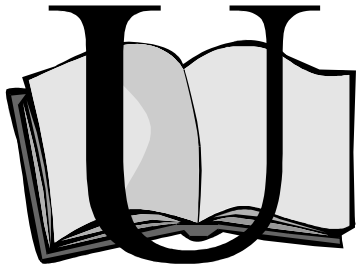
Interfacce Java



Interfacce

- Un **metodo astratto** è un metodo senza corpo, con un "**;**" dopo l'intestazione
- Una **interfaccia** (**interface**) in Java ha una struttura simile a una classe, ma può contenere SOLO **costanti** e **metodi d'istanza astratti** (quindi non può contenere né *costruttori*, né *variabili statiche*, né *variabili di istanza*, né *metodi statici*).

java.lang.Comparable

```
public interface Comparable {

        public int compareTo(Object o);

}
```

Spesso l'interfaccia comprende anche una descrizione informale del significato dei metodi (che chiameremo **specifica** o **contratto**).

Comparable

- Ad esempio, nella API di **Comparable** si vede che al metodo **compareTo** è associata una precisa interpretazione (descritta a parole): esso definisce un ordinamento totale sugli oggetti della classe. Estratto:
- **compareTo:** Compares **this** object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object. [...] The implementor must also ensure that the relation is transitive: **(x.compareTo(y)>0 && y.compareTo(z)>0)** implies **x.compareTo(z)>0**. [...] It is strongly recommended, but not strictly required that **(x.compareTo(y)==0) == (x.equals(y))**. [...]

Implementare un’interfaccia

- Si può dichiarare che una classe **implementa** (**implements**) una data interfaccia: in questo caso la classe deve fornire una **realizzazione** per tutti i metodi astratti dell'interfaccia, cioè la classe deve fornire metodi con la stessa firma descritta nell'interfaccia (e con il corpo, naturalmente). La realizzazione di un metodo deve anche rispettare la "specifica" del corrispondente metodo astratto.

Esempio: Interi

```
public class Intero implements Comparable {
        […]
        public int compareTo(Object o) {
        Intero int1 = (Intero) o;
        if (n < int1.n) return -1;
        else if (n > int1.n) return 1;
        else return 0;
        }
        […]
}
```

A cosa servono le interfacce

- per definire **Tipi di Dati Astratti** si pensi diverse possibili implementazioni di ADT (*liste con/senza ripetizioni*, *array*, *tabelle hash*, *alberi binari di ricerca*, ...);
- come **contratto** tra chi implementa una classe e chi la usa: le due parti possono essere sviluppate e compilate separatamente;
- per scrivere **programmi generici** (applicabili a più classi), evitando di duplicare il codice;
- Ad esempio, sfruttando l'interfaccia **Comparable**, si possono scrivere algoritmi di ordinamento generici, applicabili a istanze di qualunque classe che la implementi.

Regole di utilizzo

- Possiamo dichiarare una variabile indicando come tipo un'interfaccia: **Comparable cmp;**
- Non possiamo istanziare un'interfaccia: **Comparable com = new Comparable();**

  // **VIETATO**
- Ad una variabile di tipo interfaccia possiamo assegnare solo istanze di classi che implementano l'interfaccia:

  Comparable com = new Intero(5);

Su di una variabile di tipo interfaccia possiamo invocare solo metodi dichiarati nell'interfaccia (o nelle sue "super-interfacce").