SCHILLACI GABRIEL 4^C 15/11/2019

Ereditarietà

Meccanismo per definire una nuova classe (classe derivata) come specializzazione di un'altra (classe base)
– La classe **base** modella un concetto **generico**
– La classe **derivata** modella un concetto **più specifico**
• La classe derivata:
– Dispone di tutte le funzionalità (attributi e metodi) di quella base
– Può aggiungere funzionalità proprie
– Può ridefinirne il funzionamento di metodi esistenti (polimorfismo)

Come l'ho usata?

Io personalmente ho utilizzato l'ereditarietà per questi motivi, molto utili e vantaggiosi:
• Evitare la duplicazione di codice
• Permettere il riuso di funzionalità
• Semplificare la costruzione di nuove classi
• Facilitare la manutenzione
Inoltre si definisce una classe derivata attraverso la parola chiave "**extends**".
Ovvero:
– Seguita dal nome della classe base
(Gli oggetti della classe derivata sono, a tutti gli effetti, estensioni della classe base)
– Anche nella loro rappresentazione in memoria

Polimorfismo

• Java mantiene traccia della classe effettiva di un dato oggetto
– Seleziona sempre il metodo più specifico…
–Anche se la variabile che lo contiene appartiene ad una classe più generica!
• Una variabile generica può avere "molte forme"
– Contenere oggetti di **sottoclassi differenti**
– In caso di ridefinizione, il metodo chiamato dipende dal **tipo effettivo** dell'oggetto
• Per sfruttare questa tecnica:
– Si definiscono, nella super-classe, metodi con implementazione generica…
– …sostituiti, nelle sottoclassi, da implementazioni specifiche
– Si utilizzano variabili aventi come tipo quello della super-classe

• Meccanismo estremamente potente e versatile, alla base di molti "pattern" di programmazione

Utilizzo del "this" in java

Il **riferimento this** in Java viene utilizzato per fare riferimento, all'interno di un metodo o di un costruttore, agli attributi o metodi locali. Questo tipo di riferimento non fa altro che puntare all'oggetto a cui appartiene risolvendo possibili problemi di ambiguità. Vediamo subito un esempio:

```
public ContoEsteso(String numConto) {
        super(numConto);
        this.fido = 1000;
    }
```