Programmazione in Java
Parte I: Fondamenti

Lezione 4

Dott. Marco Faella

La classe Object

- Se una classe non estende nessuna altra classe, allora estende implicitamente la classe Object
- Di conseguenza, ogni classe estende, direttamente o indirettamente, Object

- Quindi, un riferimento di tipo (dichiarato) Object può puntare a qualunque oggetto
  - compreso un array!
  - in questo, è simile al puntatore universale (void *) del C

- Inoltre, tutti gli oggetti dispongono dei metodi (pubblici) di Object
- Ad esempio, Object contiene il metodo:

```
public String toString()
```

- che restituisce una stringa che descrive l'oggetto in questione

- Questo metodo viene chiamato automaticamente quando si passa un oggetto a System.out.println
- Esercizio: verificare cosa restituisce questo metodo per un oggetto qualunque
- E' opportuno ridefinire questo metodo in tutte le nostre classi

Il modificatore **final**

- Il modificatore final si può applicare a variabili locali (e parametri formali), attributi, metodi e classi
- Nel caso di variabili (locali o attributi), final indica che tale variabile è **costante**, cioè che non può essere modificata dopo la sua inizializzazione
- E' buona norma dichiarare final tutte le variabili che sono usate in sola lettura
  - in tal modo, il compilatore può segnalarci eventuali violazioni
- La sintassi è la seguente:

```
public void f(final int a, int b) {
        final Data natale = new Data(25, 12, 1995);
        ...
}
```

- Il parametro *a* e il riferimento *natale* non possono essere modificati
  - Cioè, *natale* punterà sempre allo **stesso oggetto** Data
  - Ciò nonostante, se la classe Data prevede metodi modificatori, è possibile modificare il **contenuto** dell'oggetto puntato

Il modificatore **final**

- Applicato ad un metodo, il modificatore final indica che quel metodo non può essere ridefinito tramite overriding
- In tal caso, possiamo dire che quella è la versione definitiva del metodo
- Il tentativo, da parte di una sottoclasse, di ridefinire quel metodo porterà ad un errore di compilazione per la sottoclasse

- E' opportuno dichiarare un metodo final solamente se abbiamo la certezza che la sua implementazione sarà valida per tutte le possibili sottoclassi

- La sintassi è la seguente:

```
public final void f(...) {
        ...
}
```

Il modificatore **final**

- Applicato ad una classe, il modificatore final indica che quella classe non può essere estesa tramite ereditarietà
- Ovvero, quella classe non ammette sottoclassi
- Il tentativo, da parte di un'altra classe, di estendere la prima porterà ad un errore di compilazione

- E' raro dichiarare una classe final: può capitare in contesti in cui ci siano vincoli di sicurezza, ed estendere una certa classe potrebbe indebolire le garanzie di sicurezza offerte dalla stessa

- Un esempio di classe final è dato da String
- I progettisti di Java hanno ritenuto che String fosse una classe troppo fondamentale per permettere ai programmatori di costruirne versioni personalizzate
    - in particolare, una eventuale sottoclasse di String avrebbe potuto violare l'assunzione di immutabilità degli oggetti stringa

- La sintassi è la seguente:

```
final class A {
        ...
}
```

Metodi astratti e classi astratte

- Il modificatore abstract permette di dichiarare un metodo senza fornire alcun corpo
- Questo serve a fornire una versione generica di un metodo, che le sottoclassi dovranno ridefinire

- Se una classe contiene almeno un metodo abstract, essa stessa deve essere dichiarata abstract
  - una classe può essere dichiarata abstract anche se non contiene metodi abstract

- Una classe abstract non può essere istanziata, anche se dotata di costruttori
  - gli eventuali costruttori possono comunque servire alle sottoclassi, che li possono invocare tramite super
- Per il resto, una classe abstract non ha altre restrizioni: può contenere attributi e metodi concreti, come qualsiasi altra classe