Idiomatica: codice orribile, antipattern e pratiche discutibili

In questa rubrica analizzeremo i più comuni errori di codifica, di progettazione ed i tranelli in cui è facile cadere. Programmatore avvisato, mezzo salvato!

La massima di questo mese:

> ""My guess is that object-oriented programming will be in the 1980s what structured programming was in the 1970s. Everyone will be in favor of it. Every manufacturer will promote his products as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And noone will know just what it is."
>
> *T.Rentsch*

#1: Il polimorfismo dei poveri

Voglio cominciare questa rubrica con una "worst practice" fra le più orribili, il polimorfismo dei poveri. Non che non abbia materiale a disposizione per decine e decine di rubriche, e centinaia di esempi di come non si dovrebbe scrivere del codice in Java... ma questa particolare pratica, oltre a essere veramente brutta, ha delle caratteristiche che secondo me la rendono unica. E per questo deve avere l'onore del primo numero di questa rubrica dedicata alle brutture nel design.

- E' assolutamente non necessaria, e l'alternativa è nell'abc della programmazione ad oggetti
- Non offre nessun vantaggio particolare rispetto all'alternativa corretta, ma solo svantaggi. Altre brutture perlomeno hanno una qualche giustificazione più o meno fondata, questa no.
- E' incredibilmente diffusa

Il polimorfismo dei poveri è caratterizzato da codice simile a questo:

```
if (a instanceof b) {
        // fai qualcosa
} else if (a instanceof c) {
        // faiqualche altra cosa
}
```

Ora, gia' l'uso di instanceof di per se' è criticabile: instanceof in un linguaggio ad oggetti non dovrebbe mai essere usato, se non in un caso particolare che lascio come simpatico esercizio agli amici lettori.

Ma usare instanceof per fare eseguire logiche diverse ad un oggetto a seconda del suo tipo, è veramente abominevole: a cosa serve il polimorfismo – che fa esattamente la stessa cosa - se poi devo ricostruirmelo malamente da solo? Notare che il fatto che l'operazione sia eseguita sullo lo stesso oggetto su cui si è operata la condizione o su un altro è irrilevante: in quest'ultimo caso è solo segno di un cattivo assegnamento di responsabilità e della necessità di spostare l'eventuale metodo al posto giusto.

```
a.faiQualcosa()
```

E' evidente come il vero polimorfismo sia infinitamente superiore alla sua brutta copia: l'"if" viene fatto dal runtime automaticamente, e voi non dovete fare nulla di particolare se non usare degli oggetti in maniera polimorfa. Il polimorfismo dei poveri invece va invece mantenuto, ogni volta che si introducono nuovi tipi e nuove logiche: di solito fra l'altro questo tipo di strutture condizionali si trovano ripetute in diversi punti del codice, poichè è molto comune che esistano diverse logiche da eseguire, rendendo l'introduzione di bug così probabile da essere quasi scontata. Il vero polimorfismo invece vi permette semplicemente di aggiungere nuovi metodi per ogni nuova logica, concentrando così in unico punto tutto ciò che ha gli stessi motivi per cambiare.

Eseguite per curiosità un bel

grep -R instanceof .

nella directory demo del vostro JDK, e troverete molti esempi di poliformismo dei poveri. Sembra che Swing sia un portatore (sano?) di polimorfismo dei poveri!

Permettetemi di riformulare il concetto in maniera più immediata: se trovate del codice che usa "il polimorfismo dei poveri", è codice strutturato e non codice ad oggetti. Che sia nelle librerie di java, che l'autore sia considerato un guru, che sia uno dei software open più blasonati, che l'abbiate trovato su un libro (di solito di API, non certo di design), o che l'abbiate scritto voi stessi in quello che considerate il vostro capolavoro di programmazione ad oggetti, non cambia la sostanza: non sarà che in 20 anni questa benedetta programmazione ad oggetti a molti non è ancora entrata in testa?

Esercizi:

- Eliminare il polimorfismo dei poveri da uno qualsiasi degli esempi Swing del jdk. Come dite? Eh no, così bisogna rivedere le gerarchie... e poi inserire nuove interfacce. E poi bisogna eliminare quelle istanziazioni di oggetti concreti... bene, quella si chiama **programmazione ad oggetti**.

Ora salutate con la manina quella brutta e vecchia programmazione strutturata, e cercate di lasciarvela per sempre alle spalle!

Conclusioni:

Il polimorfismo dei poveri è una delle peggiori pratiche che si possa adottare per avere codice difficilmente manutenibile, inutilmente complicato e soggetto a bug. Il polimorfismo dei poveri indica inoltre che non state sfruttando la potenza degli oggetti. Se siete pagati per linee di codice, o per bug risolti, usatelo pure liberamente! Ma se poi vi trovate nei guai, non dite che nessuno vi aveva avvertito...