

# Interfacce

Programmazione  
Corso di laurea in Informatica

## Interfacce

- Un' **interfaccia** Java è una collezione di metodi astratti (e di costanti)
- Un **metodo astratto** è un metodo non implementato
  - costituito dall'intestazione senza il corpo della definizione
- Un metodo astratto viene dichiarato mediante l'uso del modificatore **abstract**
  - ma poiché tutti i metodi di un'interfaccia sono necessariamente astratti spesso viene omesso
- L'interfaccia viene usata per definire formalmente l'insieme dei metodi che una classe deve implementare

AA2007/08 © M.A. Alberti 2 Programmazione Interfacce

## Interfacce

**interface** è una parola riservata

```
public interface Fattibile {
    public void faiQuesto();
    public int faiQuello();
    public void faiQuesto(float value, char ch);
    public boolean faiQuest'altro (int num);
}
```

Per nessuno dei metodi in un'interfaccia viene definito il corpo

L'intestazione di ciascun metodo termina con il ;

AA2007/08 © M.A. Alberti 3 Programmazione Interfacce

## Interfacce

- Un'interfaccia non può essere istanziata
- I metodi di un'interfaccia hanno visibilità **public** per default
- Le classi implementano un'interfaccia
  1. Affermandolo nella intestazione della classe
  2. Fornendo l'implementazione per ciascun metodo astratto dell'interfaccia
- Una classe che implementa un'interfaccia, deve definire **tutti** i metodi dell'interfaccia altrimenti il compilatore **segnala errore**

AA2007/08 © M.A. Alberti 4 Programmazione Interfacce

## Implementare un'interfaccia

```
public class SiDeveFare implements Fattibile {
    public void faiQuesto() {
        // codice
    }

    public int faiQuello() {
        // codice
    }

    // etc.
}
```

**implements** è una parola riservata

A ogni metodo dell'interfaccia **Fattibile** viene data la definizione appropriata

AA2007/08 © M.A. Alberti 5 Programmazione Interfacce

## Implementare un'interfaccia

- La classe deve implementare **tutti** i metodi dichiarati nell'interfaccia
- Una classe che implementa un'interfaccia può **anche definire altri metodi**
- Una classe può **implementare diverse interfacce**
  - Le diverse interfacce sono separate da virgole nella clausola di implementazione
- **Parlante.java** rappresenta l'interfaccia
- **Filosofo.java** realizza una interfaccia **Parlante**
- **Cane.java** realizza una interfaccia **Parlante**
- **Parlare.java** usa le classi **Cane** e **Filosofo**

AA2007/08 © M.A. Alberti 6 Programmazione Interfacce

### Polimorfismo con le interfacce

- Si ha **polimorfismo** quando un identificatore può riferirsi a oggetti di tipo differente in momenti diversi
  - Con le interfacce si creano riferimenti polimorfi
  - Il nome di un'interfaccia (es. **Fattibile**) può essere usato come tipo di una variabile di riferimento a un oggetto  
`Fattibile obj;`
  - Il riferimento **obj** può puntare un oggetto di una qualunque classe che implementi l'interfaccia **Fattibile**
  - Il riferimento è **polimorfo**, cioè può assumere diverse forme

AA2007/08  
© M.A. Alberti

7

Programmazione  
Interfacce

### Polimorfismo con le interfacce

- Il riferimento **polimorfo** viene risolto al tempo dell'esecuzione, cioè a **run time**
- Si attua un **legame dinamico**
  - Il metodo che viene invocato dipende dal tipo di oggetto a cui **obj** fa riferimento:  
`obj.faiQuesto();`
  - La stessa linea di codice può eseguire diversi metodi in momenti diversi se l'oggetto cui punta **obj** cambia
- L'uso di riferimenti polimorfi può portare a un disegno elegante e robusto del software
- [Parlare.java](#)

AA2007/08  
© M.A. Alberti

8

Programmazione  
Interfacce

### Alcune interfacce standard

- L'interfaccia **Comparable** contiene un metodo astratto chiamato **compareTo**, usato per confrontare oggetti
  - La classe **String** implementa l'interfaccia **Comparable** che consente di confrontare stringhe in ordine alfabetico mediante il metodo **compareTo** specificato ad hoc
  - `int compareTo(Object obj)`
- L'interfaccia **Iterator** indica i metodi da implementare per gestire una collezione di oggetti
  - Caso per caso si deve decidere l'ordine con cui gli oggetti della collezione devono essere restituiti dai metodi
  - `boolean hasNext()`
  - `Object next()`
  - `void remove()`

AA2007/08  
© M.A. Alberti

9

Programmazione  
Interfacce