Interfacce

Le interfacce in Java sono uno strumento potente per definire contratti o specifiche di comportamento che le classi devono implementare. Ecco alcuni concetti chiave sulle interfacce in Java:

1. **Definizione di un'Interfaccia:**
    - Un'interfaccia è definita utilizzando la parola chiave `interface`. Essa contiene dichiarazioni di metodi, ma non fornisce l'implementazione dei metodi stessi.

```
public interface Contratto {
    void obbligoUno();
    int obbligoDue(String parametro);
}
```

2. **Implementazione di un'Interfaccia:**
   - Una classe implementa un'interfaccia utilizzando la parola chiave `implements`. La classe deve fornire un'implementazione per tutti i metodi dichiarati nell'interfaccia.

```
public class MiaClasse implements Contratto {
    @Override
    public void obbligoUno() {
        // Implementazione del obbligoUno
    }

    @Override
    public int obbligoDue(String parametro) {
        // Implementazione del obbligoDue
        return 42;
    }
}
```

3. **Estensione di Interfacce:**
    - Un'interfaccia può estendere una o più interfacce. Questo consente di creare gerarchie di interfacce, ereditando i metodi delle interfacce padre.

```
public interface AltraInterfaccia extends Contratto {
    void nuovoMetodo();
}
```

4. **Variabili in un'Interfaccia:**
   - In Java 8 e versioni successive, è possibile dichiarare variabili di istanza in un'interfaccia. Queste variabili sono implicitamente pubbliche, statiche e finali.

```
public interface InterfacciaConVariabile {
    int VALORE_COSTANTE = 42;
}
```

5. **Default Methods:**
    - Dal Java 8, è possibile fornire un'implementazione di default per i metodi in un'interfaccia. Questo consente di estendere le interfacce senza dover modificare tutte le classi che le implementano.

```
public interface InterfacciaConDefault {
    void metodo();

    default void metodoDefault() {
        // Implementazione di default
    }
}
```

6. **Metodi Statici in un'Interfaccia:**
    - Si possono definire metodi statici all'interno di un'interfaccia, che possono essere chiamati senza creare un'istanza dell'interfaccia.

```
public interface InterfacciaConMetodoStatico {
    static void metodoStatico() {
        // Implementazione del metodo statico
    }
}
```

7. **Utilità delle Interfacce:**
    - Le interfacce consentono la progettazione di codice più flessibile e modulare. Possono essere utilizzate per definire contratti comuni, permettendo l'implementazione polimorfica e facilitando la manutenzione del codice.

Le interfacce svolgono un ruolo fondamentale nella progettazione orientata agli oggetti, consentendo una separazione chiara tra contratto e implementazione. La loro utilità diventa evidente quando si desidera che diverse classi condividano un comportamento comune senza ereditarietà multipla diretta, poiché Java non supporta la multi-ereditarietà di classi.

Un'interfaccia

- Ha tutti i vantaggi e le indicazioni d'uso delle classi astratte
- Presenta maggior flessibilità rispetto all'ereditarietà di una classe astratta
- È possibile definire solo "attributi" final (in pratica costanti)
- Definendo un attributo in un'interfaccia questo viene automaticamente considerato final

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.

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.

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 una superclasse (non è permessa in Java l'ereditarietà multipla)

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

Estendere interfacce

Un'interfaccia può estendere un'altra interfaccia, sempre con ereditarietà singola. Possono così essere implementate gerarchie di interfacce (e classi astratte, vedi per esempio JCF)

Sintassi di esempio

```
public interface NomeInterfaccia extends AltraInterfaccia {
  int metodo1( );
}
```

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

Interfacce funzionali e Lambda expressions

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

- Ad esempio: `Integer raddoppiato = (o) -> o * 2 ;`
- Interfacce funzionali

esempi classi