Corso di Laurea Ingegneria Informatica
Fondamenti di Informatica 2

Dispensa 07

Introduzione ai Tipi astratti di dato

A. Miola

Febbraio 2008

Contenuti

- **Soluzione automatica di problemi**
  - Specifica e rappresentazione di un algoritmo
  - Tipi astratti e tipi concreti
  - Specifica di un tipo astratto
- **Il tipo astratto Insieme**
  - Specifica del tipo astratto Insieme
  - Operazioni primitive e non
- **Rappresentazione del tipo astratto Insieme**
  - Rappresentazione mediante interfacce
  - La classe Insieme
  - Efficienza della rappresentazione

Soluzione automatica di problemi . . .

- La soluzione automatica di un problema richiede di svolgere attività piuttosto articolate che comprendono:
  - la specifica di un algoritmo risolutivo
  - l'implementazione dell'algoritmo nel linguaggio di programmazione prescelto
- In effetti, come abbiamo già avuto modo di vedere anche nei casi di semplici problemi, ciascuna delle due attività coinvolge molteplici fasi (in particolare la valutazione della correttezza della sua implementazione)

. . . Soluzione automatica di problemi

- In particolare è da osservare che la specifica dell'algoritmo risolutivo comprende due attività strettamente correlate tra loro:
  - la specifica dei dati e delle loro proprietà
  - la specifica delle operazioni da compiere su di essi al fine di pervenire alla soluzione del problema posto
- Questa osservazione dovrebbe essere ormai evidente
  - Pensiamo ad esempio al caso del problema della ricerca: diversi insiemi di dati e diverse loro proprietà determinano la scelta di algoritmi diversi, più o meno efficienti.

Rappresentazione di un algoritmo

- **Altrettanto possiamo dire circa l'attività di implementazione dell'algoritmo in un linguaggio di programmazione**
  - **si tratta infatti di implementare (rappresentare) sia i dati che le operazioni su di essi, come un'unica attività di rappresentazione di un algoritmo**
    - **Si pensi alla specifica dell'algoritmo per il calcolo del Massimo Comun Divisore di due numeri interi x e y che prevede la specifica e la determinazione di:**
      - **l'insieme X dei divisori di x e l'insieme Y dei divisori di y**
      - **l'insieme Z intersezione di X e di Y**
      - **il massimo valore dell'insieme Z**
- **Si pone quindi l'esigenza di rappresentare in particolare gli insiemi di numeri interi con relative operazioni**

Rappresentazione di dati e operazioni . . .

- Questa attività di rappresentazione di dati e operazioni consiste essenzialmente nel determinare una corrispondenza tra i dati, cosi come specificati nell'algoritmo, e le tipologie di dati, o di strutture di dati, disponibili nel linguaggio di programmazione prescelto
  - Si pensi ad esempio ai tipi primitivi e riferimento di Java e alla struttura di array di Java
- In maniera analoga si tratta di ragionare circa la rappresentazione delle operazioni sulle rappresentazioni dei dati

. . . Rappresentazione di dati e operazioni

- Riferiamoci al tipo Java `int`. Quando dichiariamo un dato di tipo `int` , numero intero, non solo stiamo indicando la modalità della sua rappresentazione binaria nelle celle della memoria nonché il suo valore massimo consentito, ma anche quelle che saranno le operazioni consentite, ammissibili, su di esso

- Nella dichiarazione del tipo di un dato c'è, per così dire, un aspetto statico e un aspetto dinamico

Tipi astratti e tipi concreti . . .

- Supponiamo che venga posto un problema relativo ai numeri interi
  - ad esempio il semplice problema di determinare il prodotto di due numeri
- Sappiamo bene che volendo risolvere un problema come questo, magari anche manualmente, non possiamo operare "astrattamente" sui numeri interi intesi come tipi astratti, ma dobbiamo passare ad una rappresentazione dei tipi astratti mediante tipi concreti come i numerali corrispondenti tipicamente rappresentati in base 10

. . . Tipi astratti e tipi concreti

- Creiamo quindi una corrispondenza tra il tipo astratto dei numeri interi e il tipo concreto dei numerali in base 10
- Se invece vogliamo risolvere automaticamente il problema posto, ad esempio utilizzando Java, dobbiamo creare una corrispondenza tra il tipo astratto dei numeri interi e il tipo `int` di Java
  - Anche nel caso di linguaggi di programmazione parliamo di tipi concreti per riferirci ai tipi del linguaggio
  - Il tipo `int` è un tipo concreto di Java

Tipi astratti . . .

- Consideriamo il tipo astratto dei numeri naturali con le relative operazioni, in genere denotato da

$$\mathbf{Nat} = \langle N , + , - , * , = , < , 0 , 1 \rangle$$

  - l'insieme di tutti i numeri naturali: **N**
  - un insieme di operazioni (funzioni o predicati):

$$+ , - , * , = , <$$

  - un insieme di costanti: **0** e **1** , che consentono di generare tutti gli elementi di **N** mediante le operazioni disponibili

. . . Tipi astratti

- In generale un tipo astratto, inteso come ente matematico, è definito come
  - un insieme di oggetti (detto il dominio del tipo)
  - un insieme di operazioni (funzioni o predicati del tipo)
  - un insieme di costanti (che denotano alcuni elementi del dominio del tipo)
- Nell'esempio del tipo astratto dei naturali
  - un insieme di oggetti { N }
  - un insieme di operazioni { + , - , * , = , < }
  - un insieme di costanti { 0 , 1 }

Un esempio: il tipo astratto Insieme

- Tipo astratto Insieme di oggetti di un certo dominio V
  - 1. Il dominio Ins

    (tutti gli insiemi di oggetti di un certo dominio V)
    - Il dominio Ins, dominio del tipo, è detto anche dominio di interesse;
    - V è detto dominio di sostegno
  - 2. Le operazioni
    - Vuoto (verifica se un insieme è vuoto)
    - Inserisci (inserisce un elemento)
    - Cancella (cancella un elemento)
    - Contiene (verifica se un insieme contiene un elemento)
  - 3. La costante Insieme_Vuoto

La specifica delle operazioni . . .

Vuoto

Dato un insieme H (cioè un elemento di Ins, dominio di interesse), verifica se esso contiene o meno elementi

Vuoto : Ins ⟶ Boolean

Inserisci

Dato un insieme H e un oggetto E, appartenente all'insieme V, fornisce come risultato l'insieme costituito da tutti gli elementi di H con l'aggiunta dell'elemento E (se non presente), oppure l'insieme H

Inserisci : Ins x V ⟶ Ins

. . . La specifica delle operazioni

Cancella

Dato un insieme H e un elemento E di V, fornisce come risultato l'insieme costituito da tutti gli elementi di H tranne l'elemento E (se presente), oppure l'insieme H

Cancella : Ins x V ⟶ Ins

Contiene

Dato un insieme H e un elemento E di V, verifica se E e' contenuto in H

Contiene : Ins x V ⟶ Boolean

Il tipo astratto Insieme

- Va notato che nella specifica delle operazioni si fa riferimento a vari domini: oltre al dominio Ins, si fa infatti riferimento anche ai domini V e Boolean
- La denotazione del tipo di dato astratto Insieme è quindi data dalla terna

$$< S , F , C >$$

  - S = { Ins , V , Boolean }
  - F = { Vuoto , Inserisci , Cancella , Contiene }
  - C = { Insieme_Vuoto }

Operazioni primitive e non

- Le operazioni specificate in un tipo astratto di dato, vengono dette operazioni primitive
  - Nel tipo astratto dei numeri naturali le operazioni + , – e * sono operazioni primitive
  - Nel tipo astratto Insieme le operazioni Vuoto, Inserisci, Cancella e Contiene sono operazioni primitive
- A partire dalle operazioni primitive è possibile definire altre operazioni, come ad esempio l'operazione Unione per calcolare l'unione di due insiemi

Unione : Ins x Ins ⟶ Ins

Operazioni primitive e non

Unione

Dato un insieme A e un insieme B fornisce come risultato l'insieme costituito da tutti gli elementi di A ampliato con tutti gli elementi dell'insieme B (non presenti in A), oppure l'insieme A

```
. . .
   if   Vuoto(B) risultato = A
     else considera un qualunque elemento b di B
        if Contiene(A,b) risultato = Unione  (A, Cancella (B, b) )
        else risultato = Inserisci (Unione  (A, Cancella (B, b) ), b) ;
. . .
```

Rappresentazione di tipo un astratto . . .

- I tipi astratti stanno ai tipi concreti, propri di un linguaggio di programmazione, come gli algoritmi stanno ai programmi in quel linguaggio di programmazione
- Per poter operare effettivamente (automaticamente) su dati astratti è necessario determinare una loro rappresentazione nel contesto linguistico di un linguaggio di programmazione scelto

Rappresentazione di tipo un astratto

- Per rappresentare un tipo astratto < S , F , C > è necessario rappresentare:
  - il dominio di interesse e ciascuno degli altri domini in S
  - ciascuna delle operazioni in F
  - ciascuna delle costanti in C
- Riprendiamo l'esempio del tipo astratto Insieme di oggetti di V e poniamoci l'obiettivo di determinare una sua rappresentazione in Java

Rappresentazione del tipo astratto Insieme

- Consideriamo il caso degli insiemi costituiti da oggetti che siano numeri interi appartenenti al dominio Z
- Partiamo dalla rappresentazione dei domini di S = { Ins , Z , Boolean }
- In questo caso il dominio Z può essere direttamente rappresentato dal tipo primitivo Java `int`, così come Boolean è rappresentato dal tipo primitivo Java `boolean`
- Si tratta ora di determinare una rappresentazione per il dominio di interesse Ins

Rappresentazione del dominio di interesse

- Per essere ancora più concreti, consideriamo il caso degli insiemi costituiti da numeri interi compresi tra 1 e 5
  - In questo caso gli elementi che possono comporre i nostri insiemi sono quindi appartenenti al dominio di sostegno $V = \{1, 2, 3, 4, 5\}$
- Consideriamo ad esempio l'insieme

$$H = \{2, 3, 5\}$$

Rappresentazione del dominio di interesse

- Un qualsiasi insieme H del dominio Ins può essere rappresentato da un oggetto array di 5 elementi di tipo `boolean[5]`

```
class Insieme {
  private boolean[] ins; //array degli elementi . . .
```

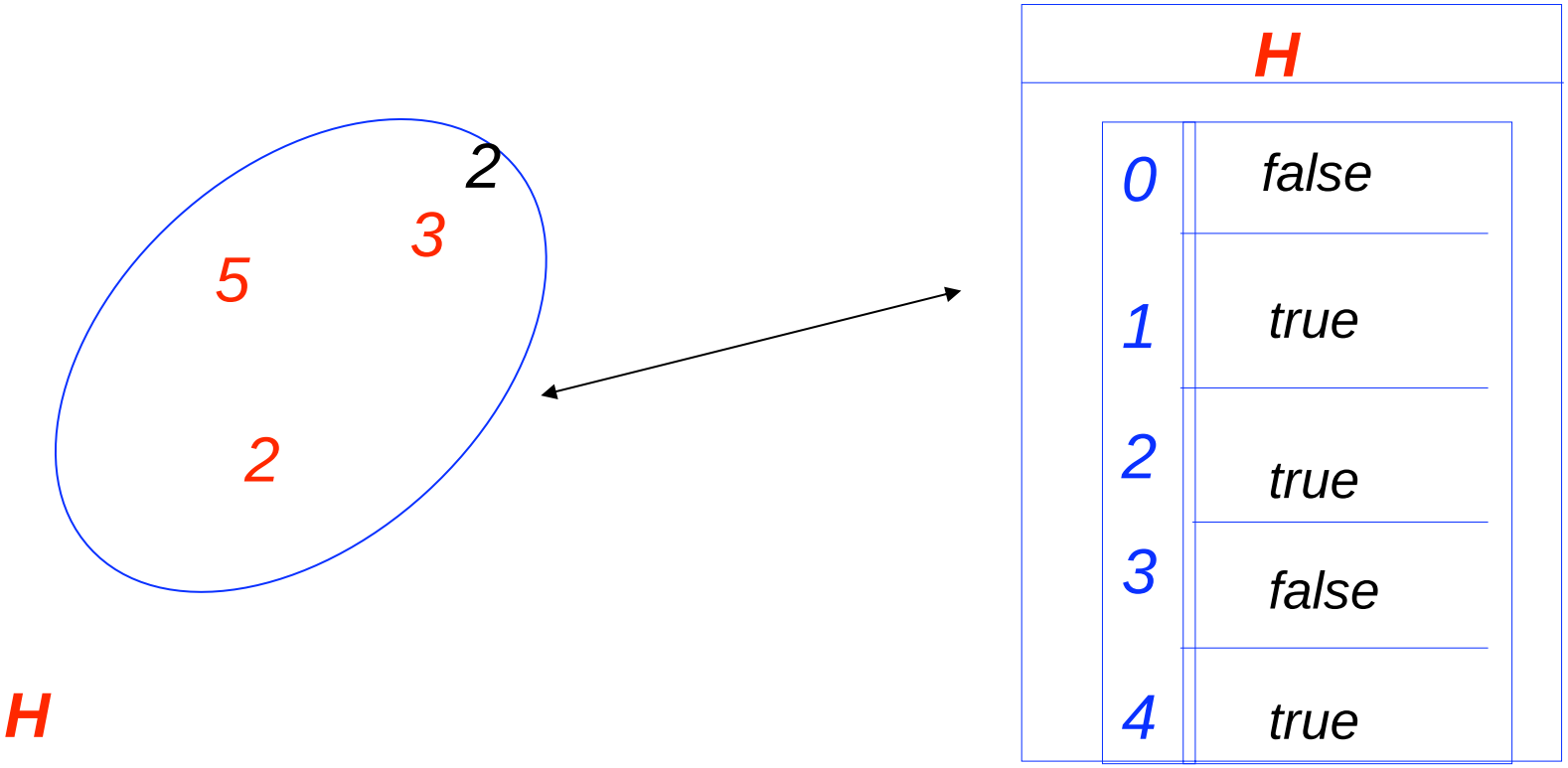
- Si stabilisce quindi una corrispondenza biunivoca tra l'insieme dei valori del tipo astratto Insieme di elementi di

$$V = \{ 1 , 2 , 3 , 4 , 5 \}$$

e l'insieme dei valori di tipo array di 5 elementi di tipo `boolean`

Rappresentazione del tipo astratto Insieme

Il valore del generico elemento ins[k] dell'array sarà true se il valore dell'intero k+1 appartiene ad H, false altrimenti.



Rappresentazione dei domini

- Rappresentazione dei domini del tipo astratto Insieme

Z	→	`int`
Boolean	→	`boolean`
Ins	→	`Insieme`

Specifica delle operazioni mediante interfacce . . .

- Per rappresentare il tipo astratto Insieme dobbiamo ora rappresentare tutte le sue operazioni primitive.
  - Per ciascuna di esse dobbiamo definire un metodo di cui, almeno inizialmente, ci limitiamo a descrivere il comportamento attraverso la sua interfaccia, cioè la descrizione del suo comportamento rispetto all'utente.
- Le interfacce di tutti i metodi relativi ad un tipo possono essere descritte nel loro complesso attraverso la dichiarazione `interface` di Java.

. . . Specifica delle operazioni mediante interfacce

- Nell'interfaccia il corpo del metodo non viene fornito, successivamente costruiremo una classe che implementi l'interfaccia, con l'implementazione di tutti i metodi in essa descritti
- Vediamo quindi l'interfaccia per le operazioni del nostro tipo Insieme con la definizione dei prototipi dei metodi relativi alle diverse operazioni

Interfaccia InsiemeDiInteri

```
interface InsiemeDiInteri {
  boolean isEmpty();
  //post: restituisce true sse l'insieme e' vuoto

  void addElement(int n);
  //pre: n > 0
  //post: inserisce il numero n nell'insieme

  void remove(int n);
  //pre: n > 0
  //post: rimuove il numero n dall'insieme

  boolean contains(int n);
  //pre: n > 0
  //post: restituisce true sse l'insieme contiene n
}
```

La classe Insieme . . .

```
class Insieme implements InsiemeDiInteri {
   private boolean[] ins;

   Insieme() {                          //costruttore
      ins = new boolean[5];
      for (i = 0; i < ins.length; i++)
          ins[i] = false;
   }
                . . . Segue . . .
```

La classe Insieme . . .

```
boolean isEmpty() {
    //post: restituisce true sse
    //l'insieme e' vuoto
    int i;
    boolean vuoto = true;
    for (i = 0; i< ins.length; i++)
         if ins[i] vuoto = false;
    return vuoto;
}
```

. . . Segue . . .

. . . La classe Insieme

```
boolean contains(int n) {
  //pre: n > 0
  //post: restituisce true sse l'insieme
  //       contiene n
     boolean appartiene;
     appartiene = ins[n-1];
     return appartiene;
  }
            . . . Da completare . . .
}
```

Due possibili interpretazioni . . .

- Per quanto riguarda il completamento dell'implementazione dei metodi va notato che esistono due possibili interpretazioni, o due possibili semantiche, per l'operazione **Inserisci** (un elemento)

$$\text{Inserisci} : \quad \text{Ins} \times V \longrightarrow \text{Ins}$$

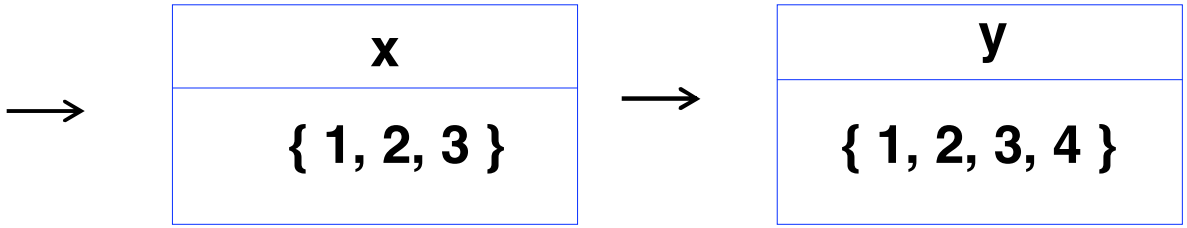
che hanno una diretta conseguenza sull'implementazione dell'operazione **Inserisci** con il metodo `addElement`

Due possibili interpretazioni . . .

- **Prima interpretazione (funzionale)**

```
/* Prototipo del metodo per
l'inserimento che restituisce un nuovo
insieme */
 Insieme addElement(Insieme i, int n)
```

```
y = addElement(x,4);
```

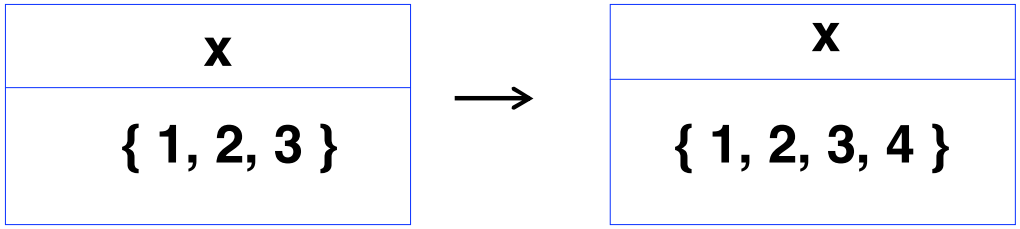


Due possibili interpretazioni . . .

- Seconda interpretazione (orientata agli oggetti) con un side-effect sulla variabile d'istanza `ins`

```
/* inserisce nell'insieme ricevente */
 this.addElement(int n);
```

`addElement(4)` →



Esercizi

Tenendo conto che noi siamo interessati alla semantica orientata agli oggetti procedere nei seguenti esercizi

- Completare con tutti i metodi necessari la classe `Insieme`
- Aggiungere anche il metodo `copy` per costruire una copia di un oggetto `Insieme`
- Discutere in che senso la disponibilità del metodo `copy` consente di procedere in modo funzionale anche nel paradigma orientato ad oggetti
- Modificare la classe `Insieme` per rappresentare insiemi di numeri interi compresi tra 1 e n > 1

Esercizi

- Definire opportuni metodi per rappresentare le operazioni non primitive unione (già definita precedentemente) e intersezione di insiemi di numeri interi
  - Sfruttare le caratteristiche della rappresentazione del tipo astratto Insieme adottata e basata sull'uso di array di boolean per migliorare l'efficienza dell'implementazione

Efficienza della rappresentazione

```
Insieme unione(Insieme a, Insieme b) {
  int i;
  Insieme risultato;
  risultato = new Insieme();
  for (i=0; i<a.ins.length; i++)
      risultato.ins[i] =
          a.ins[i] || b.ins[i];
  return risultato;
}
```

Versione funzionale

Un altro esempio: i numeri razionali

- Consideriamo il tipo astratto *Q* dei numeri razionali, ricordando quanto già visto nel corso precedente circa la classe `Razionale`
  - Gli elementi a/b del dominio di interesse possono essere denotati da una coppia di numeri interi (a, b) : a denota il numeratore della frazione, mentre b denota il denominatore
  - Le operazioni del tipo *Q* sono le usuali operazioni : Addizione, Moltiplicazione, ecc.
- In effetti anche Coppia è un tipo astratto di dato che può essere opportunamente rappresentato da una classe `Coppia`

Il tipo astratto Coppia . . .

- Il tipo astratto Coppia rappresenta il prodotto cartesiano di due domini
- I valori di tale tipo sono coppie di valori presi da due domini specificati
- Le operazioni associate sono semplicemente
  - la costruzione – mediante il costruttore
  - il calcolo del valore della prima componente
  - il calcolo del valore della seconda componente

. . . Il tipo astratto Coppia

La specifica del tipo Coppia, con i suoi domini (tra cui quello di interesse) e le sue funzioni, è quindi la seguente

Domini

- **Coppia : dominio di interesse del tipo**
- **T1 : dominio dei valori che possono comparire come prima componente delle coppie**
- **T2 : dominio dei valori che possono comparire come seconda componente delle coppie**

. . . Il tipo astratto Coppia

Funzioni

- formaCoppia(T1 a, T2 b) ⟶ Coppia
  - pre: nessuna
  - post: RESULT è l'oggetto corrispondente alla coppia la cui prima componente è a e la seconda è b
- primaComponente(Coppia c) ⟶ T1
  - pre: nessuna
  - post: RESULT è la prima componente della coppia c
- secondaComponente(Coppia c) ⟶ T2
  - pre: nessuna
  - post: RESULT è la seconda componente della coppia c

Rappresentazione del tipo Coppia . . .

- La realizzazione del tipo astratto Coppia come una classe Java `Coppia` è immediata
  - Rappresentiamo i valori del tipo astratto utilizzando due campi dati (variabili di istanza) di tipo opportuno per memorizzare le due componenti della coppia
  - Schema realizzativo: utilizziamo lo schema realizzativo con side-effect come illustrato precedentemente
  - Interfaccia di classe: come visto definiamo un metodo per ciascuna delle funzioni del tipo astratto
  - Realizziamo la funzione formaCoppia attraverso un costruttore

. . . Rappresentazione del tipo Coppia

```
public class Coppia {
// rappresentazione dei valori
   private Object c1;
   private Object c2;

// realizzazione delle funzioni del tipo astratto

   public Coppia (Object c1, Object c2) {
   // realizza formaCoppia
            this.c1 = c1;
            this.c2 = c2; }
   public Object primaComponente() { return c1; }
   public Object secondaComponente() { return c2; }
```

Alcune riflessioni

- Cosa è il tipo `Object` ?
- Che relazione c'è tra il tipo astratto *Q* dei numeri razionali e il tipo astratto Coppia ?
  - Il tipo *Q* è un caso particolare di Coppia
- Che relazione c'è tra la classe `Razionale` e la classe `Coppia` ?
  - In che senso la classe `Razionale` può essere intesa come un caso particolare della classe `Coppia` ?
  - Cosa vuol dire in Java che una classe è un caso particolare di un'altra classe ?
- Per rispondere a queste domande dobbiamo introdurre il concetto di ereditarietà e i relativi meccanismi di realizzazione