



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica



Laboratorio di Algoritmi e Strutture Dati a.a. 2023/2024

CONFRONTO TRA OGGETTI:
Le interfacce Comparable e Comparator

Giovanna Melideo
Università degli Studi dell'Aquila
DISIM

Richiami: La libreria standard

- Java possiede un'enorme e lussuosa libreria di classi e interfacce standard, che costituisce uno dei punti di forza del linguaggio.
- Essa è organizzata in vari **package** e **subpackage** che raccolgono classi e interfacce secondo un'organizzazione basata sul campo d'utilizzo.
- Una classe predefinita può essere usata ponendo una dichiarazione **import** all'inizio del codice sorgente:

```
import nome_classe.*;
```

- Una volta importata, i metodi possono essere usati con la consueta sintassi:

```
nome_classe.metodo;
```

I principali package

- **java.io** contiene classi per realizzare l'input – output in Java
- **java.awt** contiene classi per realizzare interfacce grafiche, come `Button`
- **java.net** contiene classi per realizzare connessioni, come `Socket`
- **java.applet** contiene un'unica classe: `Applet`. Questa permette di realizzare applet
- **java.util** raccoglie classi d'utilità, come `Date`
- **java.lang** è il package che contiene le classi nucleo del linguaggio, come `Object`, `System`, `String`, l'interfaccia **Comparable**, ecc.
 - **È importato automaticamente in ogni file Java**

L'interfaccia `java.lang.Comparable`

- L'interfaccia `Comparable` impone un **criterio di ordinamento** sugli oggetti della classe che la implementa (**ordinamento naturale** della classe).
- `Comparable` contiene il solo metodo di confronto naturale:

```
public interface Comparable<T> {  
    int compareTo(T obj);  
}
```
- Il metodo `compareTo` confronta l'oggetto corrente `this` con l'oggetto specificato `obj`

Il metodo compareTo()

`x.compareTo(y)` restituisce:

- un valore negativo se **`x`** è minore di **`y`**
 - ovvero **`x`** «precede» **`y`** nella sequenza ordinata
- 0 se **`x`** è uguale a **`y`**
- un valore positivo se **`x`** è maggiore di **`y`**
 - ovvero **`x`** «segue» **`y`** nella sequenza ordinata

Il metodo `compareTo()`: le eccezioni

- `x.compareTo(y)` dovrebbe lanciare l'eccezione `ClassCastException` se riceve un oggetto `y` che non è confrontabile con `x`, a causa del suo tipo effettivo
- Nota che `null` non è istanza di nessuna classe, e `x.compareTo(null)` dovrebbe sollevare l'eccezione `NullPointerException` anche se `e.equals(null)` returns `false`.

Consistenza tra compareTo() e equals()

- L'implementazione di Comparable è **consistente o compatibile con** equals() se dati **x** e **y**:
$$x.compareTo(y) == 0 \text{ se e solo se } x.equals(y) == true$$
- È fortemente raccomandato che compareTo() sia consistente (o compatibile) con equals()
- Ogni classe che implementa l'interfaccia Comparable e viola questa condizione dovrebbe dichiararlo esplicitamente
 - *"Note: this class has a natural ordering that is inconsistent with equals."*

L'interfaccia Comparable: esempi di classi

Classi di Java che implementano Comparable:

- `String`
- `File`
- `Date`
- `Byte`, `Character`, `Short`, `Integer`, `Long`, `Float`, `Double` (classi wrapper)

L'interfaccia `java.util.Comparator`

- In alternativa all'interfaccia `Comparable`, si può realizzare una seconda classe che implementa l'interfaccia `Comparator`

```
public interface Comparator<T> {  
    public int compare(T x, T y);  
}
```

L'interfaccia Comparator (1 di 2)

compare(x, y) restituisce:

- un valore negativo se x è minore di y
 - ovvero x «precede» y nella sequenza ordinata
- 0 se x è uguale a y
- un valore positivo se x è maggiore di y
 - ovvero x «segue» y nella sequenza ordinata

L'interfaccia `Comparator` (2 di 2)

L'uso di `Comparator` è indicato quando:

- la classe da ordinare non ha un unico criterio di ordinamento naturale, oppure
- la classe da ordinare è già stata realizzata e non si può o non si vuole modificarla (ad es. `String`...)
- L'interfaccia `Comparable` è definita nel package `java.lang` mentre `Comparator` in `java.util`:
 - sottolinea che il metodo `compareTo` dovrebbe essere fornita di default, mentre `compare` come un'utility

Uso di comparatori

- Nell'API Java sono presenti metodi che utilizzano le interfacce `Comparable` e `Comparator` per fornire algoritmi di ordinamento di array e di liste
- Per quanto riguarda gli array, tali metodi si trovano nella classe `java.util.Arrays`
- Per quanto riguarda le liste, tali metodi si trovano nella classe `java.util.Collections`
- Le classi `Arrays` e `Collections` contengono solo metodi statici

Arrays.sort

```
static void sort(Object[] a)
```

- ordina l'array `a` in senso non-decrescente, in base all'ordinamento naturale tra i suoi elementi
- ovvero, suppone che tutti gli elementi contenuti siano confrontabili tra loro tramite l'interfaccia `Comparable`

```
static <T> void sort(T[] a, Comparator<? super T> c)
```

- ordina l'array `a` in senso non-decrescente, in base all'ordinamento indotto dal comparatore `c`
- L'algoritmo usato è una versione ottimizzata del quicksort

Collections.sort

```
static <T extends Comparable<? super T>> void  
    sort(List<T> list)
```

- ordina la lista `list` in senso non-decrescente, in base all'ordinamento naturale tra i suoi elementi

```
static <T> void sort(List<T> list, Comparator<? super T> c)
```

- ordina la lista `list` in senso non-decrescente, in base all'ordinamento indotto dal comparatore `c`
- L'algoritmo usato è una versione ottimizzata del mergesort

Ordinamento

- In entrambi i casi, l'ordinamento è in-place e stabile:
 - l'array viene modificato senza utilizzare strutture di appoggio (in-place)
 - gli elementi equivalenti secondo l'ordinamento mantengono l'ordine che avevano originariamente (stabile)

Exercise

- Implement a `Student` class. Each student has a name and grade point average.
- 1. Declare an array whose elements come from the same `Student` class. Sort the students in alphabetical order. Next, sort the students in decreasing order of GPAs.
- 2. Declare a `LinkedList (ArrayList)` object, `list`, whose elements come from the same `Student` class. Sort the students in alphabetical order. Next, sort the students in decreasing order of GPAs.



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica



Domande?

Giovanna Melideo
Università degli Studi dell'Aquila
DISIM