Programmazione a Oggetti
Modulo B

Lezione 20

Dott. Alessandro Roncato

15/04/2013

Varie

La lezione di Lunedì 29 Aprile la spostiamo a Martedì 30 Aprile

Progetto e/o compitini

Riassunto

- Type Object
- Manager
- Indirection
- Adapter

Oggi

- Facade (cenni)
- Strategy
- Composite
- Introduzione Parser (Protect Variations, Type Object)

Facade

Come separare un sottosistema composto da un insieme disparato di implementazioni o interfacce dal resto dell'applicazione?

Definisci un punto di contatto singolo (l'oggetto facade) che nasconde il sottosistema.

Facade

Fornisce Protect Variations usando Indirection.

Molto simile ad Adapter ma lo scopo è diverso:

- Adapter per client e server esistenti;
- Facade è preventivo e/o per sperimentare implementazione diverse;

Favorisce Low Coupling all'esterno del sottosistema, ma non I.E. E L.C. nel sottosistema stesso!

Di solito si implementa con un Singleton

Esempio Facade

Per la gestione dei prodotti non è ancora stato deciso se usare Polimorfismo o Type Obejct. Come facciamo a sviluppare il resto dell'applicazione senza dover dipendere da questa scelta?

Definisco un oggetto Facade che nasconde l'interazione con la parte ancora da definire.

Nota che li oggetti nascosti sono più di uno e con molteplicità diversa a seconda della scelta.

Strategy

A chi assegnare la responsabilita' di gestire un insieme di algoritmi, regole o politiche variabili ma correlati e in modo che sia possibile modificare la scelta di tali algoritmi?

Definisci una nuova interfaccia di pura invenzione e implementa gli algoritmi diversi in diverse classi usando il polimorfismo.

Strategy

Esempio: il calcolo della punteggio per assegnare le borse di studio potrebbe essere risolto con questo pattern. Nell'esempio già fatto a lezione in caso avessimo scelto Strategy invece di polimorfismo, avremmo dovuto inventare una classe (es. CalcoloPunteggioStrategy) con l'unico metodo per calcolare il punteggio.

Strategy

Esempio: il calcolo del totale dello scontrino nei supermercati:

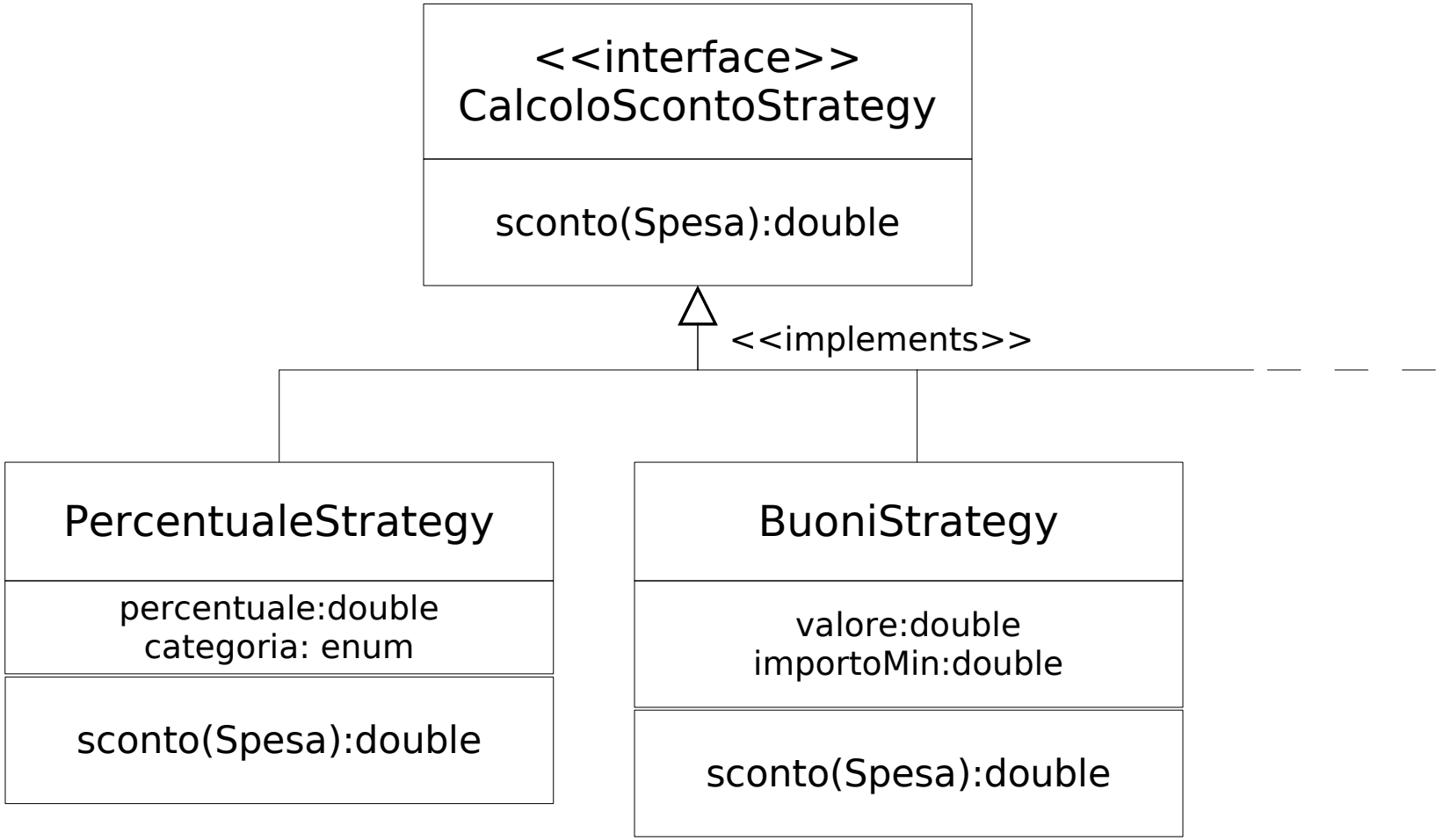
- Il martedì lo sconto del 5%:
- venerdì uno sconto a 5€ per buono presentato ogni 30€ di spesa (buoni trattati come prodotti con prezzo a zero)
- Il lunedì uno sconto di 10€ per spese > 100€
- Il sabato niente sconto ma dalle 18:00 tutti i prodotti deperibili scontati del 33%;

Strategy

Quando viene calcolato il totale si richiama quindi una delle politiche di sconto sopra elencate. Tale politica viene usata per il calcolo dello sconto e quindi del totale.

Per I.E. Il totale viene calcolato sulla Spesa che a sua volta per applicare lo sconto chiamerà il metodo opportuno di strategy (sconto per motivi di spazio, ma sarebbe meglio calcolaSconto).

Strategy


<<interface>>
CalcoloScontoStrategy
sconto(Spesa):double
<<implements>>
PercentualeStrategy
percentuale:double
categoria: enum
sconto(Spesa):double
BuoniStrategy
valore:double
importoMin:double
sconto(Spesa):double

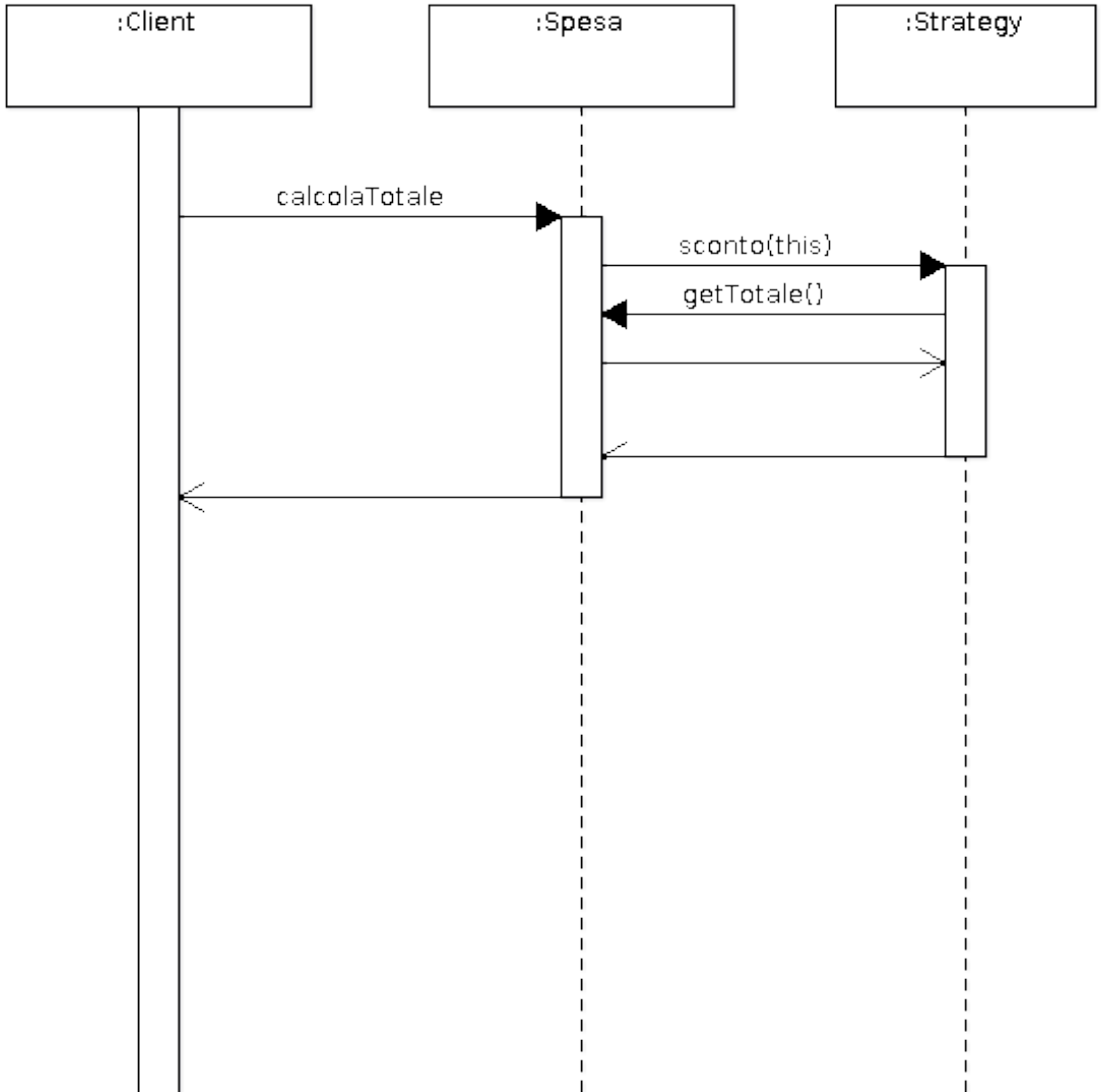
Strategy

Vediamo che l'oggetto strategia ha bisogno di un riferimento all'oggetto Spesa (Contesto).

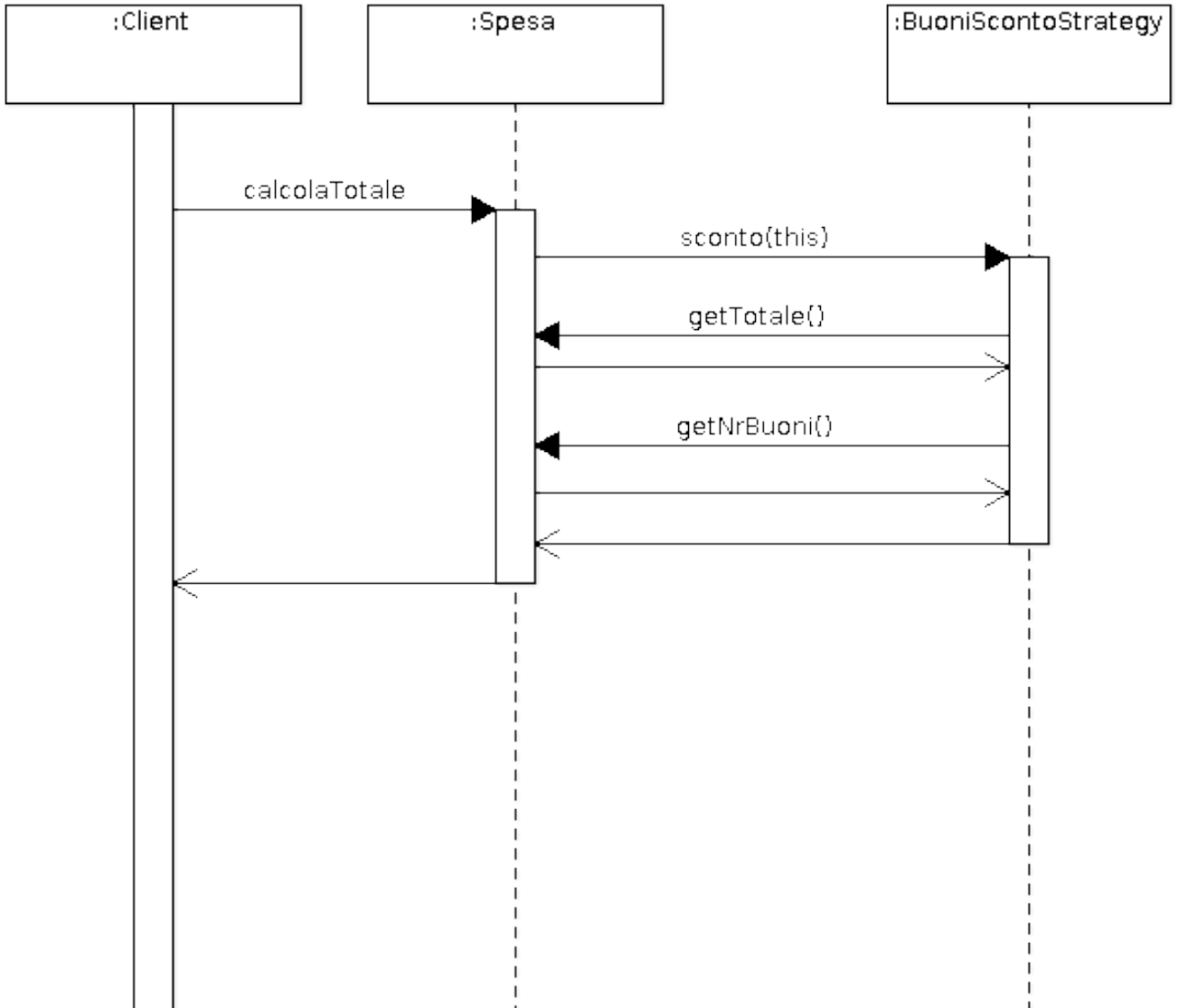
Tale riferimento viene passato come parametro nella chiamata del metodo sconto.

Tale riferimento viene usato per recuperare tutte le informazioni riguardanti la spesa per calcolare lo sconto in base alla strategia.

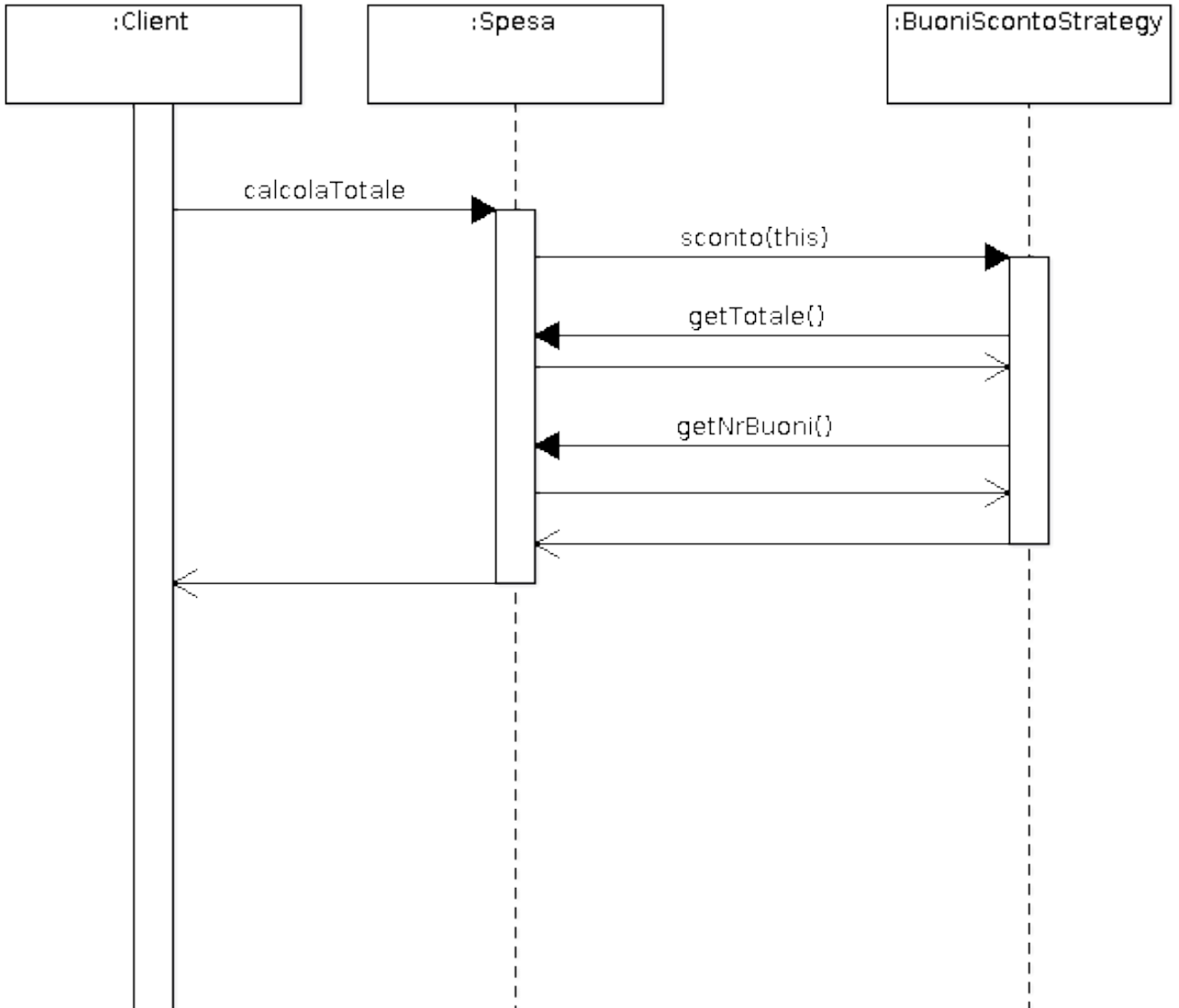
Strategy


:Client
:Spesa
:Strategy
calcolaTotale
sconto(this)
getTotale()

Strategy


:Client
:Spesa
:BuoniScontoStrategy
calcolaTotale
sconto(this)
getTotale()
getNrBuoni()

Strategy


:Client
:Spesa
:BuoniScontoStrategy
calcolaTotale
sconto(this)
getTotale()
getNrBuoni()

StrategyFactory

Da dove ricava l'istanza della strategia l'oggetto spesa per poter invocare il metodo sconto?

In questo esempio, possiamo usare Factory:

```
public class StrategyFactory{
   public Strategy create(){
    String className = file.readLine();
    Strategy strategy = (Strategy)
          Class.forName(className).newInstance();
    return strategy;
   }
}
```

Strategy

Tipicamente il nome della classe da usare per calcolare lo sconto viene memorizzato in una opportuna tabella del DB assieme alla data di inizio e fine validità dello sconto stesso.

In questo modo con una opportuna e semplice query la Factory ad ogni invocazione puo' creare l'istanza della strategy opportuna.

```
select className from sconti where now()>inizio
                                   and now()<fine
```

Strategy

E' possibile prevedere che la tabella sconti oltre al nome della classe gestisca anche eventuali parametri in modo da riusare la stessa classe con parametri diversi. Per fare questo bisogna aggiungere all'interfaccia degli opportuni metodi che permettano di impostare i vari parametri (es. setParam1(double)) e nella Factory chiamare tali parametri con i valori ricavati dalla riga della tabella sconti

Esercizio

Usate un file di property per gestire Strategie di sconto con data-ora di inizio e fine e (max) 3 parametri.

Composite

Esempio: il calcolo del totale dello scontrino nei supermercati:

1) anziani con sconto 10% il mercoledì;

2) buoni spesa di 5€ con un minimo di 30€

3) sconto 15% con scarico punti dalla tessera

4) sconto del 10€ su spese sopra i 100€

Se un anziano con tessera e buoni si presenta il mercoledì e compra 105$ di merce?

Assumiamo che il supermercato scelga di non combinare le offerte ma scelga lo sconto più favorevole al cliente.

Composite

Come facciamo a trattare un gruppo di oggetti nello stesso modo di un oggetto non composto?

Definisci una interfaccia comune per gli oggetti composti e per quelli atomici.

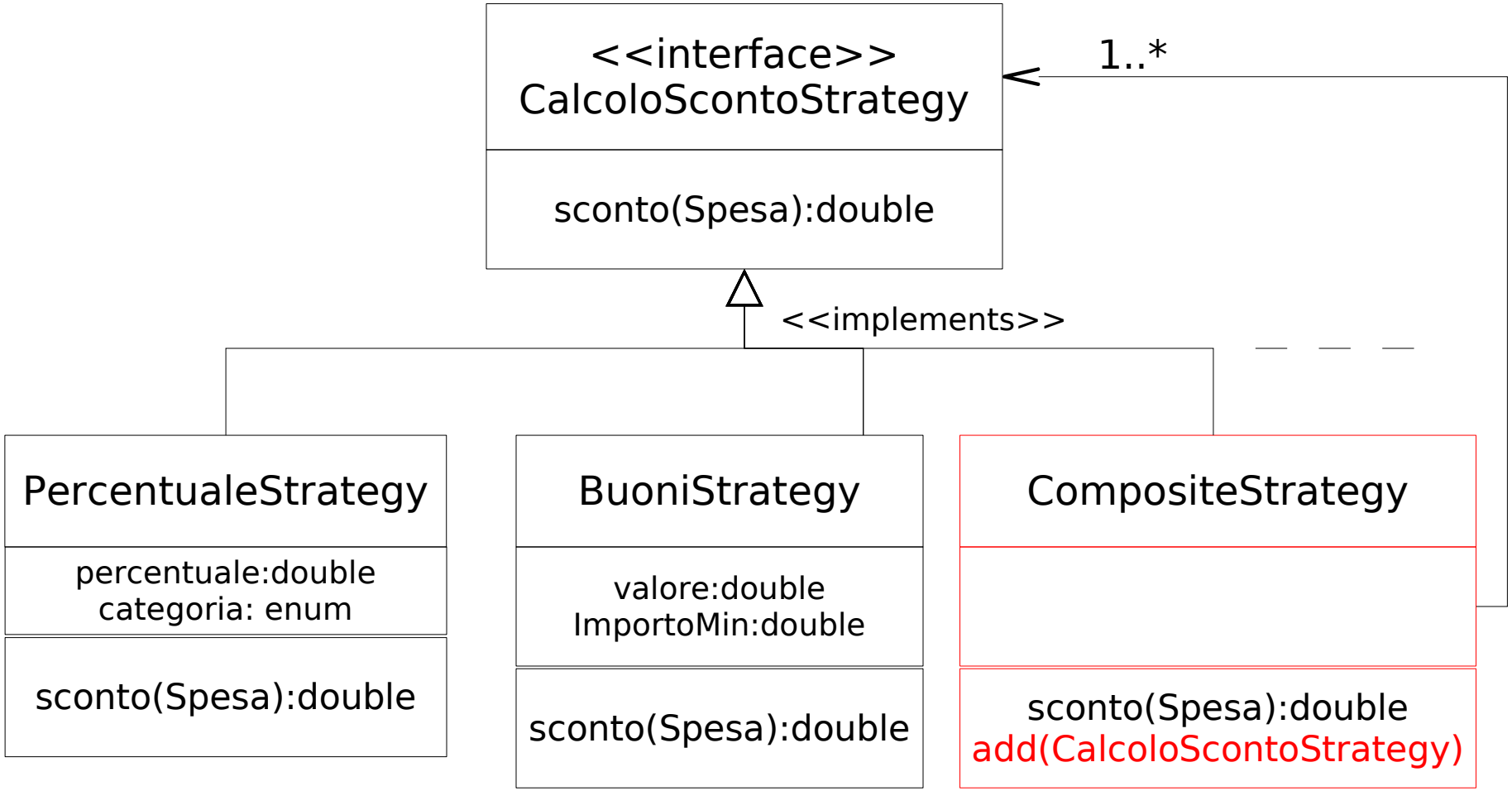
Composite

Come facciamo a trattare un gruppo di strategie di sconto nello stesso modo di una singola strategia?

(In pratica ci piacerebbe aggiungere una nuova strategia composta senza dover riscrivere niente)

Il gruppo di strategie implementa l'interfaccia della strategia singola.

Composite


<<interface>>
CalcoloScontoStrategy
sconto(Spesa):double
1..*
<<implements>>
PercentualeStrategy
percentuale:double
categoria: enum
sconto(Spesa):double
BuoniStrategy
valore:double
ImportoMin:double
sconto(Spesa):double
CompositeStrategy
sconto(Spesa):double
add(CalcoloScontoStrategy)

Esempio Implementazione

```
public class CompositeStrategy{
  Set<CalcoloScontoStrategy> strategie;
  public void add(CalcoloScontoStrategy css){
    strategie.add(css);}
  public double sconto(Spesa s){
   double max = 0.0;
   for (CalcoloScontoStrategy css: strategie)
     if (css.sconto(s)>max)
          max=css.sconto(s);
  return max;}}
```

Composite

Esempio: il calcolo del totale dello scontrino di un supermercato reale:

1) prodotti in offerta scontati 20% solo clienti con tessera;

2) prodotti scontati 30 % con buono solo su prodotti non in offerta max 2 prodotti al giorno solo clienti con tessera;

3) buoni spesa di 5€ con un minimo di 30€ di spesa tutti i clienti

4) sconto del 10%, 15% o 20% sull'intera spesa con scarico punti dalla tessera

Composite

In questo caso neanche Composite potrebbe essere sufficiente.

Infatti ci vuole una (nuova) strategia che faccia il merge di tutte le condizioni di sconto (e non che ne scelga una in particolare).

In ogni caso, resta il fatto che l'interfaccia Strategy nasconde al client l'implementazione effettiva.

Nota che per implementare questa Strategy bisogna accedere anche ai dati del cliente tramite l'oggetto Spesa.

Interprete

Abbiamo già detto che per proteggere un'applicazione dalle variazioni è possibile applicare il Pattern Protect Variations un caso particolare di questo Pattern e' il Interpreter Driven: in questo caso una (o più) stringhe di dati rappresentano un comportamento che viene interpretato per essere eseguito. Cambiando le stringhe (dati) è possibile cambiare il comportamento dell'applicazione

Interprete

Nell'esempio che segue:

- Il linguaggio interpretato è un semplice linguaggio con espressioni, variabili, costanti double, assegnamenti ;
- Non è orientato agli oggetti
- Non è tipato (o meglio viene gestito il solo tipo double)

Interprete

Per evidenziare come sia possibile integrare tale interprete in contesti applicativi diversi si prevede:

1) Possibile aggiungere nuove funzioni
2) Semantica delle variabili personalizzabile
3) Interfaccia Parser per prevedere futuri parser per diversi linguaggi

Sintassi

Esempi:

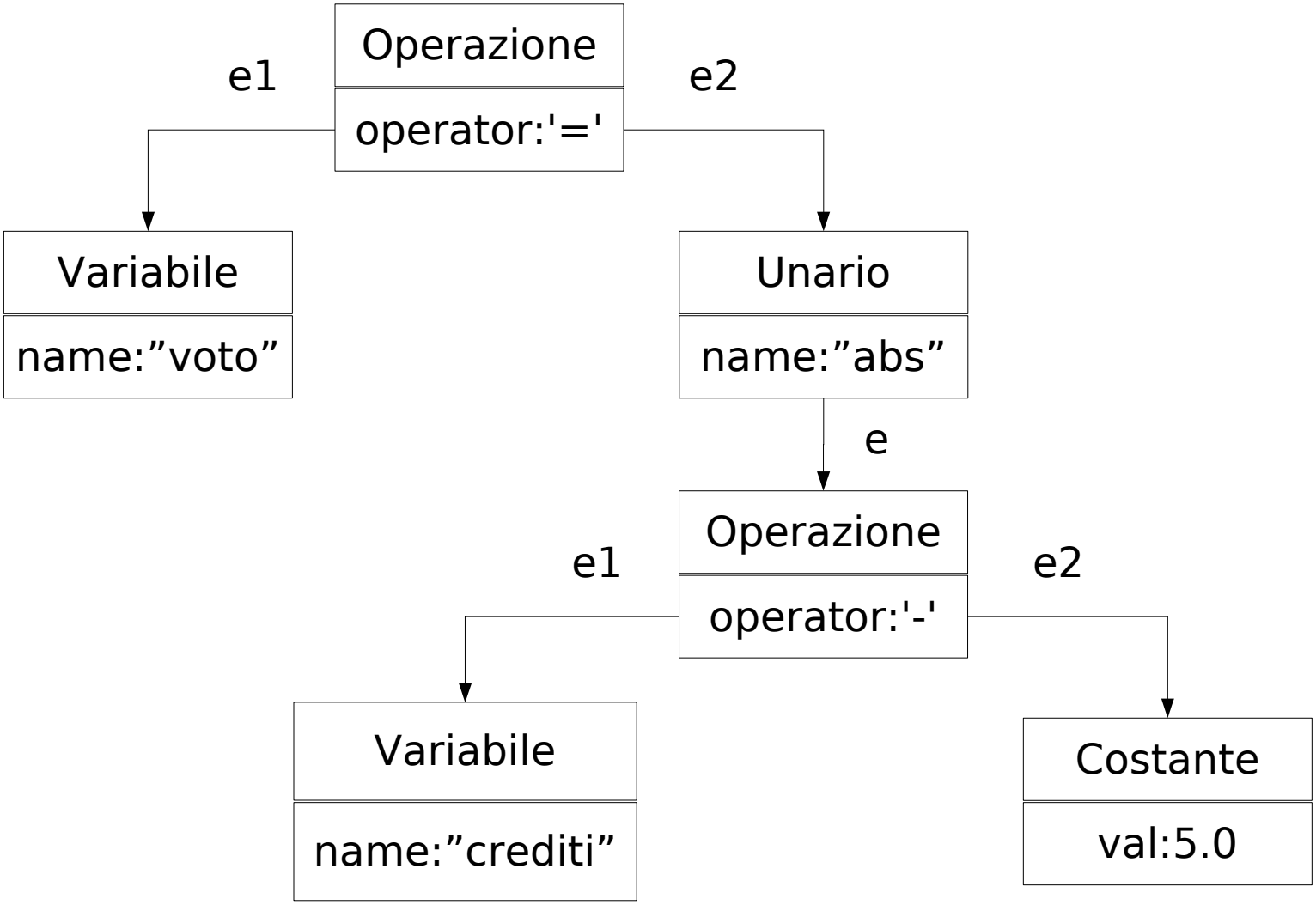
- 3+5;
- 4*(4.5+5)/(1/(1.0+5)
- myfun(4+5.6)
- voto+4.5*crediti/3.4
- voto=voto/5
- voto=crediti=0
- voto=4.5*crediti,crediti=4,voto

Interprete

- Sintassi sufficiente come esempio;
- Per sintassi più complesse conviene usare un generatore di automatico di parser (vedi Yacc e seguenti)
- Manca adeguata gestione errori sintattici
- Mancano gli operatori unari (- - - 1)
- Prototipo a scopo didattico
- Viene creata una struttura dati che rappresenta l'albero sintattico

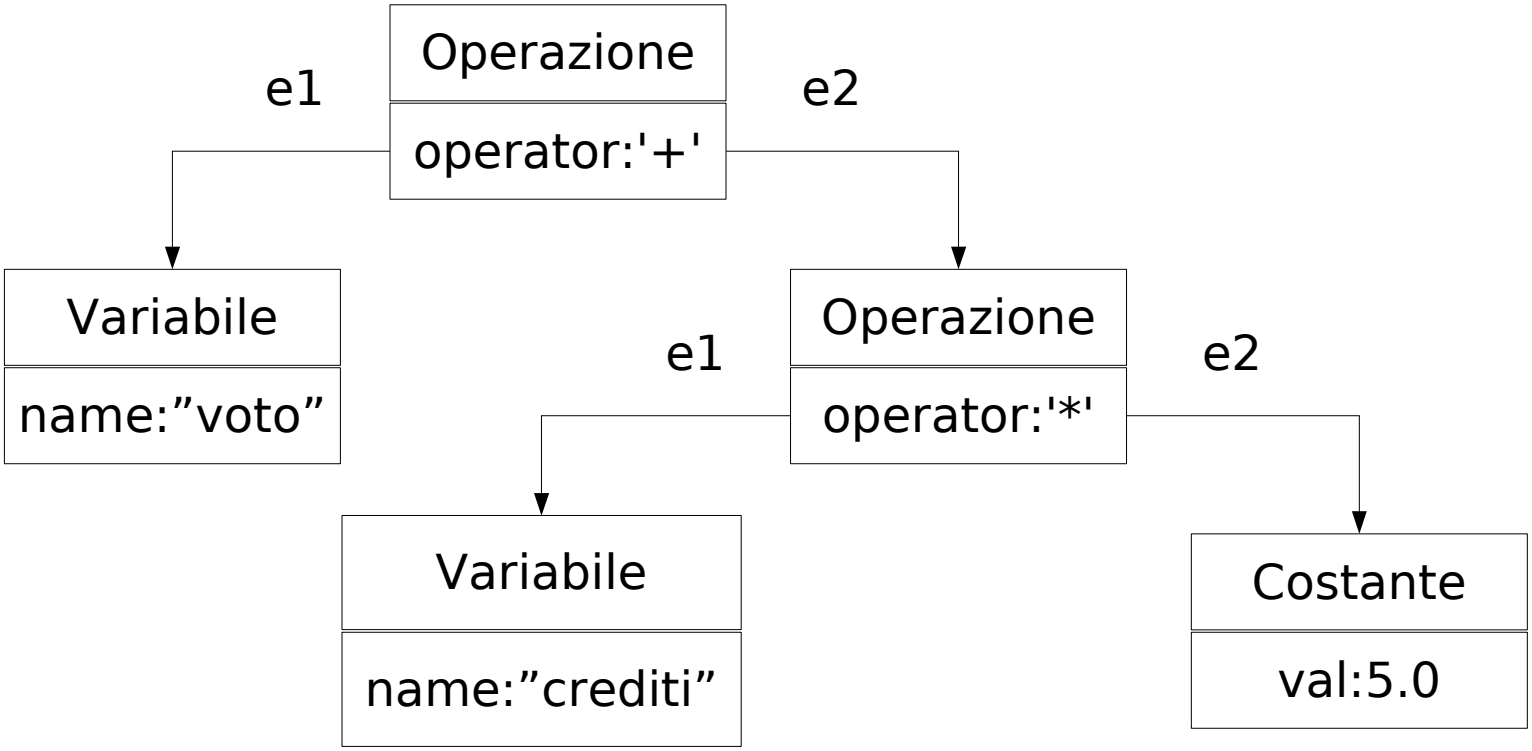
Esempio traduzione

- voto=abs(crediti-5.0)



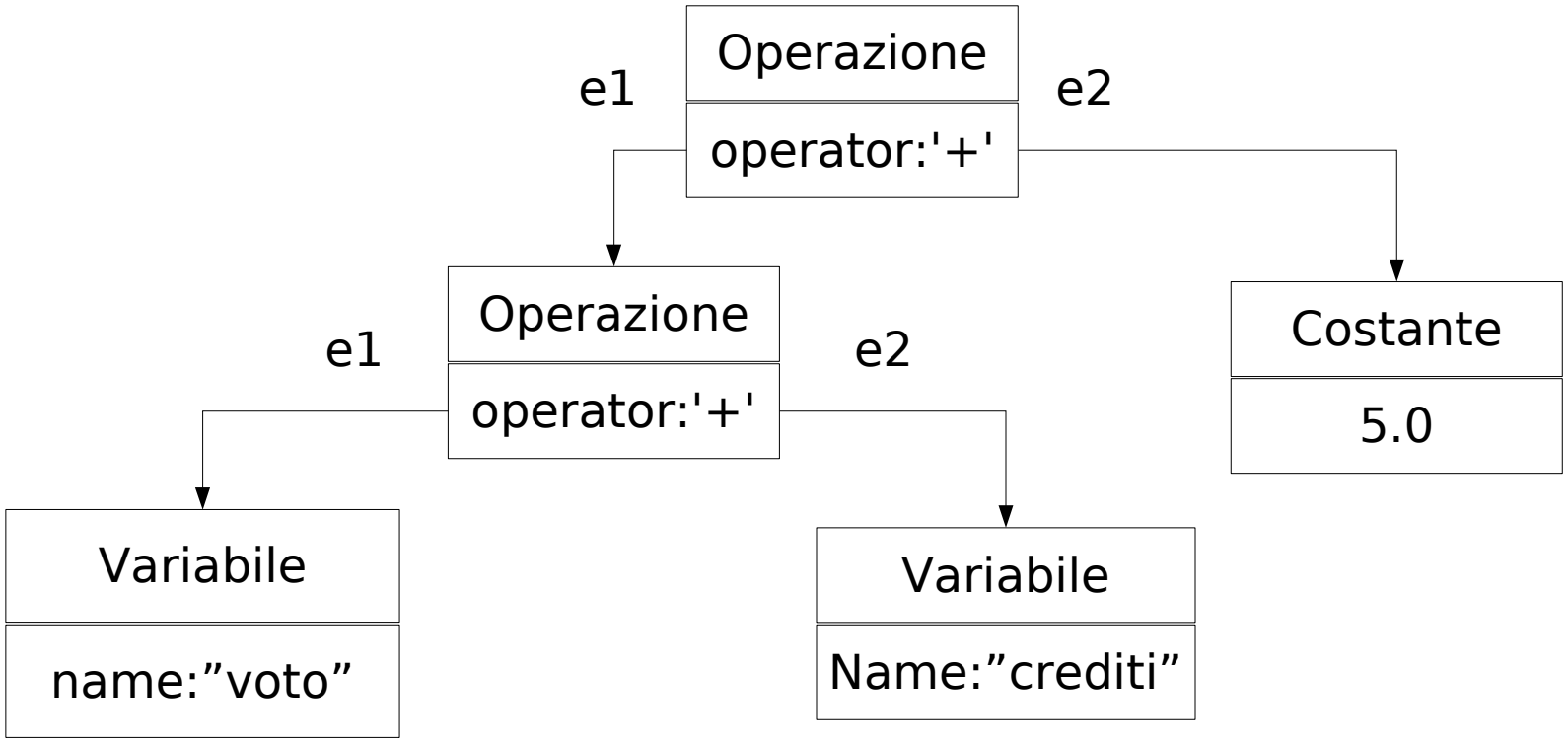
Priorità operatori

- voto+crediti*5.0



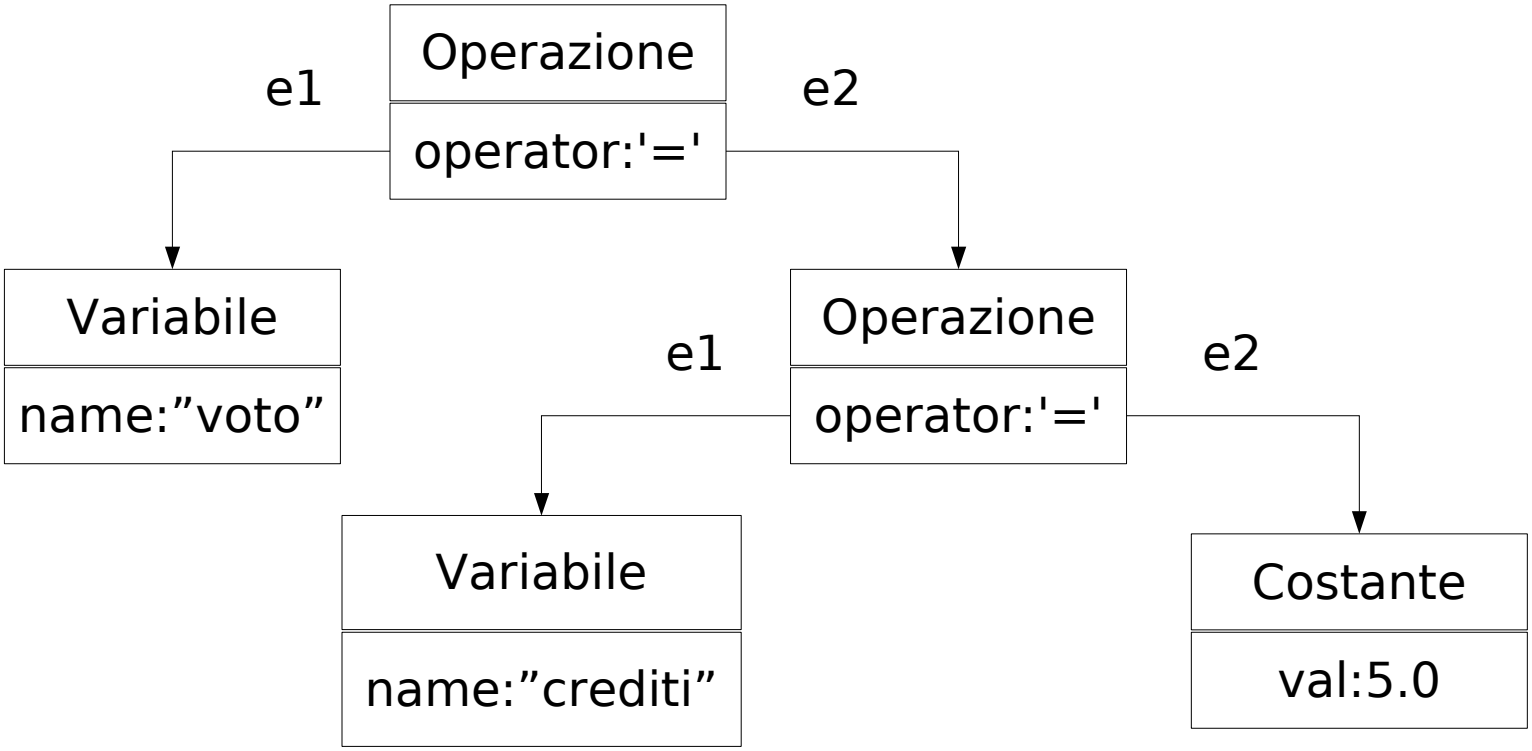
Priorità da sinistra a destra

- voto+crediti+5.0



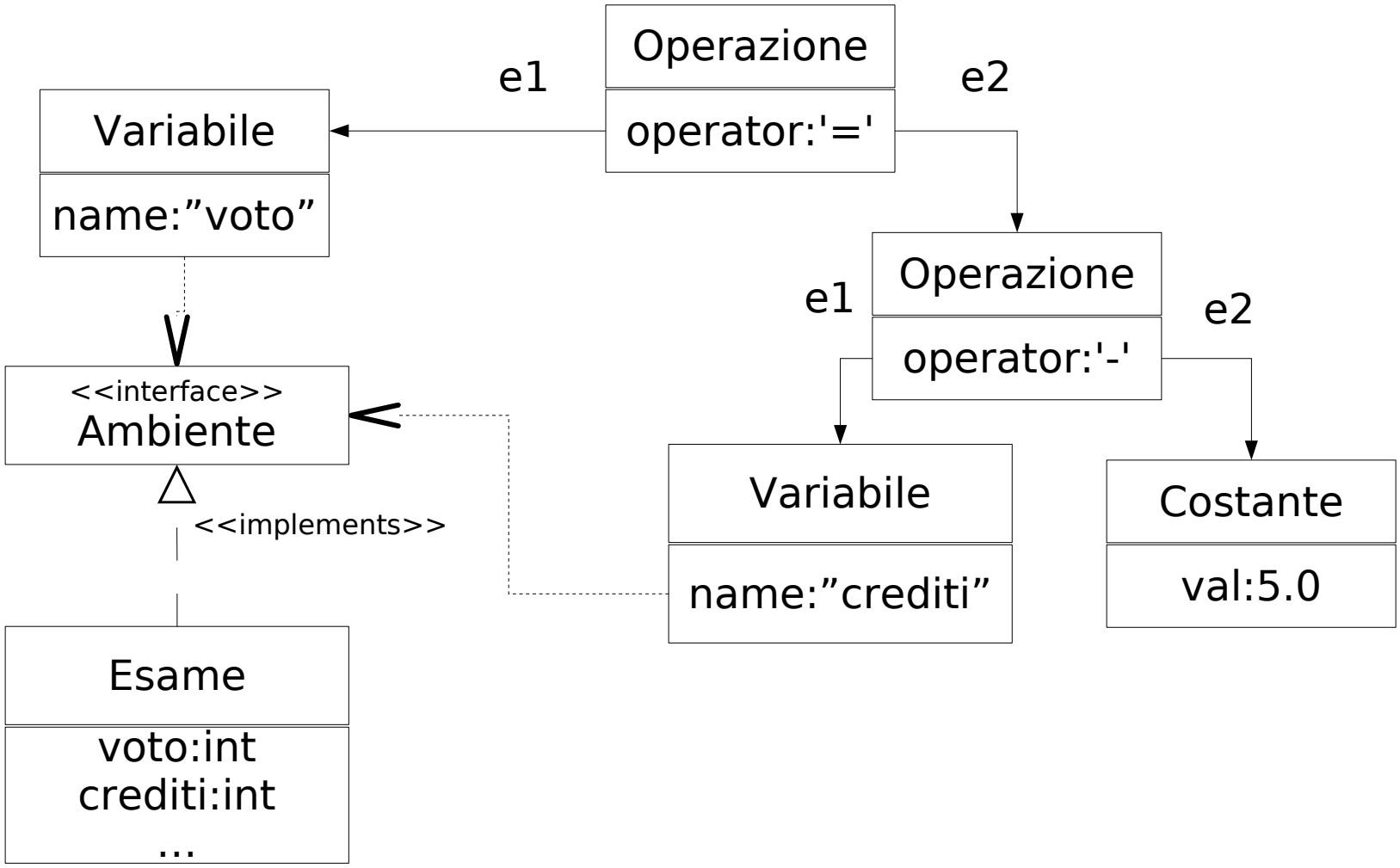
Priorità da destra a sinistra

- voto=crediti=5.0



Esempio variabili

- voto=crediti-5.0



Facile riuso

Per riusare l'interprete in contesti diversi senza dover modificare nessuna riga di codice del parser stesso sono utilizzati:

- L'interfaccia Ambiente per definire come recuperare le variabili
- Possibilità di definire le funzioni da usare nell'interprete

Ambiente

```
public interface Ambiente {
    double getVariableValue(String name);
    double setVariableValue(String name, double val);
}
```

I 2 metodi permettono rispettivamente di recuperare e impostare il valore di una variabile di nome “name”.

Il metodo che imposta il valore della variabile ritorna anche il valore impostato in modo che possa essere utilizzato negli assegnamenti in cascata

Es voto=credito=5.5;

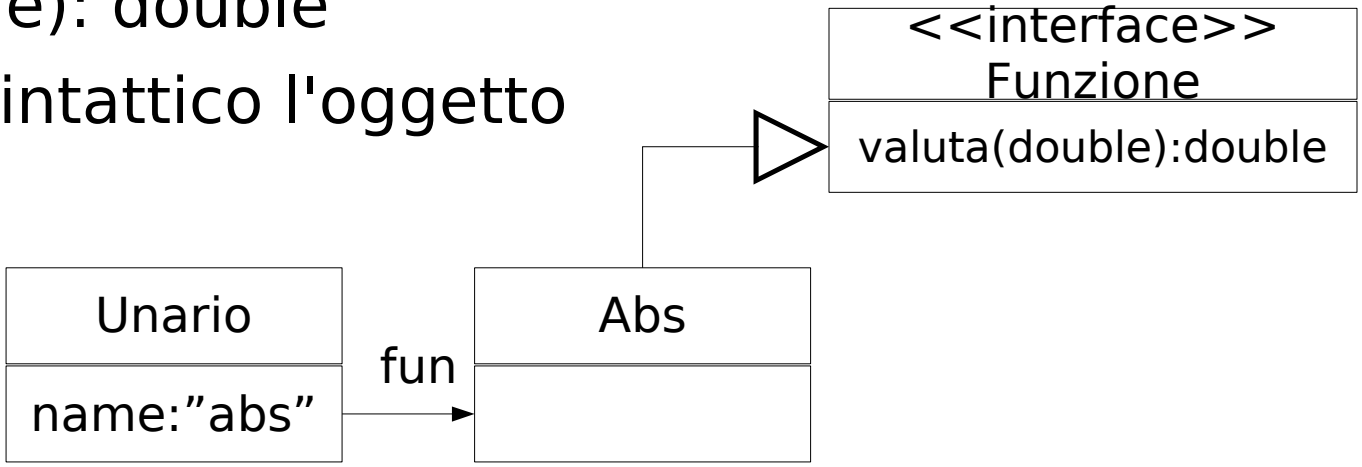
E' compito di chi vuole utilizzare l'interprete implementare questi metodi in modo da definire la semantica opportuna (vedi classe Esame)

Funzioni

Il parser riconosce come chiamate a funzione tutti gli identificativi Java a cui segue la sintassi per l'applicazione di funzione. Es. abs(-3)

Ma ogni funzione effettivamente usata dalla particolare istanza del parser deve essere registrata nel parser col metodo addFunzione(name,fun) dove fun è oggetto di una classe che implementa l'interfaccia Funzione che contiene il solo metodo valuta(double): double

Nell'albero sintattico l'oggetto



Esempio traduzione

- voto=abs(crediti-5.0)