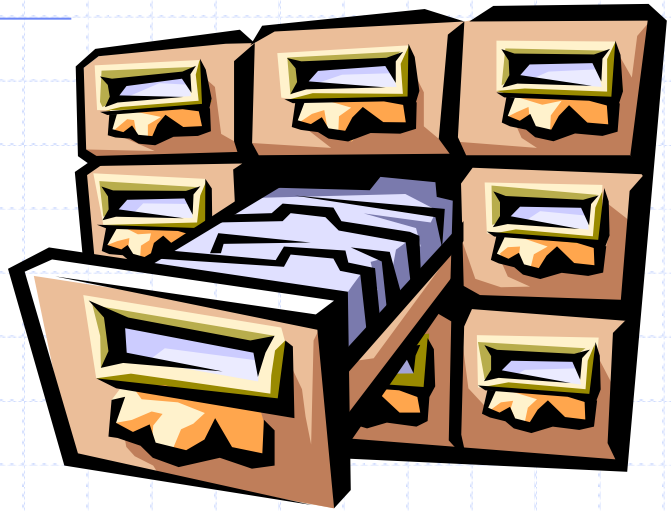
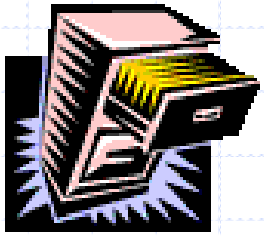
Mappe



Mappe

- Le mappe modellano collezioni di elementi (entry) chiave-valore sulle quali è possibile effettuare ricerche
- Le principali operazioni svolte mediante una mappa sono l'inserimento, la ricerca e la rimozione di elementi
- **Non** sono ammessi entry multipli aventi la stessa chiave
- Applicazioni:
  - rubrica
  - database studenti

Il TDA mappa (9.1)



- Metodi del TDA mappa:
  - get(k): se la mappa M ha una entry con chiave k, restituisce il valore ad essa associato, altrimenti restituisce null
  - put(k, v): inserisce nella mappa M la entry (k, v); se la chiave k non è già in M restituisce il risultato null; altrimenti, se w è il valore già presente associato a k, sostituisce w con v, restituendo come risultato proprio w
  - remove(k): se la mappa M ha una entry con chiave k, rimuove tale entry da M e restituisce il valore associato a k; altrimenti, restituisce null
  - size(), isEmpty()
  - keys(): restituisce una collection iterabile contenente tutte le chiavi contenute in M (keys().iterator() restituisce un iteratore alle chiavi)
  - values(): restituisce una collection iterabile contenente tutti i valori associati alle chiavi contenute in M (values().iterator() restituisce un iteratore ai valori)
  - entries(): restituisce una collection iterabile contenente tutte le entry chiave-valore contenute in M (entries().iterator() restituisce un iteratore alle entry)

Esempio

Operazione	*Output*	*Mappa*
isEmpty()	**true**	Ø
put(5,*A*)	**null**	(5,*A*)
put(7,*B*)	**null**	(5,*A*),(7,*B*)
put(2,*C*)	**null**	(5,*A*),(7,*B*),(2,*C*)
put(8,*D*)	**null**	(5,*A*),(7,*B*),(2,*C*),(8,*D*)
put(2,*E*)	*C*	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
get(7)	*B*	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
get(4)	**null**	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
get(2)	*E*	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
size()	4	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
remove(5)	*A*	(7,*B*),(2,*E*),(8,*D*)
remove(2)	*E*	(7,*B*),(8,*D*)
get(2)	**null**	(7,*B*),(8,*D*)
isEmpty()	**false**	(7,*B*),(8,*D*)

Confronto con java.util.Map

Metodi del TDA Mappa

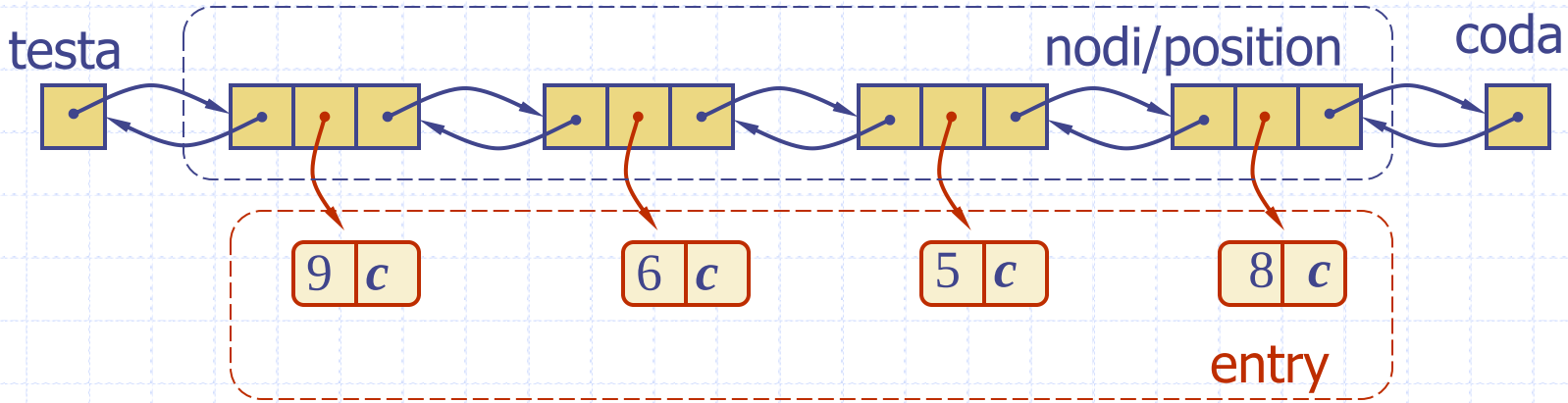
- size()
- isEmpty()
- get(*k*)
- put(*k*,*v*)
- remove(*k*)
- keys()
- values()

Metodi di java.util.Map

- size()
- isEmpty()
- get(*k*)
- put(*k*,*v*)
- remove(*k*)
- *keySet().iterator()*
- *values().iterator()*

Mappa basate su semplici liste

- Possiamo facilmente implementare una mappa usando una lista non ordinata
  - Memorizziamo gli elementi della mappa in una lista S (doppiamente collegata), in ordine arbitrario



Algoritmo get(k)

Algorithm get(*k*):
    B = *S*.positions() {*B* è un iteratore delle position in *S*}
    while *B*.hasNext() **do**
        p = *B*.next() { prossima position in *B* }
        if *p*.element().key() = *k* **then**
            return *p*.element().value()
    return null { non esistono entry con chiave *k*}

Algoritmo put(k,v)

Algorithm put(*k*,*v*):
B = *S*.positions()
while *B*.hasNext() **do**
    p = *B*.next()
    if *p*.element().key() = *k* **then**
        t = *p*.element().value()
        B.replace(*p*,(*k*,*v*))
        return *t* { restituisce vecchio valore }
S.insertLast((*k*,*v*))
n = *n* + 1 { incrementa variabile contenente numero di entry }
return null { non c'era una entry precedente con chiave *k*}

Algoritmo remove(k)

```
Algorithm remove(k):
B = S.positions()
while B.hasNext() do
    p = B.next()
    if p.element().key() = k then
        t = p.element().value()
        S.remove(p)
        n = n − 1        { decrementa il numero di entry }
        return t         { restituisce il valore eliminato }
return null              { non esiste entry con chiave k}
```

Prestazioni di mappe basate su liste

- Prestazioni:
  - put viene eseguito in tempo $O(1)$ poiché possiamo inserire la nuova entry all'inizio o alla fine della sequenza
  - get e remove vengono eseguite in tempo $O(n)$ poiché, nel caso peggiore (chiave non trovata), dobbiamo scandire l'intera sequenza durante la ricerca di chiave
- L'implementazione tramite liste non ordinate è efficace solo per mappe di dimensione limitata o per mappe dove l'operazione predominante è put, mentre ricerche ed eliminazioni sono eseguite raramente (es.: registrazione dei login utente in una workstation)