# Interfacce

Un'interfaccia è una specie di classe completamente astratta, cioè del tutto priva della parte di implementazione (quasi... vedi sotto)

* Tutti i metodi sono astratti
* Non vi sono attributi
* È possibile definire solo "attributi" final (in pratica costanti)

## Definendo un attributo in un'interfaccia questo viene automaticamente considerato final

### Un'interfaccia

* Ha tutti i vantaggi e le indicazioni d'uso delle classi astratte
* Presenta maggior flessibilità rispetto all'ereditarietà di una classe astratta

Si può pensare a un'interfaccia come a una classe astratta che ha tutti e soli metodi astratti (ci sono però differenze).

Un'interfaccia può essere considerata un modo per cosa dovrebbero fare le classi senza specificare come farlo.

Quindi un'interfaccia non è una classe ma un insieme di requisiti per le classi che si vogliono conformare ad essa.

## Sintassi:

public interface NomeInterfaccia  
{  
 int metodo1( ... );  
}

## interface

Si utilizza la parola chiave interface anzichè class

I metodi sono implicitamente pubblici e astratti, non bisogna indicarlo

Se una classe decide di soddisfare i requisiti di un'interfaccia si dice che la classe implementa l'interfaccia.

## esempio d'uso

Per indicare che una classe implementa un'interfaccia si utilizza la seguente sintassi:

public class NomeClasse implements NomeInterfaccia  
{  
 //codice relativo alla classe  
}

Tale classe deve implementare tutti i metodi elencati nell'interfaccia.

### proprietà

Le interfacce non sono classi; non si può utilizzare new per crearne oggetti.

I metodi di un'interfaccia sono automaticamente public (quindi non è necessario scriverlo)

Gli attributi di un'interfaccia sono sempre public static final (non è necessario scriverlo).

Un'interfaccia con il nome NomeInterfaccia va salvata nel file NomeInterfaccia.java (come accade per le classi).

### ereditarietà multipla

Una sottoclasse può estendere solo 1 superclasse (non permettendo l'ereditarietà multipla)

Con le interfacce invece la situazione è diversa: una classe può implementare quante interfacce vuole.

### variabili

Posso dichiarare variabili del tipo dell'interfaccia e, sfruttando il polimorfismo, assegnargli oggetti di classi che implementano tali interfacce.

## esempio d'uso

public interface NomeInterfaccia {  
int metodo1( );  
}  
public class NomeClasse implements NomeInterfaccia{  
int metodo1( ) {  
//codice del metodo  
}  
}  
NomeInterfaccia a = new NomeClasse ( );

## Tipi di interfacce

* Normali
* Single Abstract method - @FunctionalInterface
* Marker

Con java 1.8 le interfacce sono state modificate: è possibile implementare due tipi di metodi (!!!)

* default
* static

## Lambda expressions

Con le interfacce contenenti un singolo metodo astratto, è possibile utilizzare le espressioni lambda

Integer raddoppiato = (o) -> o \* 2 ;