# Corso di Laurea Ingegneria Informatica
# Fondamenti di Informatica 2

Dispensa 11

## Tipi astratti di dato e loro rappresentazione

**A. Miola**

Marzo 2008

# Contenuti

- Definizione di Tipo astratto di dato
  - il tipo astratto Numero naturale
  - il tipo astratto Insieme
- Rappresentazione di un Tipo astratto
  - definizione di rappresentazione di un tipo astratto
  - rappresentazione del tipo astratto Insieme
- Correttezza della rappresentazione
  - Correttezza della rappresentazione del tipo astratto Insieme
- Realizzazione di tipi astratti in Java
- I tipi astratti Matrice e Vettore

# Prerequisiti

- Questo capitolo presuppone la conoscenza di tutti gli argomenti trattati in precedenza nel corso e in particolare la Dispensa n. 07 sull'Introduzione ai tipi astratti di dato (ADT)

# Definizione di Tipo astratto di dato

- **Un tipo astratto di dato è una tripla**

$$< S , F , C >$$

**dove**

- **S è un insieme di domini $\{ V_1, V_2, \ldots , V_n \}$, tra cui un dominio speciale chiamato "dominio di interesse"**
- **F è un insieme di funzioni (o operazioni) $\{ F_1, F_2, \ldots , F_m \}$, ciascuna delle quali è del tipo**

$$F_i : V_{i1} \times V_{i2} \times \ldots \times V_{ih} \longrightarrow V_{ik}$$

**dove ogni elemento di $\{V_{i1}, V_{i2}, \ldots , V_{ih}, V_{ik}\}$ è un elemento di S, e uno tra essi è il dominio di interesse**

  - **il dominio di interesse o è il codominio di $F_i$ oppure è uno dei domini del dominio (prodotto cartesiano) di $F_i$**
- **C è un insieme di elementi del dominio di interesse**

# Esempio . . .

- 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

# . . . Esempio

- **Secondo la definizione data con la tripla**

**< S , F , C >**

**il tipo astratto dei naturali è quindi definito come**

- **S è l'insieme di domini {N , Boolean }, con N "dominio di interesse"**
- **F è l'insieme di funzioni { + , - , * , = , < }**
- **C è l'insieme di costanti { 0 , 1 }**

# Il tipo astratto Insieme

- Secondo la definizione data con la tripla

$$< S , F , C >$$

la denotazione del tipo di dato astratto

Insieme è quindi data da

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

# Rappresentazione di un tipo astratto . . .

- Per poter operare effettivamente (automaticamente) su dati astratti è necessario determinare una loro rappresentazione nel contesto linguistico di un linguaggio di programmazione scelto
- I tipi astratti stanno ai tipi concreti, propri di un linguaggio di programmazione, come gli algoritmi stanno ai programmi in quel linguaggio di programmazione
  - Ad esempio in Java i tipi concreti sono i tipi primitivi e quelli riferimento (già disponibili come String e Array oppure definiti dall'utente come NodoLista)

# . . . Rappresentazione di un tipo 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
- Ricordiamo l'esempio del tipo astratto Insieme di oggetti di V  e la sua rappresentazione in Java già visto nella dispensa 07 su ADT

# Definizione di Rappresentazione di un tipo astratto

- **Una rappresentazione di un tipo T = < S , F , C > è una tripla**

  **< T' , *rapp* , OP >**

  **dove**

  - **T' = < S' , F' , C' > è il tipo mediante il quale si rappresenta T**
  - ***rapp* è una corrispondenza tra i valori dei domini del tipo T e i valori dei domini del tipo T'**
  - **OP è una funzione che associa ad ogni operazione primitiva definita per il tipo T una operazione (primitiva o non) definita per il tipo T'**

# Rappresentazione di un tipo astratto

- La definizione fa riferimento alla generale esigenza di rappresentare un tipo T con un altro tipo T' - *non necessariamente concreto*
- In generale per un tipo astratto si possono avere diverse rappresentazioni, alcune delle quali con tipi concreti relativi ad uno specifico linguaggio
- Ogni rappresentazione deve rispettare le caratteristiche di correttezza ed efficienza

# Esempio: la rappresentazione del tipo astratto per gli insiemi . . .

- Tipo astratto Insieme = <S, F, C> con
  - S = {Ins, V, boolean}
    - Ins = dominio d'interesse
    - V= dominio di sostegno (elementi degli insiemi)
  - F = {Vuoto, Inserisci, Cancella, Contiene}
  - C = {Insieme_Vuoto}
- Rappresentazione: <T', rapp, OP>
  - T' = <S', F', C'> è il tipo mediante il quale si rappresenta il dominio d'interesse e tutti gli altri domini in S
- Consideriamo il caso di insiemi che sono sottoinsiemi dell'insieme {1,2,3,4,5}

# . . . Esempio: la rappresentazione del tipo astratto per gli insiemi

- S' = { `boolean[5], int, boolean` }
  - La rappresentazione di Ins è un array di 5 componenti booleane: `boolean[5]`
  - La rappresentazione del dominio di sostegno è un sottoinsieme di `int` = {1,2,3,4,5}
  - boolean è rappresentato con il tipo `boolean`
- F' = tutte le operazioni definite per gli elementi di S' `&&, ||, !, [], +, -, *, /, %, ...`
- C' = tutte le costanti definite per gli elementi di S' `true, false, 0,1,2,3,4,5 []`

# Correttezza di una rappresentazione . . .

- **Una rappresentazione <T', *rapp*, OP> di un tipo T è corretta se:**
  - **Per ogni valore di tipo T esiste almeno un valore di tipo T' che lo rappresenta e, viceversa, per ogni valore di tipo T' è possibile risalire univocamente al corrispondente valore del tipo T, cioè**

    **per ogni elemento d di D (dominio di interesse del tipo T) esiste almeno un elemento d' di D' (dominio di interesse del tipo T') tale che d' = *rapp*(d) e da un elemento d' di D' è possibile ottenere un elemento d di D tale che d' = *rapp*(d)**
    - **non sempre *Rapp* è invertibile cioè, non sempre,**

      $$d = rapp^{-1}(d') = rapp^{-1}(rapp(d))$$

# Esempio: la rappresentazione del tipo di dato astratto per gli insiemi è corretta . . .

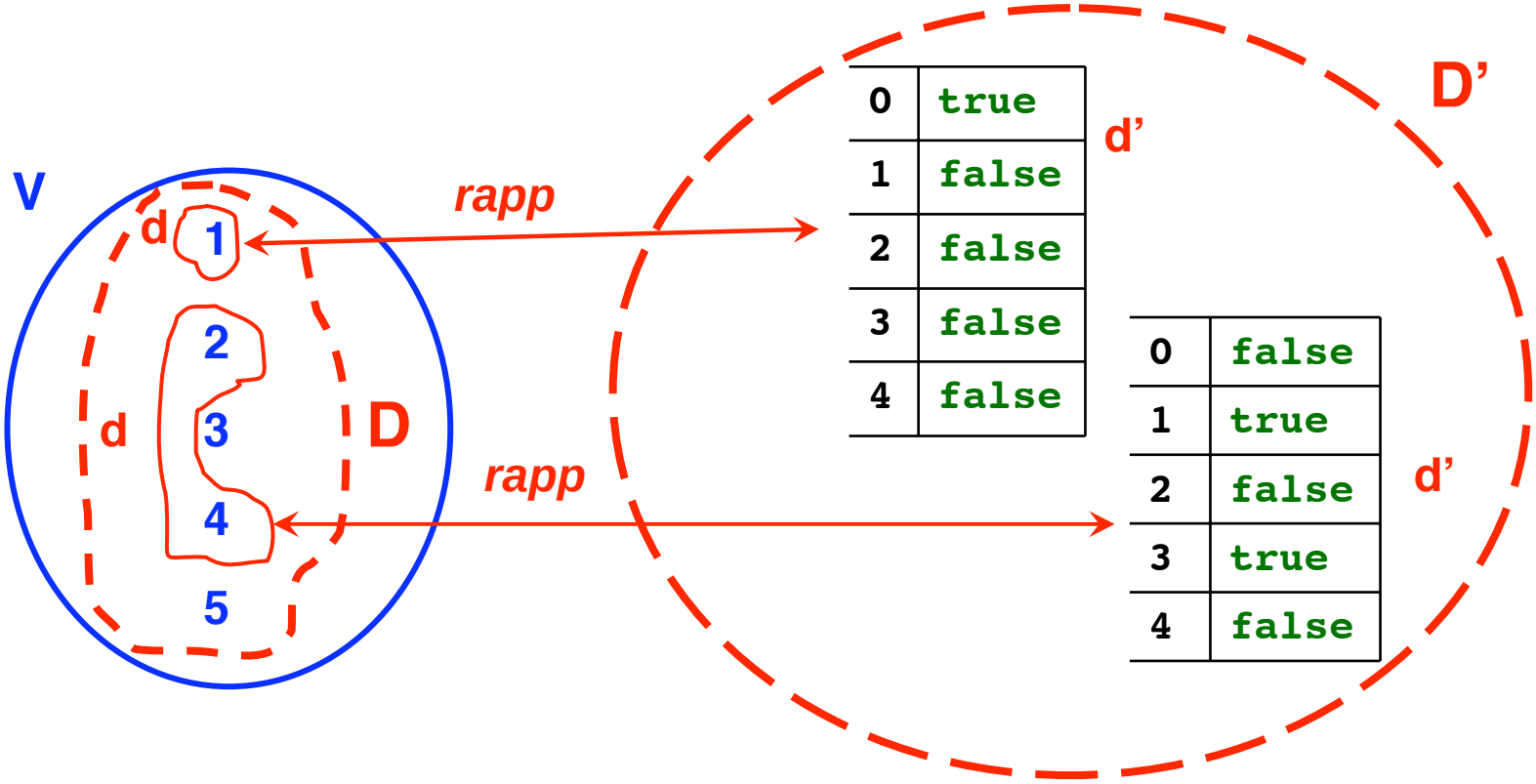
- D = Ins (dominio d'interesse del tipo T)
  - Un elemento d di D è un elemento dell'insieme delle parti dei primi 5 numeri naturali
    - {}, {3}, … {2,4}, …, {1,2,3,4,5}
- D' = `boolean[5]`
  - Un elemento d' di D' è un array di boolean
    - `{false, false, false, false, false}`
    - `{true, false, false, false, false}`
    - `{true, true, false, true, false}`
    - `{true, true, true, true, true}`
- rapp: D → D'
  - Esempio:

    rapp ({2,4}) = `{false, true, false, true, false}`

# . . . Esempio: la rappresentazione del tipo di dato astratto per gli insiemi è corretta

- Per ogni elemento d di D (= Ins) esiste almeno un elemento d' di D' tale che

$$d' = rapp(d) \quad \text{e} \quad d = rapp^{-1}(d') = rapp^{-1}(rapp(d))$$



# . . . Correttezza di una rappresentazione

- **E se:**
  - Ogni operazione primitiva f di T è correttamente realizzata dalla corrispondente operazione f' = OP(f) definita per T', cioè

    se $f: D_1 \times D_2 \times \ldots \times D_n \longrightarrow D$

    e $f': D'_1 \times D'_2 \times \ldots \times D'_n \longrightarrow D'$

    allora f' = OP (f) realizza correttamente f se per ogni $d_1 \epsilon D_1, \ldots, d_n \epsilon D_n$ e $d' \epsilon D'$ si ha

    $d' = f'( \mathit{rapp}(d_1), \ldots, \mathit{rapp}(d_n) ) = \mathit{rapp}(f(d_1, \ldots, d_n))$

# ... Esempio: la rappresentazione del tipo di dato astratto per gli insiemi è corretta...

- Le operazioni primitive di T
  - F = {Vuoto, Inserisci, Cancella, Contiene}  (f ϵ F)
- sono correttamente realizzate dalle corrispondenti operazioni
  - f' = OP(f) definita per T'  (f' ϵ F')
  - $D'_1$ = `boolean[]`, $D'_2$ = `int`, $D'_3$ = `boolean`
- Corrispondenza delle operazioni:
  - Vuoto: Ins → boolean

    ```
    boolean isVuoto(boolean[] ins)
    ```
  - Inserisci: Ins x V → Ins

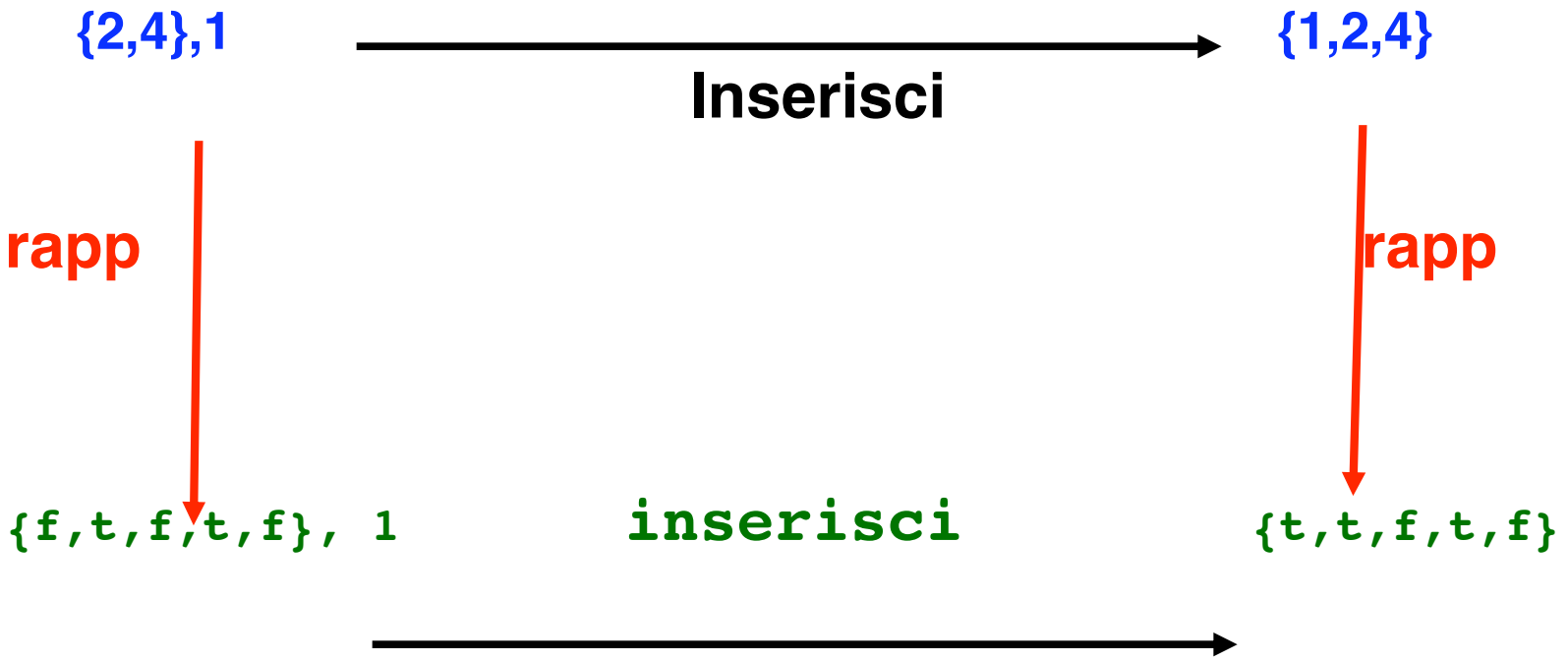
    ```
    boolean[] inserisci(boolean[] ins, int el)
    ```
  - Cancella: Ins x V → Ins

    ```
    boolean[] cancella(boolean[] ins, int el)
    ```
  - Contiene: Ins x V → boolean

    ```
    boolean contiene(boolean[], int el)
    ```

# ... Esempio: la rappresentazione del tipo di dato astratto per gli insiemi è corretta

`{f,t,f,t,f}` = rapp({2,4}) , `1` = rapp(1) , `{t,t,f,t,f}` = rapp({1,2,4})




`{t,t,f,t,f}` =

`inserisci`(rapp({2,4}), rapp(1)) = rapp (Inserisci({2,4},1)

# Una considerazione fondamentale

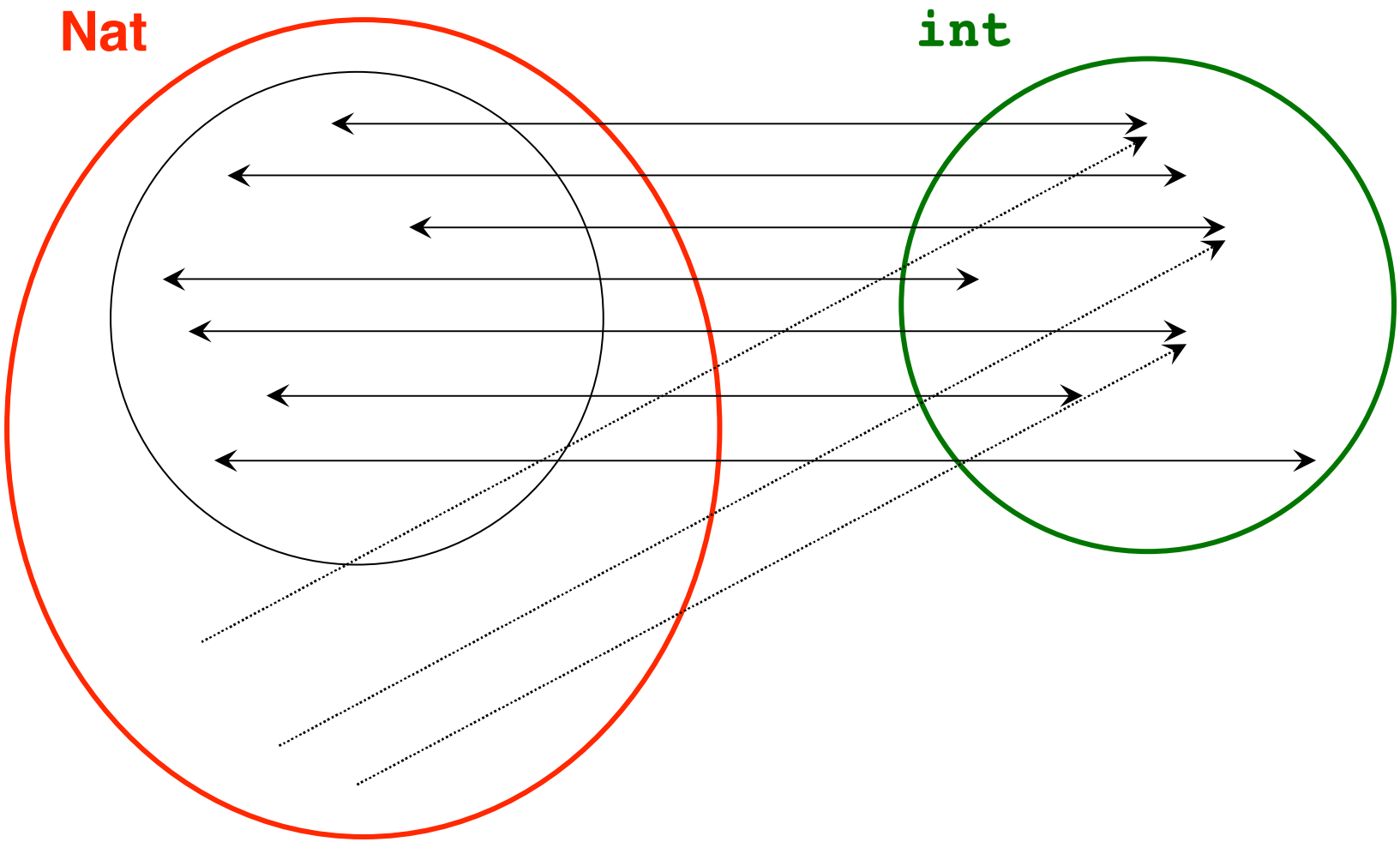
- Nella rappresentazione con array per il tipo astratto Insieme sono stati definiti metodi per le operazioni primitive, (ad esempio Inserisci) per le quali si può verificare la correttezza della rappresentazione
- Per questa verifica dobbiamo assumere che la corrispondenza *rapp* sia biunivoca sui domini di T
- Ciò non è sempre vero, specie nel caso di rappresentazione con tipi concreti
  - Si pensi al caso degli interi e del tipo concreto `int` di Java

# Esempio: gli interi rappresentati mediante `int`

**Nat = <S, F, C> < N , + , - , * , = , < , 0 , 1 >**

- **Rappresentazione di Nat: <T', *rapp*, OP>**
  - **T' = `int`**
  - **OP = { + , - , * , = , < } → { `+`, `-`, `*`, `=`, `<` }**
- **Esempio:**
  - **Sia f = + e f' = OP(+) = +**
  - **Siano $d_1$, $d_2$ ϵ Nat: $d_1 = 2^{31}-1$, $d_2 = 1$**
    **f'(rapp($d_1$), rapp($d_2$)) = f'($2^{31}-1$, 1) = $2^{31}-1+1 = -2^{31}$**
  - **rapp(f($d_1$,$d_2$)) = rapp($2^{31}-1 + 1$) = rapp($2^{31}$) = $-2^{31}$**
- **Ma la corrispondenza rapp: Nat → `int`**
  **non è biunivoca**

# La corrispondenza tra Nat e int non è biunivoca



# Esercizio

- Definire la rappresentazione del tipo di dato astratto  per gli insiemi, mediante una lista collegata

# Realizzazione di tipi astratti in Java . . .

- I passi fondamentali per la realizzazione di tipi astratti mediante classi Java sono i seguenti :
- Ad ogni tipo astratto viene associata una classe Java
- Va scelta la maniera con cui vengono rappresentati i valori del tipo astratto, definendo come campi privati le strutture necessarie
- I servizi che la classe offre ai moduli clienti vanno dichiarati come campi pubblici

# . . . Realizzazione di tipi astratti in Java

- Viene scelto lo schema realizzativo della classe
  - realizzazione delle operazioni del tipo astratto (side-effect o funzionale)
  - gestione della memoria utilizzata per la rappresentazione dei valori del tipo
- Viene decisa la maniera di tradurre l'intestazione delle funzioni del tipo astratto nell'intestazione dei corrispondenti metodi di classe, nonchè di quali metodi eventualmente ereditati fare overriding (è di particolare interesse il metodo equals)
- Vengono definiti i metodi della classe

# I tipi astratti Matrice e Vettore

- Le matrici sono tipi di dato i cui valori rappresentano una corrispondenza tra un insieme di dati detti indici, ed un insieme di valori di un tipo V (insieme di sostegno)

- Il valore di tipo V che è in corrispondenza con il valore i dell'indice in una matrice M, viene detto l'elemento (o la componente) di M di indice i

# Il tipo di dato astratto Matrice . . .

- Il tipo astratto matrice è definito da:
- S = {mat, indice, V}, dove mat è il dominio di interesse, e indice e V sono insiemi di valori di tipo qualunque, con indice di cardinalità numerabile
- F = {accedi, memorizza }, dove:
  - *accedi:* *mat* X *indice* → V
  - *memorizza:* *mat* X *indice* X V → *mat*
- C=∅

# . . . Il tipo di dato astratto Matrice

- In generale, indice è il prodotto cartesiano di n insiemi: indice = $ind_1$ X $ind_2$ X ... X $ind_n$

  cioè i suoi valori sono n-ple della forma

  $< i_1, i_2, ... i_n >$

  dove ogni $i_k$ appartiene all'insieme $ind_k$
- Il valore n stabilisce la dimensione della matrice
- Per n = 1 si ha il caso dei vettori

# Le operazioni accedi e memorizza

- **L'operazione accedi ha il seguente significato:**
  - se M è un valore di tipo matrice ed I è un valore di tipo indice, accedi(M,I) restituisce il valore dell'elemento di M che è in corrispondenza con il valore I dell'indice.
    - Si noti che accedi(M,I) viene spesso scritto come M[I] o $M_I$
- **L'operazione memorizza ha il seguente significato:**
  - memorizza (M,I,VAL) restituisce il valore M', di tipo matrice, in cui gli elementi di indice diverso da I sono uguali a quelli di M con lo stesso indice, mentre l'elemento di indice I è uguale a VAL
    - Restituisce una nuova matrice in cui è stato memorizzato in M[I] l'elemento con valore VAL

# Il tipo Matrice in Java . . .

- Il tipo Matrice può essere rappresentato in Java mediante il tipo concreto array, creando una corrispondenza biunivoca tra gli elementi di una matrice e gli elementi di un array
  - Al fine di realizzare le operazioni astratte accedi e memorizza, Java offre un semplice meccanismo che consente di riferirsi alla singola componente dell'array
  - Se A è un array, e i è un valore per il suo indice, il termine A[i] è la componente di A di indice i

# . . . Il tipo Matrice in Java

- Esempio: se A è di tipo matrice di interi a due dimensioni, per effettuare l'operazione astratta memorizza (A, <1,3>, 120), si eseguirà la seguente istruzione di assegnazione:

```
A[1][3] = 120;
```

- mentre per accedere alla componente di indice <3,5>, ad esempio per estrarne il valore, si scriverà `A[3][5]` come in

```
System.out.println(A[3][5])
```

# Una diversa rappresentazione . . .

- Ci sono dei casi, però, in cui le caratteristiche dei valori di tipo matrice che si utilizzano in un programma consentono una memorizzazione più efficiente rispetto a quella appena descritta
  - Ad esempio, per una matrice di interi in cui sia noto che la grande maggioranza degli elementi è uguale a zero, si potrebbero memorizzare i soli elementi diversi da zero
- Si usa il termine matrice sparsa . . .

# . . . Una diversa rappresentazione

- Si usa il termine matrice sparsa per indicare una matrice in cui la gran parte degli elementi ha un valore prefissato (detto valore dominante)
- Per tali matrici si possono utilizzare rappresentazioni ad hoc, dette rappresentazioni compatte
- L'idea fondamentale di tali rappresentazioni è quella di memorizzare solo gli elementi con valore diverso dal valore dominante

# Esempio di matrice sparsa

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 8 | 0 | 0 | 11 | 0 | 0 |
| 1 | 0 | 0 | 21 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 15 | 0 | 0 |
| 3 | 0 | 0 | 3 | 16 | 0 | 0 |

- Elemento dominante = 0
- Ogni elemento con valore diverso dal dominante viene rappresentato da una terna
  - Indice di riga
  - Indice di colonna
  - Valore dell'elemento