Konstruktive Qualitätssicherung

- Software Richtlinien
- Typisierung
- Vertragsbasierte Programmierung
- Fehlertolerante Programmierung
- Portabilität
- Dokumentation

Idee von DbC

- Definition von formalen und messbaren Vereinbarungen für Schnittstellen zwischen Modulen.
- Überprüfung der Vereinbarungen
- **Design** by Contract: Vor der Implementierung werden die Vereinbarungen festgeschrieben.

Design by Contract

Gründer: Betrand Meyer im Zusammenhang mit Entwicklung der Programmiersprache Eiffel.

Es wird ein **Vertrag** zwischen Aufrufer und Aufgerufenem vereinbart, der Vor- und Nachbedingungen sowie Invarianten festlegt.

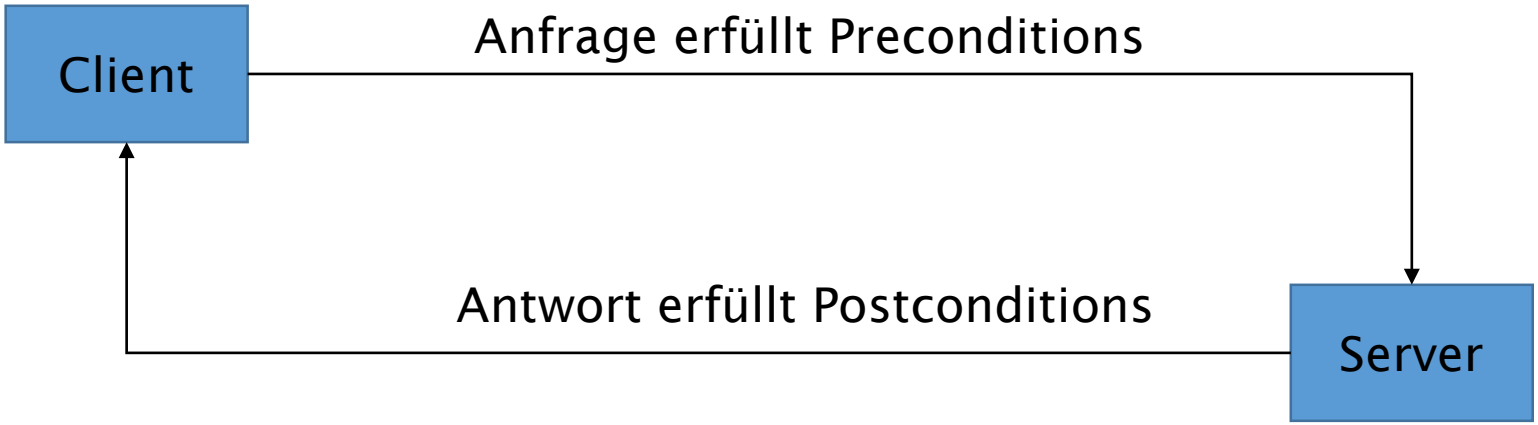
Grundprinzipien von DbC

Vorbedingungen:
Zusicherungen, die der Aufrufer zu beachten hat.

Nachbedingungen:
Zusicherungen, die der Aufgerufene zu beachten hat.

Invarianten:
Gesundheitszustand der Klasse. (logische Aussagen, die für alle Instanzen einer Klasse über den gesamten Objektlebenszyklus hinweg gelten)

DbC – Pre- und Postconditions



Verletzung der Constraints führt zu Programmabbruch.

Nutzen

- Steigerung der Qualität durch genaue Spezifikation der Schnittstellen
- Contracts als Dokumentation
- Unterstützung beim Testen/Fehlerfinden.

Umsetzung von Design by Contract

In manchen Sprachen nativ verankert: Eiffel, D

Design by Contract in Java:

- Nicht in der Sprache verankert.
- Konzept anwenden: javadoc (oder Äquivalent) nutzen um Vor-, Nachbedingungen und Invarianten zu definieren.
- Rudimentär: Arbeiten mit assert.
- Möglichkeit mit frameworks das Konzept zu implementieren (z.B. http://www.valid4j.org/)
- Bsp: Contracts for Java

Beispiel

Contract for Java
(https://github.com/nhatminhle/cofoja):

```
@Requires("x >= 0")
@Ensures("result >= 0")
static double sqrt(double x);
```

Default: keine Auswirkung,
Contracts angeschalten: Spezifische Laufzeitexceptions bei Verletzungen (PreconditionError, PostConditionError, InvariantError

DbC und Vererbung

- PreConditions können in den vererbten Methoden gelockert werden.
- PostConditions können strenger gemacht werden.
- Invariants können strenger gemacht werden.
- Zusätzliche Einschränkungen sind bei den abgeleiteten Methoden möglich.