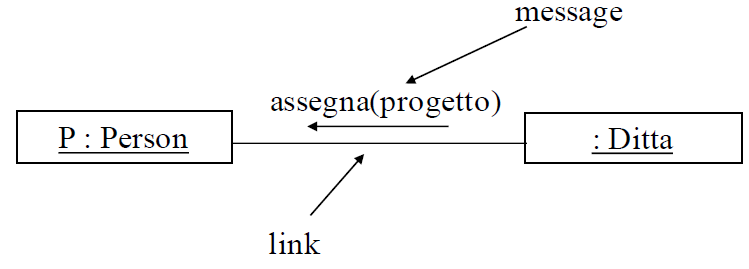
4.3 **Sequence Diagrams (Interaction Diagram)**

Descrivono le sequenze di azioni e le interazioni tra le componenti, servono a dettagliare gli ***use case***, definire le interfacce del sottosistema e per trovare tutti gli oggetti partecipanti al sistema, evidenziando la sequenza temporale delle azioni. Sono utili per evidenziare la distribuzione del controllo nel sistema (“chi” fa “che cosa” …).

Una ***interazione*** è un comportamento che comprende un insieme di messaggi scambiati tra un insieme di oggetti per ottenere un risultato, avviene tra oggetti tra cui esiste un link (istanza di un’associazione).

Gli oggetti collaborano scambiandosi ***messaggi*** che specificano la comunicazione tra oggetti che trasmettono informazioni.

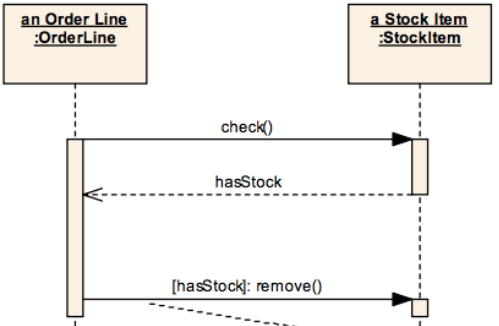
Per l’invio di un messaggio è necessario: ricevente, messaggio e eventuali informazioni aggiuntive.

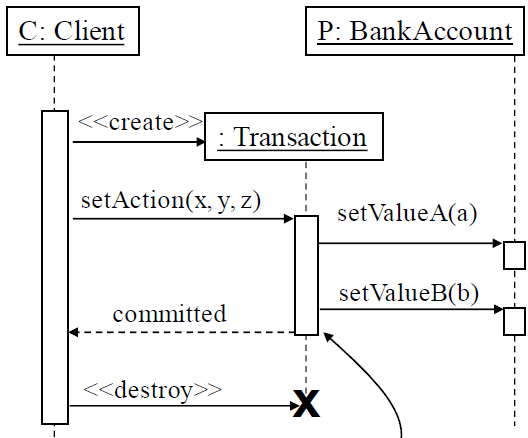
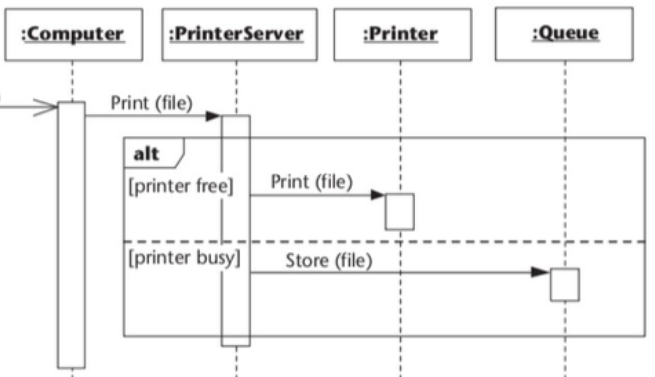
4.3.1 **Componenti Sequence Diagrams**

* ***Linee verticali***: rappresentano le attività svolte dagli oggetti;
* ***Linee orizzontali***: rappresentano sequenze dei messaggi scambiati tra oggetti;
* ***Sequence Diagrams***: possono corrispondere ad uno scenario o ad un intero caso d’uso (si possono annotare vincoli temporali);
* ***Oggetti***: istanza di una classe, sono disposti orizzontalmente. Sintassi: ***nomeOggetto : NomeClasse;***



* ***Life-time di un oggetto***: linea tratteggiata verticalmente all’oggetto.
* ***Flusso del Tempo***: è descritto verticalmente;
* ***Life-time di un metodo***: rappresentato da un rettangolo che collega la freccia di invocazione con la freccia di ritorno, rappresentata con una freccia tratteggiata (ritorno opzionale, se si omette, la fine è decretata dalla fine del life-time).



* ***Messaggi sincroni***: si disegna con una *freccia chiusa*, etichettata col nome del metodo. Il chiamante attende la terminazione del metodo prima di proseguire.
* ***Messaggi asincroni***: usati per descrivere messaggi concorrenti, si disegna con una *freccia aperta*, etichettata col nome del metodo. Il chiamante non attende la terminazione, ma prosegue subito dopo l’invocazione.
* ***Condizione di un metodo***: invocato solo se la condizione è verificata, posizionata sulla freccia del metodo. Sintassi: ***[cond]:nomeMetodo()***.
* ***Messaggi iterativi***: sintassi: ***\*messaggio()***
* ***Messaggi condizionati***: sintassi ***[cond]messaggio()***
* ***Costruzione nuovo oggetto***: corrisponde all’allocazione dinamica. Etichettata con ***new*** o ***create***.
* ***Distruzione oggetto***: corrisponde alla deallocazione dinamica. Etichettata con ***X*** posta alla fine dell’attività dell’oggetto.
* ***Box di attività***: indica il periodo di tempo durante il quale un oggetto sta eseguendo una action.
* ***Iterazioni (loop)***: esecuzione ciclica di più messaggi, si disegna raggruppando con un blocco i messaggi su cui si vuole iterare. Si può aggiungere la condizione di iterazione, rappresentata tra parentesi quadre.
* ***If-then-else (Alt)***: la condizione si indica in cima, se ci sono else si usa una linea tratteggiata per separare la zona then.
* ***If-then (Opt)***: sequenza che viene eseguita solo se la condizione è verificata.
* ***Auto-chiamata***: oggetto che invoca un proprio metodo. Rappresentato con una freccia circolare.

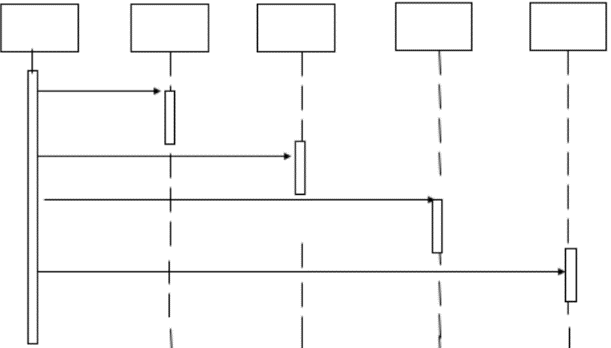
4.3.2 **Euristiche Sequence Diagrams**

Le colonne rappresentano gli oggetti che partecipano al caso d’uso.

* ***1 colonna***: rappresenta l’attore che inizia lo *use case*;
* ***2 colonna***: rappresenta l’oggetto *boundary* con cui l’attore interagisce per rappresentare lo *use case*;
* ***3 colonna***: rappresenta l’oggetto *control* che gestisce il resto dello *use case*;
* ***Dalla 4 colonna***: vengono rappresentati gli oggetti *entity*.

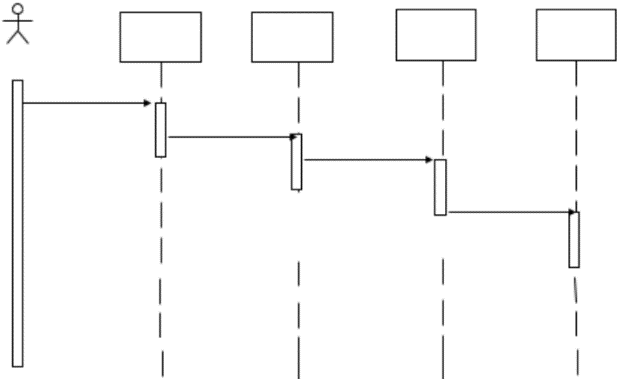
Gli oggetti control creano oggetti boundary e possono interagire con altri control. Gli oggetti *entity* non devono mai chiamare oggetti *control* o *boundary*.

Al top del diagramma si trovano oggetti esistenti dall’inizio.

4.3.3 **Fork Diagram**

Gran parte del comportamento dinamico è inserito in un singolo oggetto, generalmente l'oggetto di controllo.

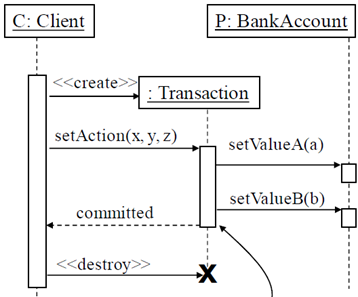
Conosce tutti gli altri oggetti e li usa spesso per domande e per comandi diretti.

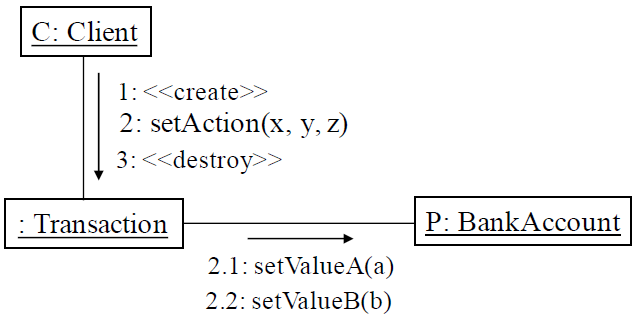
4.3.4 **Stair Diagram**

Il comportamento dinamico è distribuito. Ogni oggetto delega alcune responsabilità ad altri oggetti.

Ogni oggetto conosce solo alcuni degli altri oggetti e sa quali oggetti possono aiutare con un comportamento specifico.

4.3.5 **Collaboration Diagram**

La sequenza dei messaggi è meno evidente che nel diagramma di sequenza, mentre sono più evidenti i legami tra gli oggetti. Adatti per concorrenza e thread, invocazioni innestate.



🡪