Design For Testing

TDD, DIP, Mock & C.
gabriele.zannoni@unibo.it

Design For Testing

- I sistemi software andrebbero progettati con un design tale da agevolare i test
- …è molto lontano da ciò che si è imparato finora?

Design For Testing VS Design Principles

Per agevolare i test:

- Ogni unità (classe) dovrebbe essere testabile singolarmente
  - Minimizzazione delle dipendenze: rispettare **DIP**
- Ogni unità (classe) dovrebbe avere solo le responsabilità che le competono
  - Se ha troppe responsabilità si rischia la "stratificazione" del codice → possono non esserci dipendenze (è tutto nella stessa classe…)
  - Meglio poche (o una) responsabilità: rispettare **SRP, ISP**

Design For Testing VS Design Principles

Per agevolare i test:

- L'aggiunta di una funzionalità non dovrebbe comportare l'alterazione dei test esistenti
  - Sistema aperto alle estensioni, chiuso alle modifiche: rispettare **Open/Close**

È strano che entrino in gioco i principi?

Mica tanto…

Design For Testing VS Design Principles

Ribaltiamo la frittata…

- Se una classe non rispetta **DIP** ci sono delle dipendenze
  - L'unità **riusabile** è la classe più tutte le sue dipendenze
  - Ciò che si riesce a testare è la classe… più tutte le sue dipendenze
- Se una classe non rispetta **SRP** + **ISP** ci sono troppe responsabilità "concentrate" e "troppo codice"
  - Difficile testare tutto correttamente → le funzionalità sono stratificate

Design For Testing VS Design Principles

Ribaltiamo la frittata…

- Se non rispettato **Open/Close**, l'aggiunta di una funzionalità è molto costosa in termini di modifiche
  - Si pensi ad un test come una estensione del sistema
  - L'aggiunta del test non deve alterare in alcun modo il codice da testare (ci mancherebbe!)

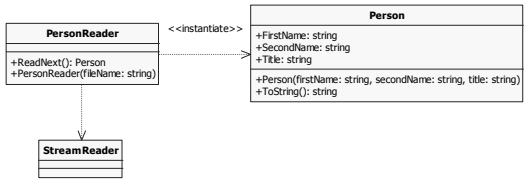
Un semplice esempio

- Leggere da un file un insieme di persone caratterizzate da nome, cognome, titolo.
- Il problema è evidentemente semplice.
- La soluzione **deve** essere semplice ma non semplicistica…
- Il file da leggere:

```
Gabriele;Zannoni;Ing.
Giuseppe;Bellavia;Prof.
Ugo;Fantozzi;Rag.
Francesco Maria;Barambani;Duca Conte, Gran Lup. Man., Farabut.
```

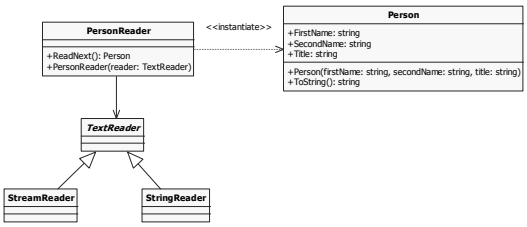
Dummy Mode 0

- Il minimo sindacale…



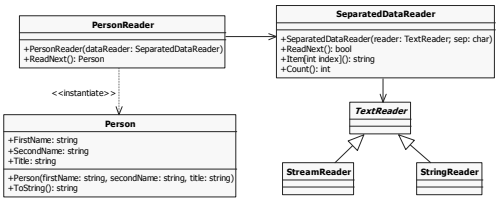
- Si può testare Person
- Si può testare PersonReader ma solo in **blocco**
  - La sorgente dati può essere solo un file
  - Tokenizzazione del file (separatori?) e creazione delle Persone
- Violati TUTTI i principi

Dummy Mode 1



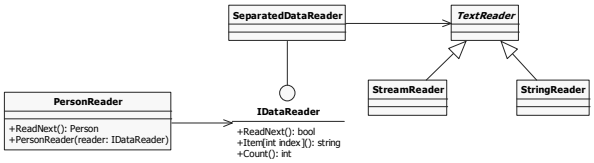
- La sorgente dati non deve più essere necessariamente un file – TextReader è astratta!
- Per il resto ci sono gli stessi problemi → testabilità solo in blocco!

Dummy Mode 2



- Separata la tokenizzazione dalla creazione.
- SeparatedDataReader è testabile singolarmente
- PersonReader è testabile solo in **blocco**

Smart Mode 0



- Ora anche PersonReader è testabile singolarmente a patto di fornire un'implementazione di IDataReader "adeguata" che non sia (ovviamente) SeparatedDataReader → Come?
- L'utilizzatore di PersonReader è testabile singolarmente?

Mock Objects

- Un oggetto Mock è un oggetto simulato che mima il comportamento di un oggetto "reale" in modo controllato (e controllabile)
- Nel caso specifico, PersonReader necessita di un IDataReader per essere testato correttamente in un ambiente "artificiale"
- Creare un Mock di IDataReader significa implementare IDataReader in modo "semplice" e, possibilmente, automatico

Mock Objects

- Esistono librerie per la generazione automatica dei Mock (generazione di codice "al volo")…
- …sui quali è possibile impostare:
  - **Risultati**
    - Cosa risponde il Mock di IDataReader all'invocazione dei metodi?
  - **Attese**
    - Quante volte deve essere invocato un metodo?
    - Con quali parametri?
- …ed è possibile verificare che le attese siano rispettate
  - **Verifiche**
    - i metodi sono stati invocati nell'ordine corretto, il numero di volte corretto e con i parametri corretti?

Demo

- Utilizzati i test di Visual Studio (anche generazione automatica degli stub di test)
- Utilizzata libreria di Mocking: **Rhino.Mocks** (open source): http://www.ayende.com/projects/rhino-mocks/downloads.aspx
  - Possibile scaricare il codice sorgente di Rhino.Mocks usando un client subversion
- Questo e altro sulla demo DesignForTesting!
- PS: Necessario VS2008