Design by Contract

H1 # Introduzione

Il **design by contract** è una metodologia di progettazione SW che segue un'idea valida soprattutto per la programmazione OO:
H2 l'**interfaccia** di un modulo definisce un **contratto**.

Contratto

> H2 Accordo esplicito tra un **cliente** e un **fornitore** che specifica *obblighi* e *benefici* delle due parti.

Nel SW avremo:

- **cliente**: chi usa il programma (*classe*)
- **fornitore**: chi scrive il programma (*classe*)

Precondizioni e postcondizioni

In un contratto si definisce:

H4
- cosa ogni metodo richiede (***precondizioni***)
- cosa ogni metodo fornisce (***postcondizioni***)

Esempio metodo `insert(element)` di una classe `Array`:

- Precondizioni:
  - l'array non è pieno:

    `n_elements < size`
- Postcondizioni:
  - l'array contiene un elemento in più:

    `n_elements' = n_elements+1`

==Le **precondizioni** permettono di limitare il numero e i tipi di possibili input di un metodo.== Inserire delle precondizioni forti semplifica il metodo, ma lo rende meno "universale", al contrario delle precondizioni deboli costringono il metodo a gestire più casi possibili.

Inoltre esiste la seguente notazione di casi particolari:

- programma(metodo) A
- precondizioni P
- postcondizioni Q

Potremmo avere $\{P\}A\{Q\}$ con:

- $P = false \implies$ il metodo non può essere chiamato
- $P = true \implies$ il metodo non ha precondizioni
- $Q = false \implies$ il metodo non può essere scritto
- $Q = true \implies$ il metodo non ha postcondizioni (può essere scritto in qualsiasi modo)

Invariante

Nel contratto si può definire anche una proprietà che vale sempre per tutte le istanze: l'**invariante**.

H4 Esso è vero dopo la creazione dell'oggetto e dopo ogni operazione.

Implicitamente, l'invariante definisce *obblighi ulteriori*:

- l'implementazione di ogni metodo non deve violare l'invariante
- chi implementa il metodo sa che l'invariante vale (**precondizione ulteriore**)

Esempio metodo `insert(element)` di una classe `Array` :

- Invariante:
  - Il numero di elementi è sempre compreso tra 0 e la dimensione massima:

    `0 ≤ n_elements ≤ size`

Eccezioni

A livello di contratto, le **eccezioni** corrispondono a una sua *violazione*:

H5

- violazione delle precondizioni (input errati, cioè colpa del client)
- violazione delle postcondizioni o dell'invariante

H3

Vantaggi del DbC

✔ Lo sviluppo diventa più focalizzato

✔ Basi solide per la scrittura di SW riusabile

✔ Gestione delle eccezioni guidata dalle precise definizioni di casi "normali" e "anormali"

✔ La documentazione è generata automaticamente

✔ Gli errori avvengono più vicini alla loro causa