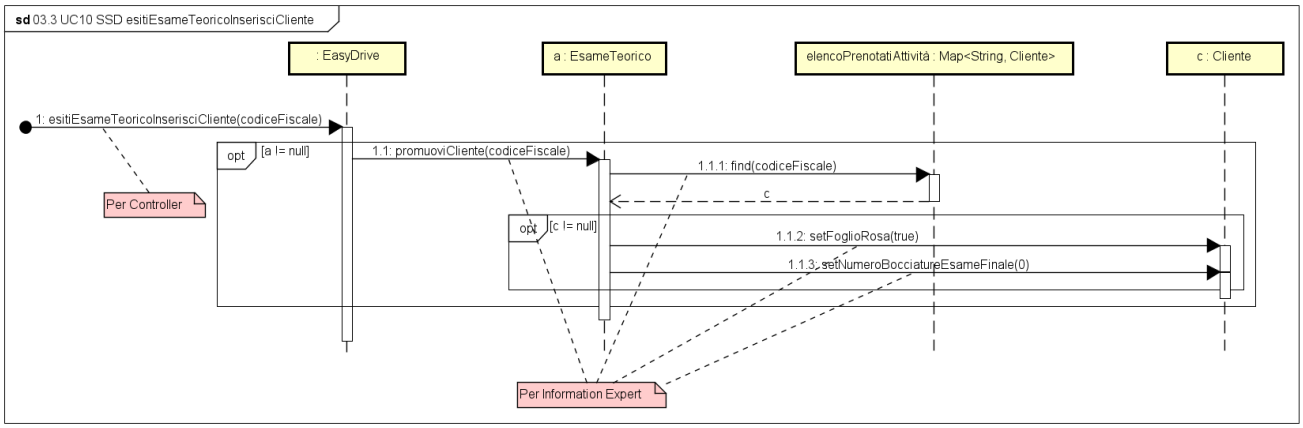


- **Esiti Esame Finale Inserisci Cliente**



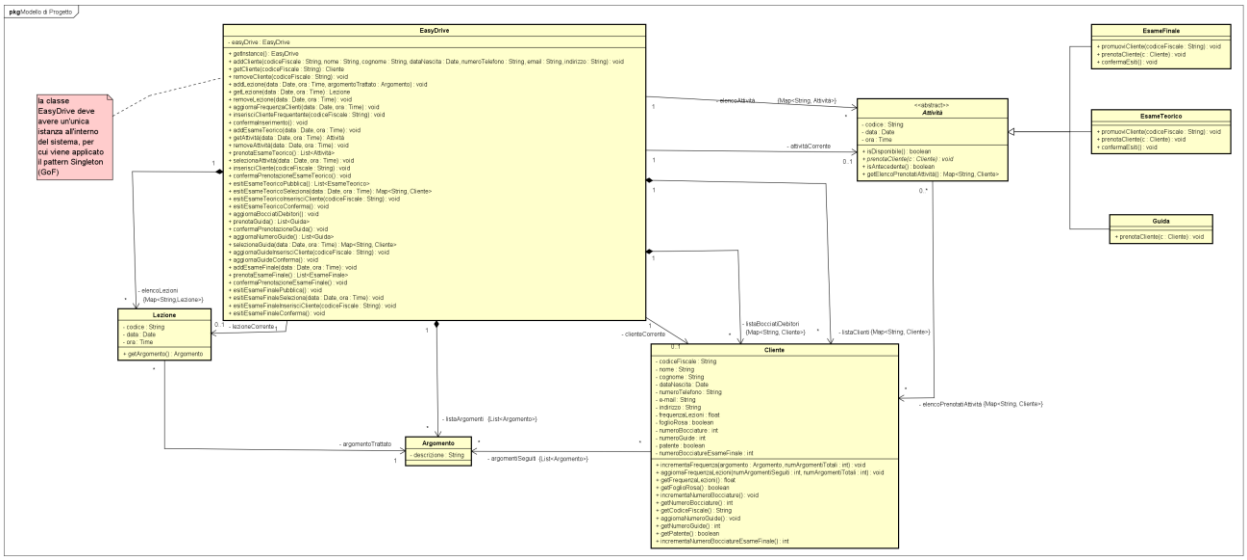
- **Esiti Esame Finale Conferma**





(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)

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)**:
  - **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):**
  - **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.
- **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.