04 - Modellazione del Dominio

Modello di Dominio

Il Modello di Dominio è una rappresentazione visuale delle classi concettuali, ovvero oggetti reali del dominio (non sono oggetti software!).

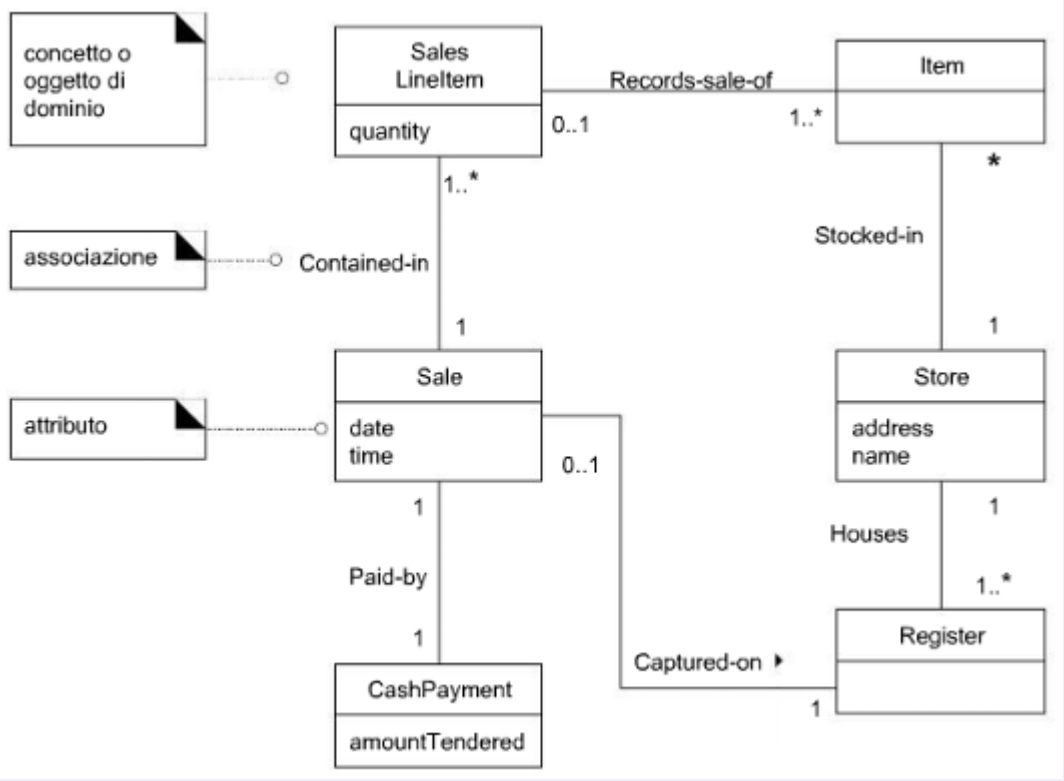
Viene sviluppato nell'ambito della **disciplina di Modellazione del Business**.
È una specializzazione del "**Business Object Model**".

È un insieme di diagrammi di classi UML che includono:

- **Oggetti** di dominio - classi concettuali.
- **Associazioni** tra classi concettuali.
- **Attributi** di classi concettuali.
  Non appaiono operazioni (firme di metodi o responsabilità).

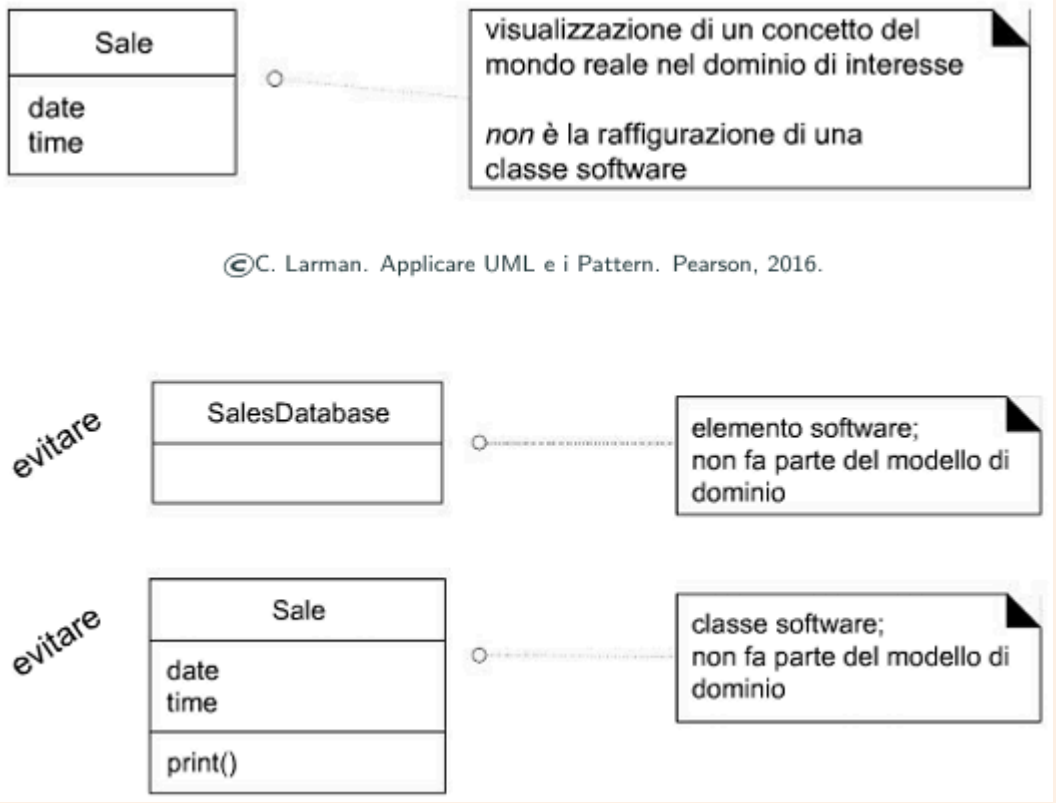
È un dizionario visuale e non è un modello dei dati. È un modo particolare di utilizzare un diagramma delle classi di UML con uno scopo specifico.

Un esempio di Modello di Dominio per POS NextGen:



Classi Concettuali

Una **classe concettuale** rappresenta un **concetto** del mondo reale o del dominio di interesse di un sistema che si sta modellando.

> ⚠ **Classi concettuali, NON classi software!**
>
> 
>

- Il **simbolo** è una parola o un'immagine che rappresenta la classe concettuale.

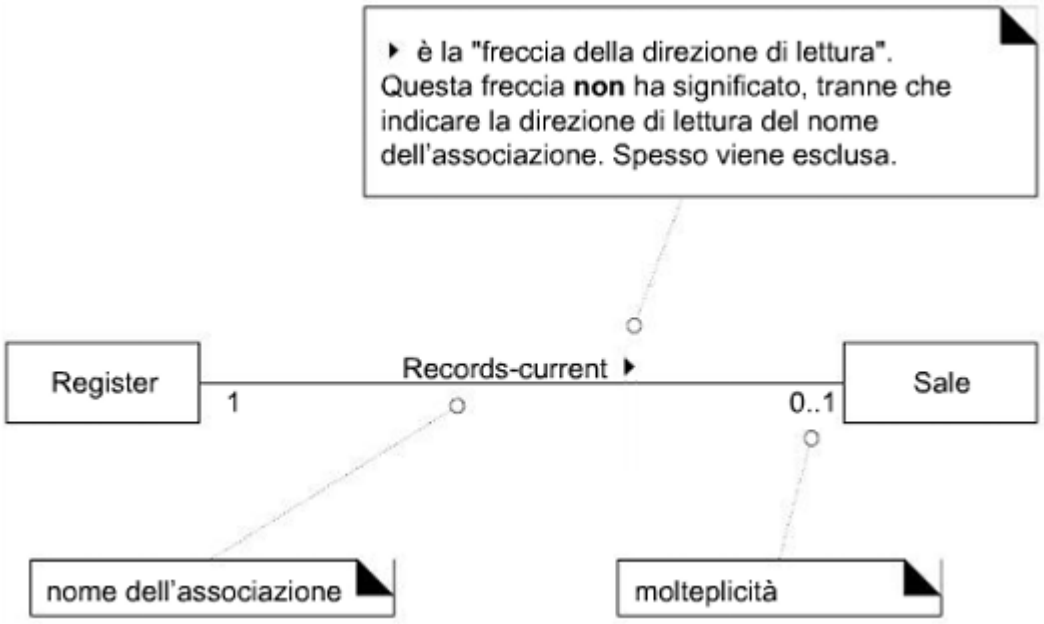
- L'**intensione** è la definizione (in linguaggio naturale) della classe concettuale.

- L'**estensione** è l'insieme degli oggetti della classe concettuale.

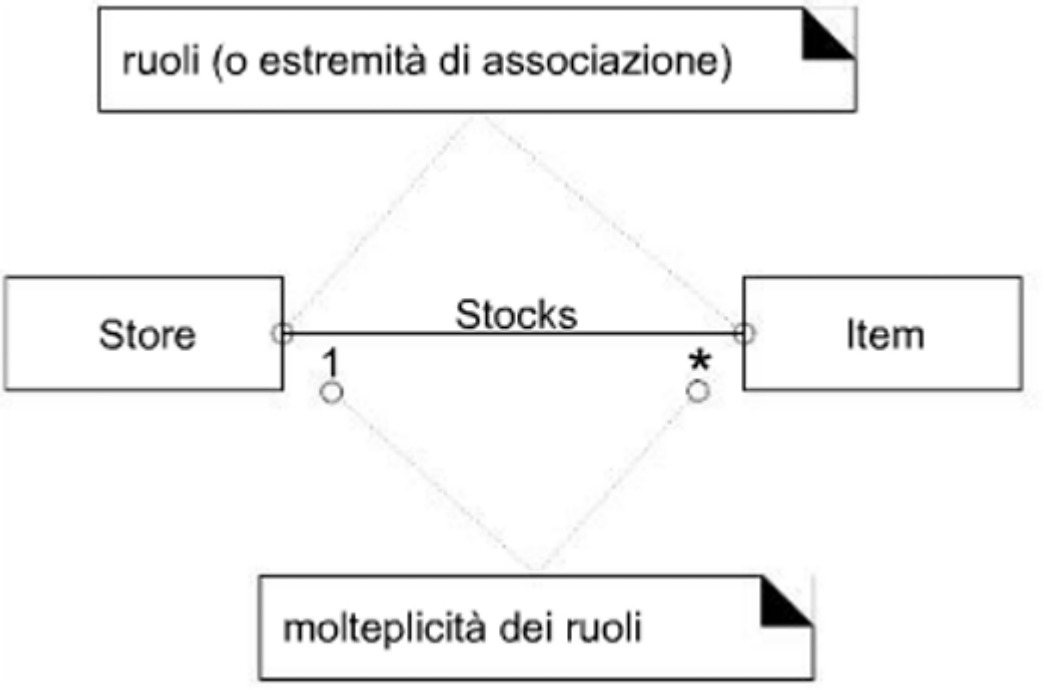
Associazioni

Un'associazione è una **relazione tra classi** (o più precisamente, tra le istanze di queste classi) che indica una **connessione significativa e interessante**.
Se si pensa alle classi in maniera estensionale, un'associazione tra due classi è un insieme

di coppie di oggetti dalle due classi.



Ciascuna estremità di un'associazione è anche chiamata **ruolo**.
I ruoli possono avere, opzionalmente, espressione di **molteplicità**, **nome** e **navigabilità**.



Molteplicità

- La molteplicità di un ruolo definisce quante istanze di una classe possono essere associate a un'istanza di un'altra. I casi possibili sono uno-a-molti, molti-a-uno, molti-a-molti, uno-a-uno.
- La molteplicità comunica quante istanze possono essere associate in modo valido a un'altra istanza **in un particolare momento**, non in un arco di tempo.
- Il valore di una molteplicità dipende dall'interesse del modellatore e dello sviluppatore del software, poiché comunica un vincolo di dominio che si rifletterà nel

software.

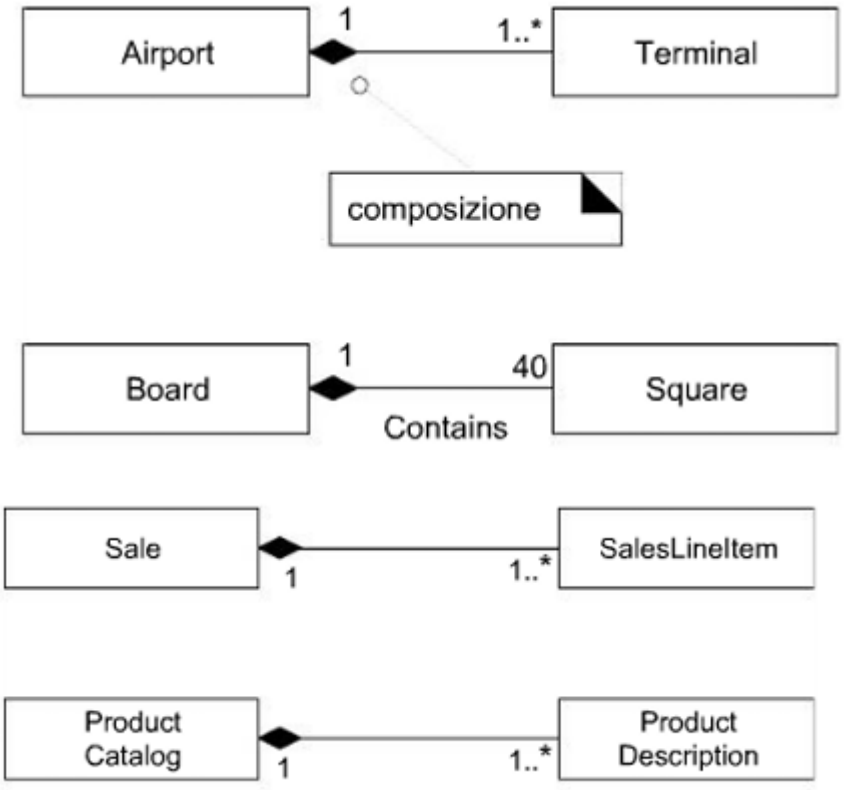
Due classi possono essere collegate da più di una associazione.

Una classe può anche avere un'associazione con se stessa (**associazioni riflessive**).

Composizioni

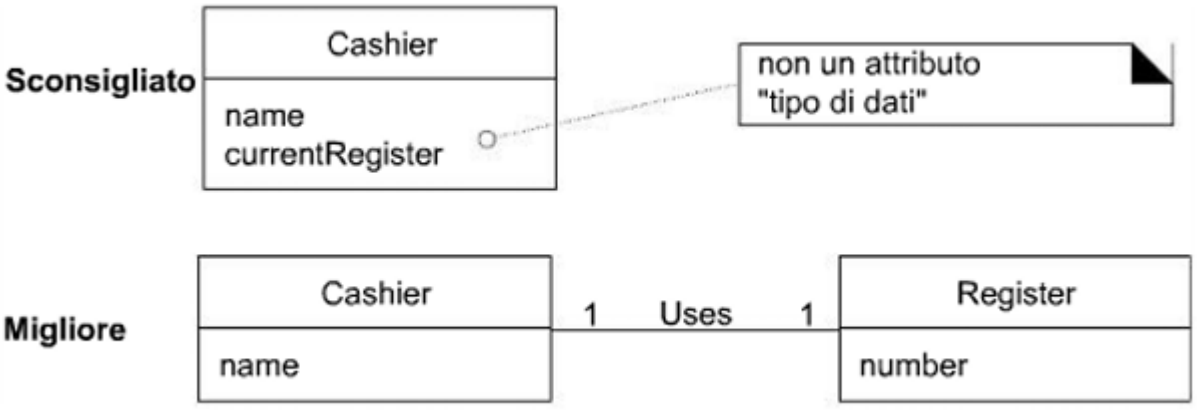
La **composizione**, o aggregazione composta, è un tipo di aggregazione forte **intero-parte**.

- Ciascuna istanza della parte appartiene ad **una sola** istanza del composto alla volta.
- Ciascuna parte deve sempre appartenere **a un** composto
- La vita delle parti è limitata da quella del composto: le parti possono essere create dopo il composto (ma non prima) e possono essere distrutte prima del composto (ma non dopo).



Attributi

Un **attributo** è un valore logico (un dato, una proprietà elementare) degli oggetti di una classe.
Ciascun oggetto della classe ha un proprio valore separato per quella proprietà. È una funzione che associa un valore a ciascun oggetto della classe.

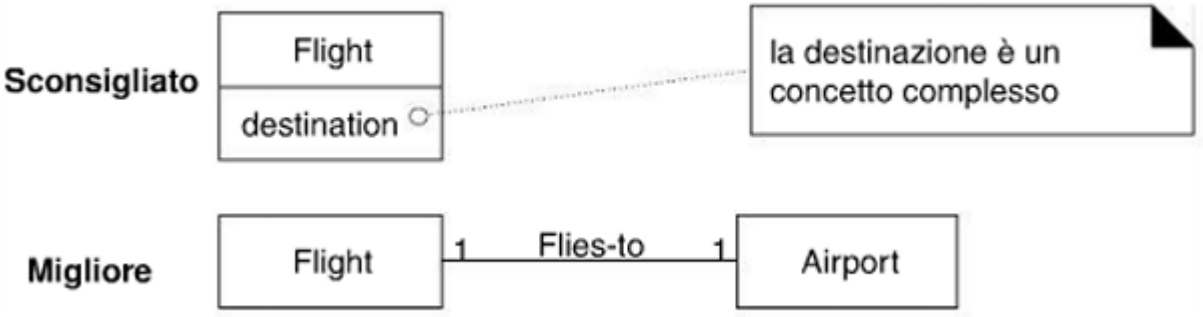


Si aggiungono attributi che sono **tipi di dati** primitivi (boolean, date, number, character, string, ecc.) o **tipi enumerativi**. Un attributo si caratterizza con la sua origine (derivato/non derivato) e il tipo di dato (un vincolo sui valori del dominio).

- Gli attributi in un modello di dominio devono preferibilmente essere di **tipo di dato**.

> **Regola pratica:**
> Se nel mondo reale non si pensa a un concetto X come a un numero, un testo o un valore di un tipo di dato, allora X è probabilmente una classe concettuale, non un attributo.

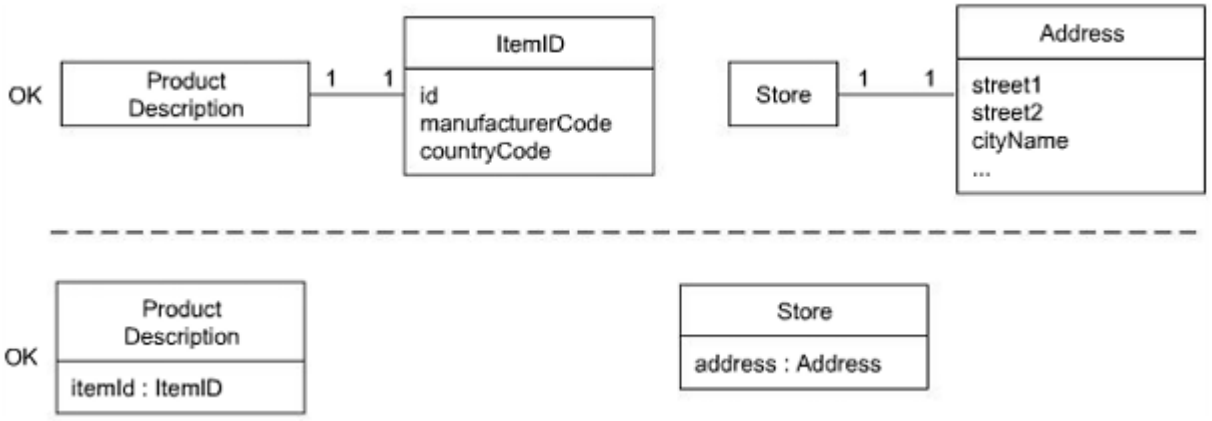
- Le classi concettuali vanno correlate con un'associazione, non con un attributo (no chiavi esterne).



Classi tipo di dato

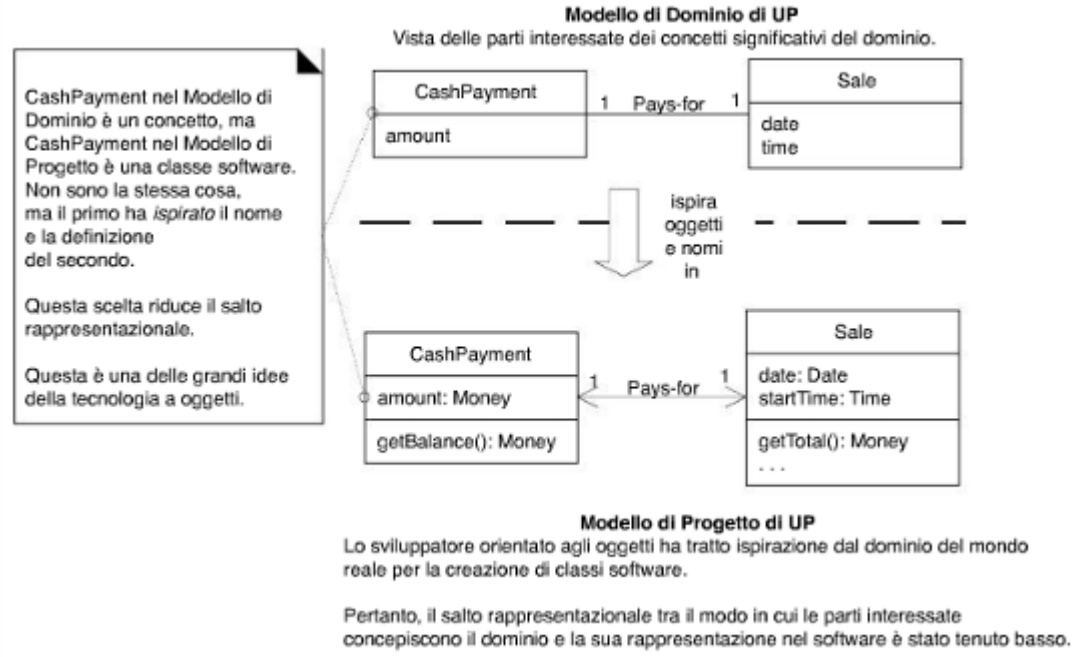
Si introducono quando i dati sono composti da sezioni separate (es. nome, numero di telefono), quando ci sono operazioni associate ai dati (es. validazione codice fiscale), o per

quantità con unità di misura (es. totale da pagare con valuta).



Riduzione del "gap di rappresentazione"

Comprendere i concetti chiave e la terminologia del dominio del problema è cruciale per ridurre il "gap di rappresentazione" tra il mondo reale e il software.
Questo include **comprendere** il dominio e il suo vocabolario, definire un **linguaggio comune** per la comunicazione tra le parti interessate, e servire come **fonte di ispirazione** per la progettazione dello strato del dominio.



Creare un Modello di Dominio

Ci si restringe ai requisiti scelti per la progettazione nell’iterazione corrente. I passi sono:

1. Trovare le classi concettuali:
   Attraverso riuso/modifica di modelli esistenti, liste di categorie (es. per il dominio POS, enfatizzando transazioni commerciali), e analisi linguistica delle descrizioni testuali (es. casi d'uso dettagliati).
   Il mapping nome-classe non è automatico a causa dell'ambiguità del linguaggio naturale.

Esempio: elenco di categorie

Categoria di classe concettuale	Esempi
transazioni commerciali *Linea guida*: sono aspetti critici (riguardano denaro), dunque si inizi con le transazioni.	*Sale, Payment (o CashPayment)* *Reservation*
elementi/righe di transazioni *Linea guida*: le transazioni spesso sono composte da righe per gli articoli correlati, quindi queste vanno considerate subito dopo le transazioni.	*SalesLineItem*
prodotto o servizio correlato a una transazione o a una riga di transazione per articolo *Linea guida*: le transazioni sono *per* qualcosa (un prodotto o un servizio). Vanno considerate subito dopo.	*Item* *Flight, Seat, Meal*
dove viene registrata la transazione? *Linea guida*: importante.	*Register, Ledger (o SalesLedger), FlightManifest*
ruoli di persone o organizzazioni correlati alle transazioni; attori nei casi d'uso *Linea guida*: normalmente dobbiamo sapere quali sono le parti coinvolte in una transazione.	*Cashier, Customer, Store* *MonopolyPlayer* *Passenger, Airline*
luogo della transazione; luogo del servizio	*Store* *Airport, Plane, Seat*
eventi significativi, spesso con un'ora o un luogo che è necessario ricordare	*Sale, Payment (o CashPayment)* *MonopolyGame* *Flight*
oggetti fisici *Linea guida*: questo è particolarmente importante quando si crea software per il controllo di dispositivi, oppure simulazioni.	*Item, Register* *Board, Piece, Die* *Airplane*
descrizioni di oggetti *Linea guida*: vedere il Paragrafo 12.13 per una discussione.	*ProductDescription* *FlightDescription*
cataloghi *Linea guida*: le descrizioni sono spesso contenute in un catalogo.	*ProductCatalog* *FlightCatalog*
contenitori di oggetti (fisici o informazioni)	*Store, Bin* *Board* *Airplane*
oggetti in un contenitore	*Item* *Square (in un Board)* *Passenger*
altri sistemi che collaborano	*CreditAuthorizationSystem* *AirTrafficControl*
registrazioni di questioni finanziarie, di lavoro, contrattuali e legali	*Receipt, Ledger* *MaintenanceLog*
strumenti finanziari	*Cash, Check, LineOfCredit* *TicketCredit*
piani, manuali, documenti cui si fa regolarmente riferimento per eseguire il lavoro	*DailyPriceChangeList* *RepairSchedule*

Esempio: analisi linguistica

Scenario principale di successo (o Flusso di base):

1. Il **Cliente** arriva alla **cassa POS** con gli **articoli** e/o i **servizi** da acquistare.
2. Il **Cassiere** inizia una nuova **vendita**.
3. Il **Cassiere** inserisce il **codice identificativo di un articolo**.
4. Il Sistema registra la **riga di vendita per l'articolo** e mostra la **descrizione dell'articolo**, il suo **prezzo**, il **totale** parziale. Il prezzo è calcolato in base a un insieme di regole di prezzo.

Il Cassiere ripete i passi 2-3 fino a che non indica che ha terminato.

5. Il Sistema mostra il totale con le **imposte** calcolate.
6. Il Cassiere riferisce il totale al Cliente e richiede il **pagamento**.
7. Il Cliente paga e il Sistema gestisce il pagamento.
8. Il Sistema registra la vendita completata e invia informazioni sulla **vendita** e sul pagamento ai sistemi esterni di **Contabilità** (per la contabilità e le **commissioni**) e di **Inventario** (per l'aggiornamento dell'inventario).
9. Il Sistema genera la **ricevuta**.
10. Il Cliente va via con la ricevuta e gli articoli acquistati.

Estensioni (o Flussi alternativi):

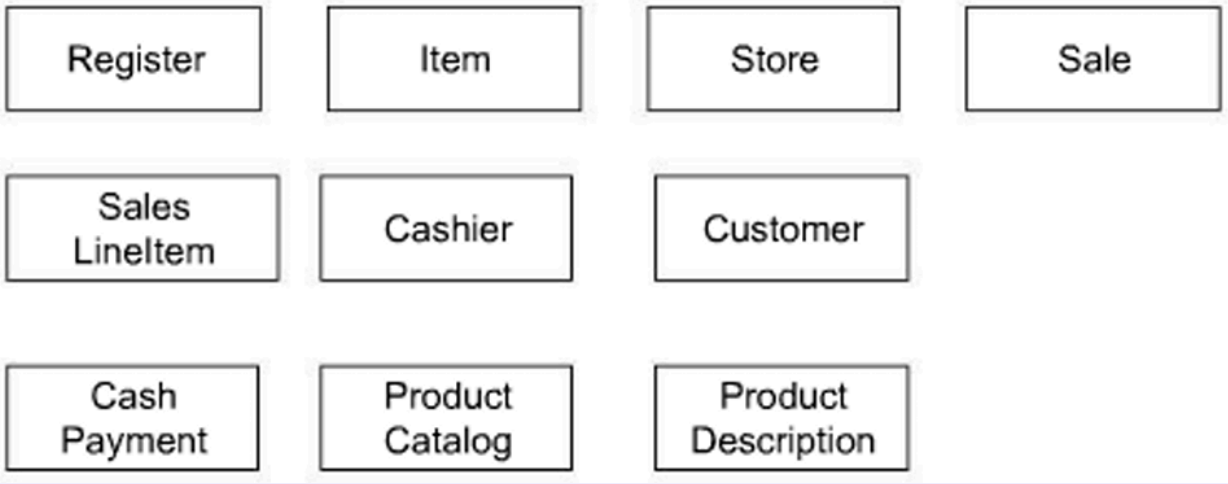
. . .

7a. Pagamento in contanti:

1. Il Cassiere inserisce l'**importo in contanti** presentato dal Cliente.
2. Il Sistema mostra il **resto dovuto** e apre il cassetto della **cassa**.
3. Il Cassiere deposita il contante presentato e restituisce il resto in contanti al Cliente.
4. Il Sistema registra il **pagamento in contanti**.

Esempio: definizione delle classi

Un modello di dominio conterrà una collezione, in un certo senso arbitraria, di astrazioni e termini del dominio.



Si consideri il concetto "*Ricevuta*":

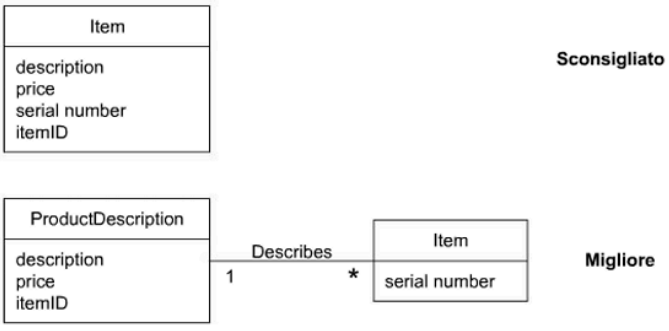
- Se le informazioni che contiene si possono derivare da altre fonti (altre classi concettuali) si può escludere dal modello di dominio
- Se ha un ruolo in termini di regole di business, business ad esempio conferisce il diritto al possessore di restituire oggetti acquistati acquistati, allora è bene includere il concetto nel modello di dominio

Nota: nell'iterazione corrente di POS la restituzione articoli non è considerata, quindi *Ricevuta* viene esclusa dal modello di dominio in questa iterazione

Classi "Descrizione"

Contengono informazioni che descrivono qualcos'altro e sono utili per ridurre informazioni ridondanti. È utile avere un'associazione che collega la classe e la sua classe descrizione (*pattern Item-Descriptor*)

- È necessaria una descrizione di un articolo o servizio indipendentemente dall'attuale esistenza di istanze dell'articolo o servizio
- L'eliminazione delle istanze di un articolo (esempio, Item) darebbe luogo ad una perdita di informazioni che è necessario conservare
- Si vogliono ridurre le informazioni ridondanti



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

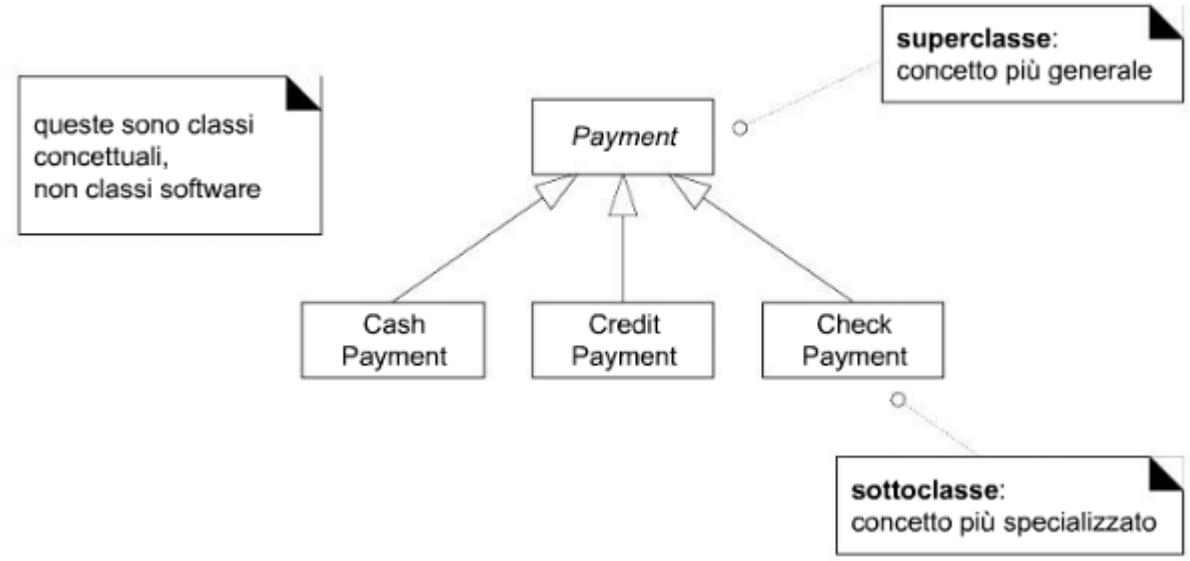
Ultime verifiche al modello:

- Verificare le classi concettuali introdotte (alternative classe-attributo, classi descrizione).
- Verificare le associazioni (indipendenza delle diverse associazioni tra le stesse classi).
- Verificare gli attributi (non introdurre attributi per riferirsi ad altre classi concettuali, usare le associazioni).

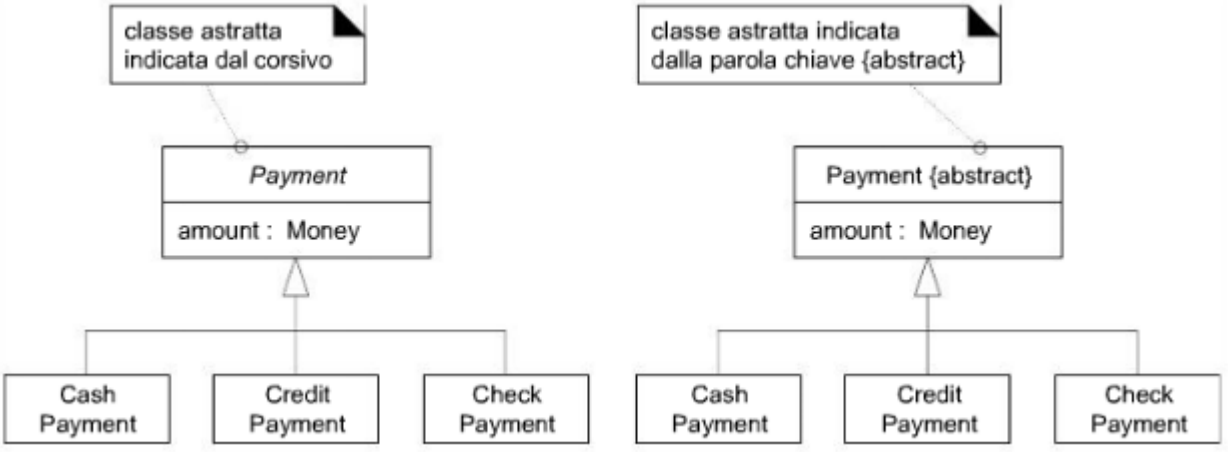
Il modello di dominio in UP viene creato principalmente durante le iterazioni dell'elaborazione, quando c'è massima necessità di **capire i concetti significativi** e rappresentarli come classi software.

Generalizzazione

La **generalizzazione** è un'astrazione basata sull'identificazione di *caratteristiche comuni tra concetti*, che porta a definire una relazione tra un concetto più generale (*superclasse*) e uno più specializzato (*sottoclasse*). Vale il principio di **sostituibilità**.



Una classe concettuale è **astratta** se ogni suo elemento è anche elemento di una delle sue sottoclassi.



Classi di Associazioni

Permettono di aggiungere attributi e altre caratteristiche alle associazioni.