Soluzione dell'esercizio del 12 Febbraio 2004

1. Casi d'uso
I casi d'uso sono riportati in Figura 1.

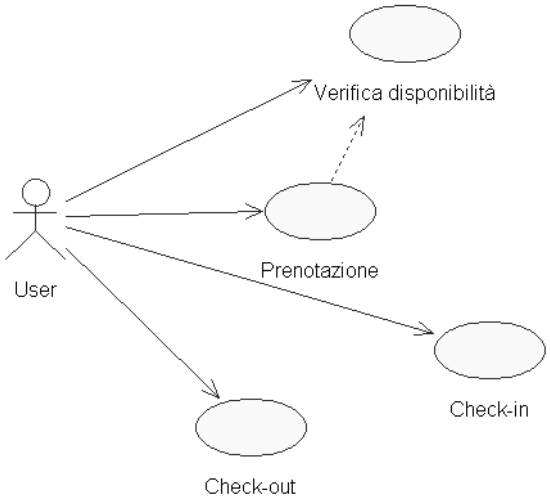

Figura 1: Diagramma dei casi d'uso.

2. Modello concettuale
Osserviamo anzitutto che la Catena è fatta di Hotel e che un Hotel si compone di camere. Ogni camera ha un costo e prevede un certo numero di letti (Figura 2).

L'associazione di prenotazione è espressa attraverso la classe Prenotazione. E' un'associazione molti a molti (una camera può avere più prenotazioni da parti di clienti diversi, in tempi diversi, e un cliente può aver prenotato più prenotazioni di camere in tempi diversi). Gli attributi della classe prenotazione (D_inizio, D_fine) servono a individuare la prenotazione tra camera e cliente ad una certa data.

Inoltre Camera può avere o no un Soggiorno in corso. Il soggiorno ha una data di inizio (quando il cliente ha fatto il check-in) e una data di fine prevista. In questo intervallo la camera non può essere data ad altri né prenotata. La data di fine soggiorno è normalmente dedotta dalla prenotazione, ma è possibile che all'atto del check-in il cliente voglia estendere la permanenza oltre la prenotazione o che voglia soggiornare er un periodo più corto. Il sistema dovrà prendere i provvedimenti del caso.

A un Soggiorno è associato un solo Cliente. Per come è presentato il problema, è naturale associare gli Extra a un (solo) soggiorno. Le relazioni Camera-Soggiorno e Soggiorno-Cliente sono uno a uno.

Un cliente può avere degli accompagnatori. Nel modello non serve rappresentarli con una specifica classe, infatti essi intervengono in maniera anonima. Serve solo il loro numero nel calcolo del conto finale. A tal fine basta prevedere la componente NumeroAccomp nel Soggiorno[1].

Pure gli extra intervengono solo nel calcolo del conto finale (l'esercizio non chiede di saper come si trattano gli extra, chiede solo di metterli in conto alla partenza). Gli extra vengono fruiti durante la permanenza e non è prevedibile il loro numero. Per questo occorre l'associazione con Soggiorno.

Il Delta è una costante valida per tutto il sistema, Ipotizzando che possa in qualche modo essere funzione della camera si può prevedere che esso sia un attributo della classe Camera.

Osservazione: Nel caso specifico (poiché in una camera c'è al più un soggiorno di un cliente), si poteva tenere traccia del soggiorno prevedendo l'associazione "soggiorno" diretta tra camera e cliente. La data di inizio e di fine prevista avrebbero potuto essere componenti di Camera o di Cliente (c'era poi da associare Extra a

[1]Se si volesse modellare il fatto che gli accompagnatori possono variare in numero durante il soggiorno, l'ipotesi di tenerne conto con la componente NumeroAccomp di Soggiorno non è più sufficiente. Occorrerebbe prevedere la classe Accompagnatore associandola in una relazione uno a molti con Soggiorno o con Cliente.

Cliente o a Camera). Per rendere efficiente questa soluzione sarebbe stata utile l'attributo (variabile di stato) "occupata" per la camera. Nel seguito, si presentano i diagrammi di sequenza il riferimento al modello di Figura 2. Chi ha voglia approfondisca la differenza con quanto appena esposto.

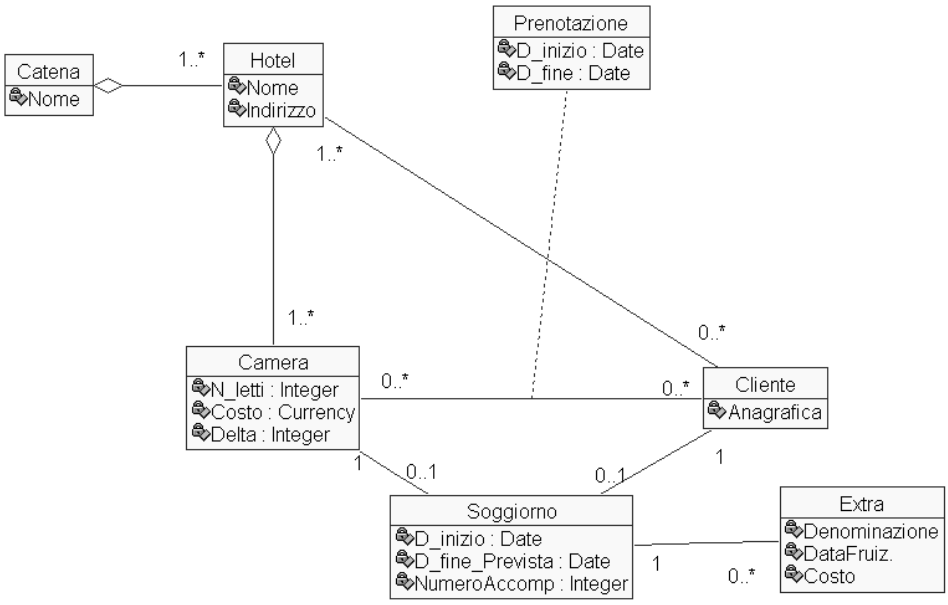

Figura 2: Diagramma UML: associazioni tra le classi

3. Diagrammi di sequenza

a) Esame disponibilità

In Figura 3 viene riportato il diagramma di sequenza corrispondente al caso d'uso della verifica della disponibilità.

Il metodo Get_disp() di Catena viene chiamato a seguito della richiesta dell'utente (potrebbe essere il cliente da postazione remota o il gestore). Occorre prevedere come parametri l'Hotel a cui girare la richiesta, la data di arrivo e partenza e l'eventuale fascia di prezzo. La risposta sarà l'indicazione del numero di camere e il costo minimo e massimo in cui può incorrere l'utente.

Il metodo Get_disp() di Hotel gira la domanda a ogni sua camera. La Camera esamina le Prenotazioni (metodo intervallo()) a essa associate per verificare se nel periodo richiesto è libera. La risposta di Camera a Hotel dà l'ampiezza dell'intervallo di tempo (in cui è libera) entro cui casca la richiesta di prenotazione, al fine di consentire l'ottimizzazione dell'impiego delle camere nell'eventuale scelta.

Bisogna anche tenere conto che all'atto del check-in può essere consentito anticipare la data di fine permanenza rispetto a quella prevista nella prenotazione (ovviamente anche anche consentito di posticipare la data di fine soggiorno, se la camera è libera o se ci sono altre camere libere). Per questo motivo, conviene stabilire di cancellare la prenotazione all'inizio del relativo soggiorno (che invece deve essere istanziato). Ma ciò comporta che la verifica della disponibilità esamini se la prenotazione non va a collidere con i soggiorni in corso.

Notare che Camera restituisce a Hotel il suo costo. L'Hotel provvede a formare la risposta sulla base delle risposte ottenute dalle camere.

Si osservi che User è stato fatto dialogare direttamente con Catena, in quanto stiamo illustrando il dominio applicativo. In fase di implementazione occorrerà prevedere una adeguata interfaccia tra l'utente o l'impiegato della reception dell'albergo e gli oggetti del modello concettuale. Cio comporterà l'introduzione di ulteriori classi.

b) Prenotazione

In Figura 4 viene riportato il diagramma di sequenza corrispondente al caso d'uso della prenotazione. Il metodo prenota() di Catena richiede come parametri: l'Hotel su cui si effettua la prenotazione, il periodo e il tipo di

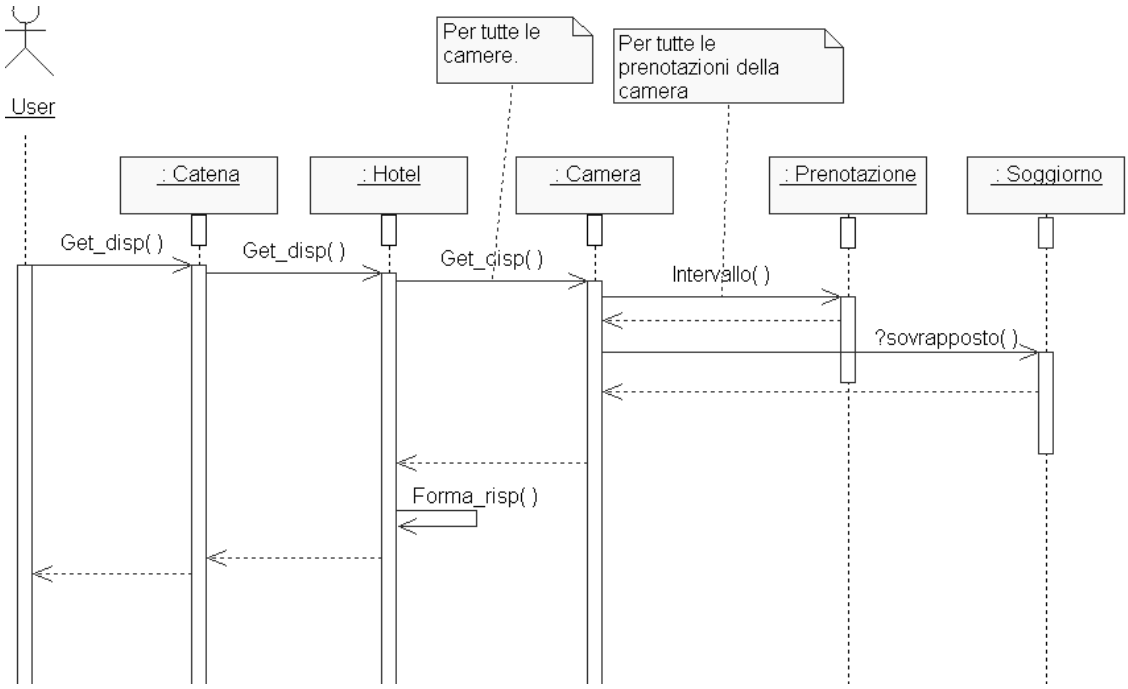

Figura 3: Sequence diagramma della verifica della disponibilità (3a). Occorre esaminare le prenotazioni della camera e verificare che non ci sia un soggiorno in corso. La sequenza presentata si riferisce ad una interrogazione del cliente. Se la verifica viene fatta per il cliente arrivato all'albergo senza prenotazione, viene saltata l'intermediazione dell'oggetto catena.

camera, oltre che i dati del cliente. L'Hotel interroga la camera come nel caso d'uso della verifica (una soluzione più efficiente consiste nel passare anche l'esatto intervallo, e il tipo di camera, in modo da velocizzare le risposte delle camere). In base alla risposta delle camere e alle preferenze del cliente viene effettuata la scelta della camera. (Nella pratica realizzativa, occorrerà prevedere un passo di conferma da parte dell'utente.)

La prenotazione richiede la verifica dell'esistenza del cliente. Per questo nel modello ci deve essere l'associazione Hotel-Cliente.In Figura si è fatto il caso che il cliente esista già. In caso contrario occorre anche istanziare il nuovo Cliente (da parte dell'Hotel).

E' la Camera che costruisce (istanzia) la Prenotazione. Il metodo prenota() di camera deve avere tra gli argomenti il (riferimento al) cliente, per la costruzione dell'associazione.

Si noti la Prenotazione "usa" il caso d'uso Verifica disponibilità.

c) Check-in di un prenotato

In Figura 5 viene riportato il diagramma di sequenza corrispondente al check-in di un prenotato. Il metodo prendi() della camera serve a bloccare la camera prenotata (che comunque fa la verifica col metodo ?conferma() di Prenotazione). Se la conferma è positiva, la prenotazione viene distrutta e viene creato un Soggiorno.

La data di fine del soggiorno è stata dedotta dalla prenotazione, oppure essa può essere data dal Prenotato, in variazione rispetto a quella della prenotazione, sempre che sia compatibile. E' certamente compatibile se viene anticipata. Nel caso il cliente intenda prolungare la permanenza oltre la data di prenotazione, occorre prima esaminare le prenotazioni della camera (non mostrato in figura).

d) Check-in di un non prenotato

Il diagramma di sequenza differisce da quello del caso precedente per il fatto che ora deve essere ripetuta la fase di verifica della disponibilità. Inoltre, non c'è da distruggere nessuna prenotazione perché questa non esiste.

e) Check-out

In Figura 6 è riportato il diagramma di sequenza relativo. Notare che vengono distrutti il Soggiorno e gli eventuali extra associati al soggiorno. Il metodo calcola() di Camera si rende necessario avendo messo Delta in Camera.

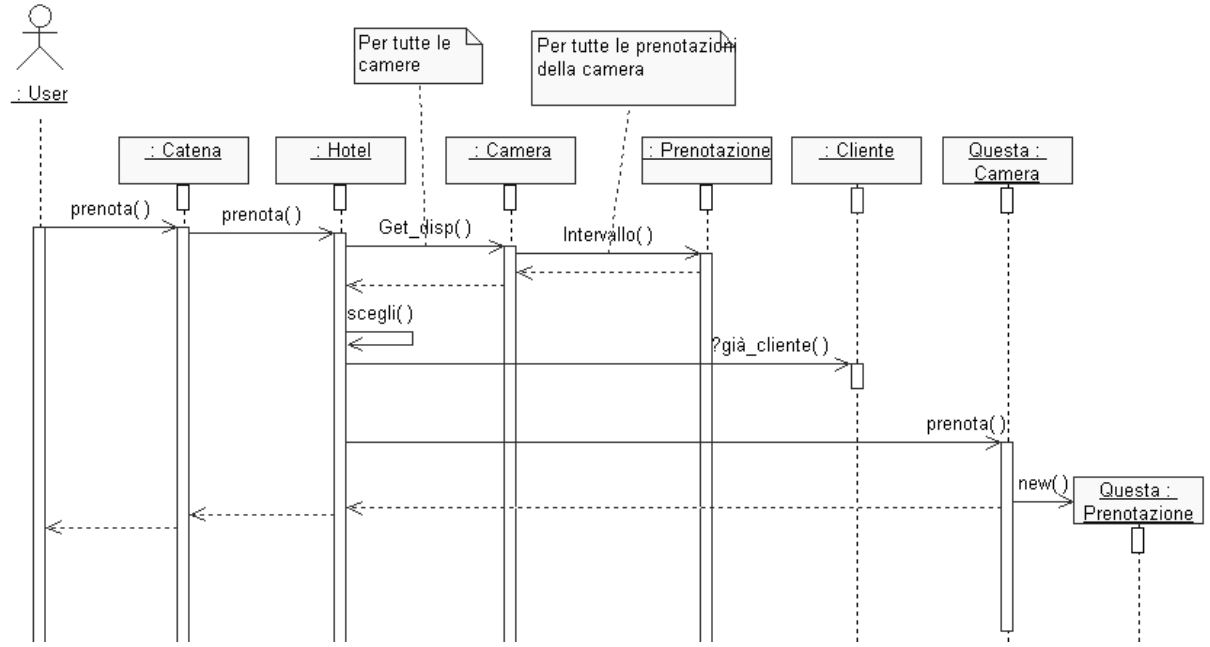

Figura 4: Sequence diagram della prenotazione (3b). Per semplicità si è trascutato di indicare la verifica dell'eventuale conflitto con il soggiorno in corso. La verifica dell'esistenza di un soggiorno in corso per una data camera può essere il passo che precede la verifica dell'intervallo di prenotazione (e che la esclude se il soggiorno si estende sul periodo richiesto). Inoltre, in figura, viene mostrata la sequenza in cui il cliente esiste già nel sistema (metodo già_cliente()). Se il cliente non esiste occorre provvedere alla sua istanziazione.

4. Responsabilità delle classi
Vedere i diagrammi di sequenza e la Figura 7, in cui si mostrano le operazioni di ciascuna classe.

5.
Se deve esser considerato effettivo cliente solo chi ha concluso (almeno) un soggiorno occorrono queste variazioni:

- prevedere un componente (boolenano) di cliente che indichi se esso è o no un cliente stabile. Chiameremo questo campo "Stabile";
- in prenotazione, se il cliente non esiste, viene istanziato il nuovo cliente con Stabile=false, se il cliente esiste Stabile non viene toccato;
- all'atto del check-out il campo Stabile viene comunque portato a true;
- nel caso in cui una prenotazione venga annullata (caso per il quale non è stato richiesto il diagramma di sequenza), il cliente con Stabile==false viene cancellato.

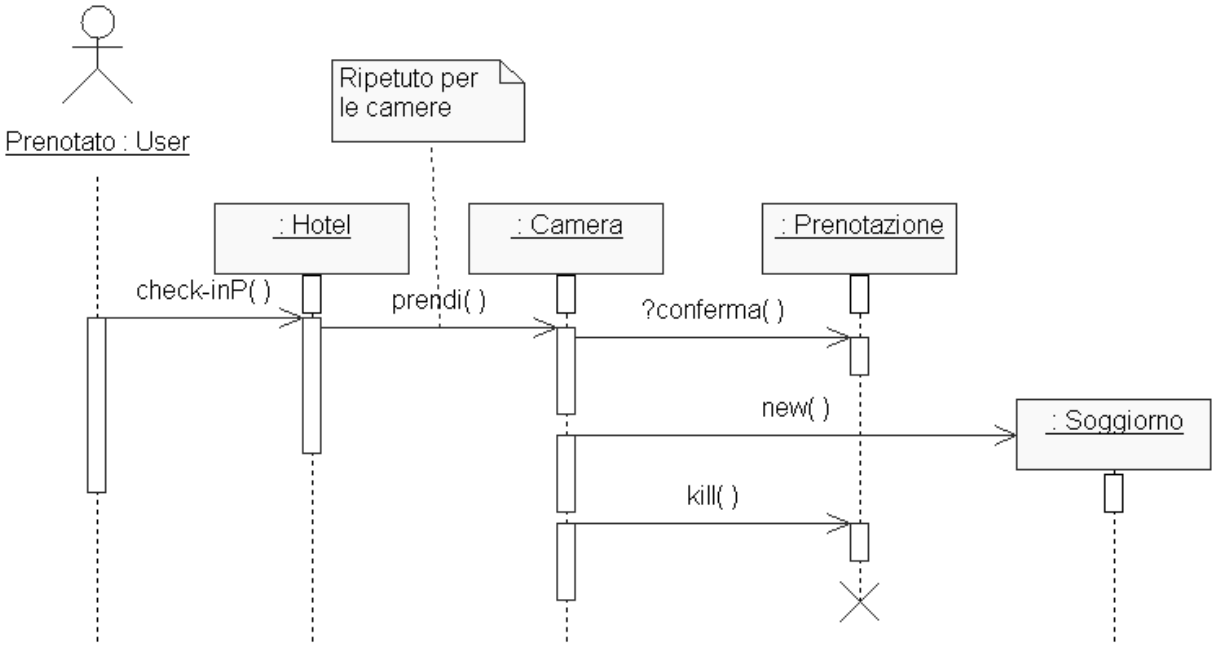

Figura 5: Sequence diagram del check-in di un prenotato (3c).

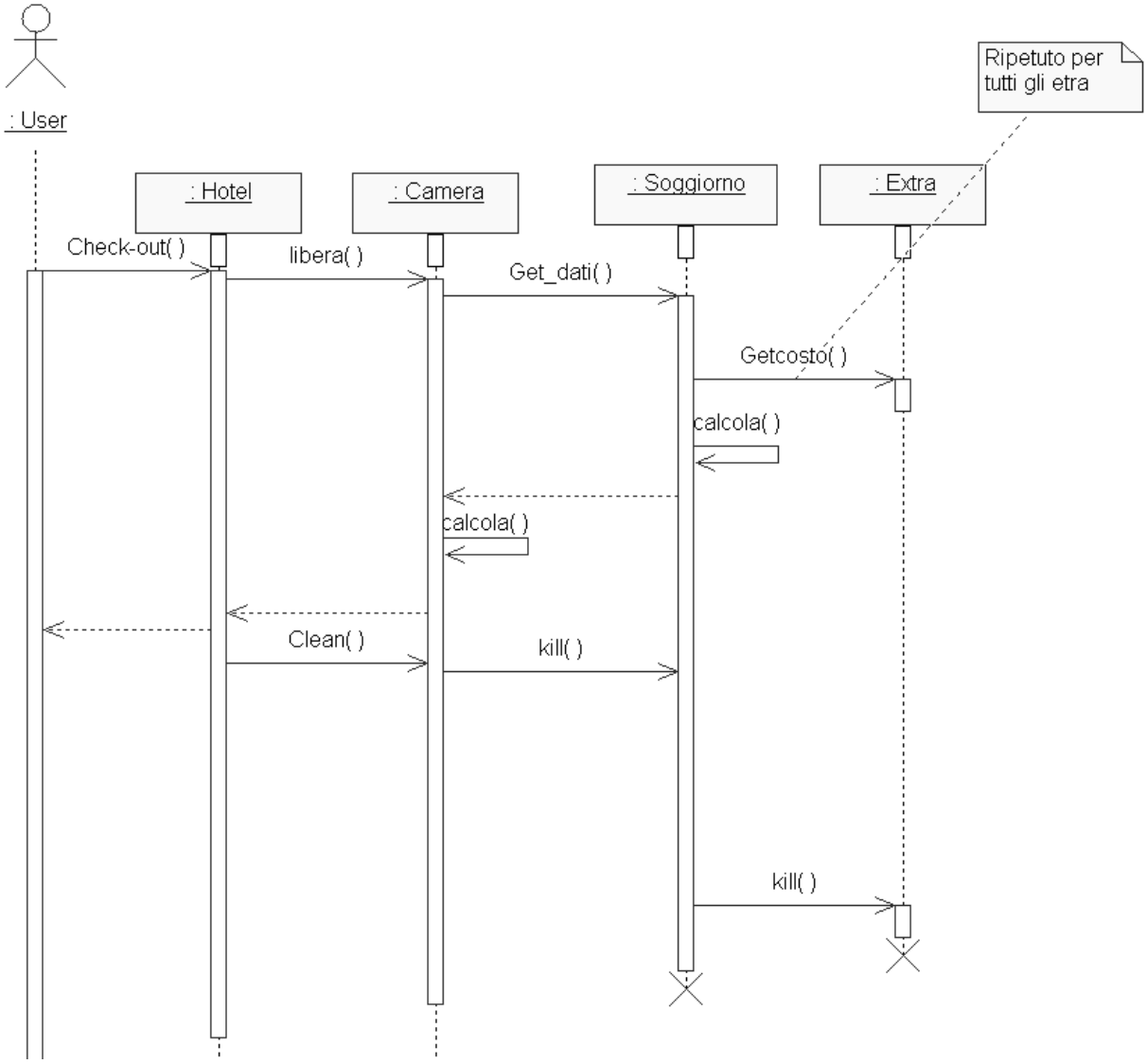

Figura 6: Sequence diagram del check-out (3e).

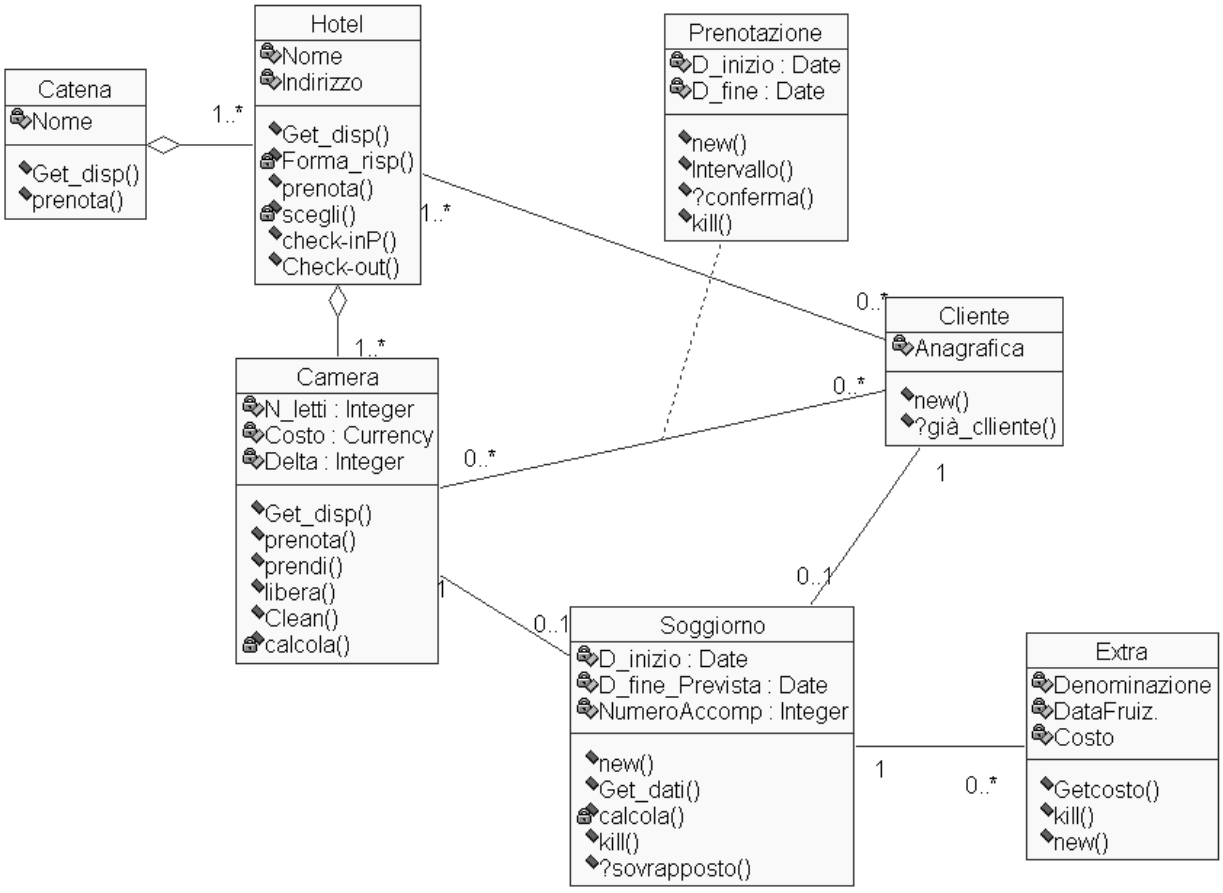

Figura 7: Interfacce delle classi.