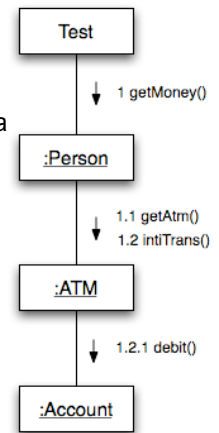


# Pattern AOP Wormhole

- Il pattern wormhole rende informazioni del contesto del chiamante disponibili al chiamato, senza dover passare tali informazioni ad ogni metodo del flusso di controllo
- Problema in assenza di AspectJ
  - Passaggio di parametri aggiuntivi che tengono il contesto del chiamante attraverso vari moduli intermedi
    - Svantaggio: tutti i moduli hanno questi parametri aggiuntivi
  - Oppure
  - Creazione di una variabile che memorizza le informazioni del contesto
    - Svantaggio: modificare sia chiamante che chiamato a tale scopo e capire come il contesto è memorizzato

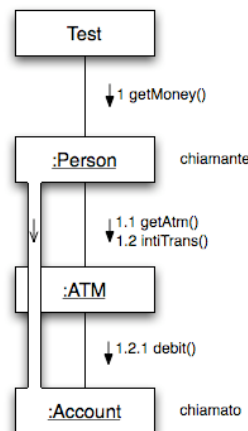
# Esempio

- Il chiamante è un oggetto Person
- Il chiamato è un oggetto Account
  - Tramite un oggetto ATM
- Vorrei poter conoscere al momento di iniziare debit() sia l'istanza di Person che l'istanza di Account
  - Non ho entrambi i dati (insieme) in nessuna delle classi
    - Person conosce la sua istanza corrente e l'istanza di ATM
    - ATM conosce la sua istanza corrente e l'istanza di Account
    - Account conosce solo la sua istanza corrente
  - Usando un aspetto che cattura this() e target(), come visto qualche lezione prima, avrei
    - L'istanza di Person e l'istanza di ATM, oppure
    - L'istanza di ATM e quella di Account



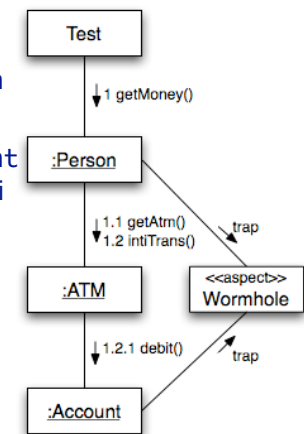
# AOP Wormhole

- Con AspectJ
  - L'idea è di specificare due pointcut
    - Uno per il chiamante ed
    - Uno per il chiamato
  - Il primo pointcut colleziona il contesto che verrà trasferito (tramite wormhole)
  - Il secondo pointcut specifica i join point interessati al contesto e che sono nel flusso del chiamante



# AOP Wormhole

- L'aspetto Wormhole cattura
  - Il contesto (l'istanza di Person) quando vengono avviate alcune operazioni su Person
  - Il contesto (l'istanza di Account) quando vengono avviate alcune operazioni su Account che appartengono al flusso delle operazioni di Person
- Conseguenze dell'uso dell'aspetto Wormhole
  - Wormhole conosce entrambe le istanze precedentemente raccolte
  - Le classi Person e ATM non sono costrette a passare il proprio riferimento ad Account



## Template dell'aspetto wormhole

```
public aspect WormholeAspect {
    pointcut callerSpace(<caller context>) : <caller pointcut>;

    pointcut calleeSpace(<callee context>) : <callee pointcut>;

    pointcut wormhole(<caller context>, <callee context>) :
        cflow(callerSpace(<caller context>)) &&
        calleeSpace(<callee context>);

    void around(<caller context>, <callee context>) :
        wormhole(<caller context>, <callee context>) {
        ... advice body
    }
}
```

Ing. E. Tramontana - AOP Wormhole - 13-Dic-06 5

## Aspetto wormhole

- L'aspetto crea un wormhole tra l'oggetto che inizia una transazione (Person) e le operazioni sull'Account
  - Un primo pointcut cattura l'esecuzione di operazioni di un oggetto Person e colleziona il riferimento a tale oggetto
  - Un secondo pointcut cattura l'esecuzione di metodi credit() e debit() della classe Account e colleziona l'istanza di Account e l'ammontare come contesto
  - Il pointcut wormhole crea il passaggio tra le operazioni di Person e quelle di Account, catturando i join point del secondo pointcut nel flusso del primo pointcut
  - L'advice che esegue grazie al pointcut wormhole può usare il contesto
    - Conosce sia l'account e l'ammontare, sia chi causa l'attività su account

Ing. E. Tramontana - AOP Wormhole - 13-Dic-06 6

### • Come detto:

- cflow(pointcut) cattura tutti i join point dal momento in cui un certo pointcut specificato è catturato

Ing. E. Tramontana - AOP Wormhole - 13-Dic-06 7

## Aspetto Wormhole

```
public aspect Wormhole {
    pointcut callerSpace(Person p) : execution(* Person.*(..)) && this(p);

    pointcut calleeSpace(Account acc, int amount) :
        this(acc) && args(amount) &&
        execution(* Account.*(int));

    pointcut wormhole(Person p, Account acc, int amount) :
        cflow(callerSpace(p)) && calleeSpace(acc, amount);

    void around(Person p, Account acc, int amount) : wormhole(p, acc, amount) {
        Store.write(" **** "+p.getName()+" preleva "+amount+" ****");
        if ((p.getHappiness() < 3) && (acc.getBalance() > 10*amount))
            proceed(p, acc, amount);
    }
}
```

Ing. E. Tramontana - AOP Wormhole - 13-Dic-06 8