Ziel

- Die Möglichkeiten von **Code Contracts** kennen lernen.
- **Weiterführende Aspekte** berücksichtigen.

Inhalt – Teil 1: Überblick

- Beispiel ohne Code Contracts
- Überblick zu Code Contracts
- Beispiel += Code Contracts

Inhalt – Teil 2: Erweiterte Aspekte

- Vererbung/ Interfaces/ abstrakte Klassen
- Zusätzliche statische Prüfungen
- Exceptions und Unit-Testing
- Laufzeitverhalten
- Contracts und bestehende Projekte
- Weitergeben von Libs

DEMO

ÜBERBLICK

Design by Contract

- Reibungsloses Zusammenspielen von Komponenten durch Verträge
- Fehler bereits im Zuge der Entwicklung entdecken
- Eiffel, 1985 (!)

Vertragsarten

- Vorbedingungen
- Nachbedingungen
- Invarianten

Code Contracts

- **System.Diagnostic.Contracts**

  → Bestandteil von .Net 4

- **Tools** für statische und dynamische Analyse

  → Download:

  http://msdn.microsoft.com/de-de/devlabs/dd491992.aspx

.

Code Contracts

```
public int DoStuff(int i)
{
    Contract.Requires(i >= 0);
    return i * 2;
}
```

Code Contracts

```
public int DoStuff double(int i)
{
  Contract.Requires(i >= 0);
  Contract.Ensures(
      Contract.Result<int>() >= 0);

    return i * 2;
  }
```

BASTA! SQLCON 2010 Share Connect 2010

Code Contracts

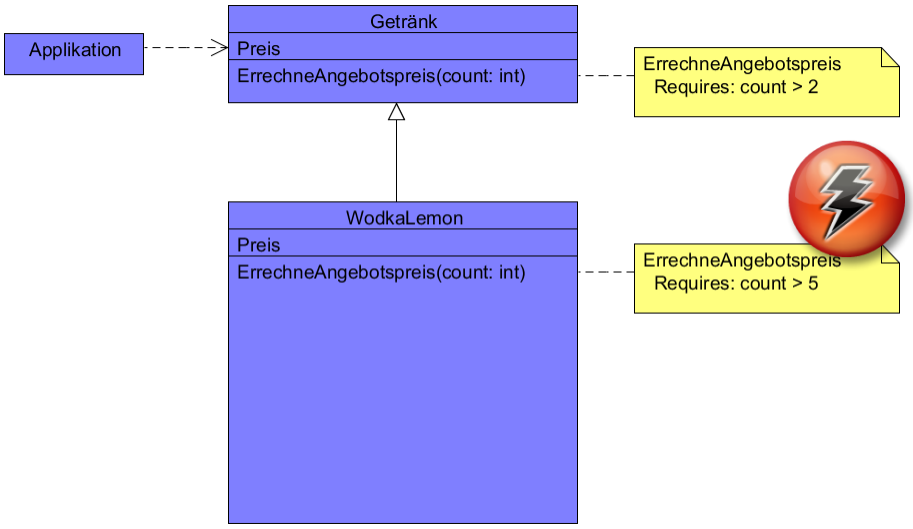
```
[ContractInvariantMethod]
private void ObjectInvariant() {

  Contract.Invariant(Counter >= 0);

}
```

DEMO

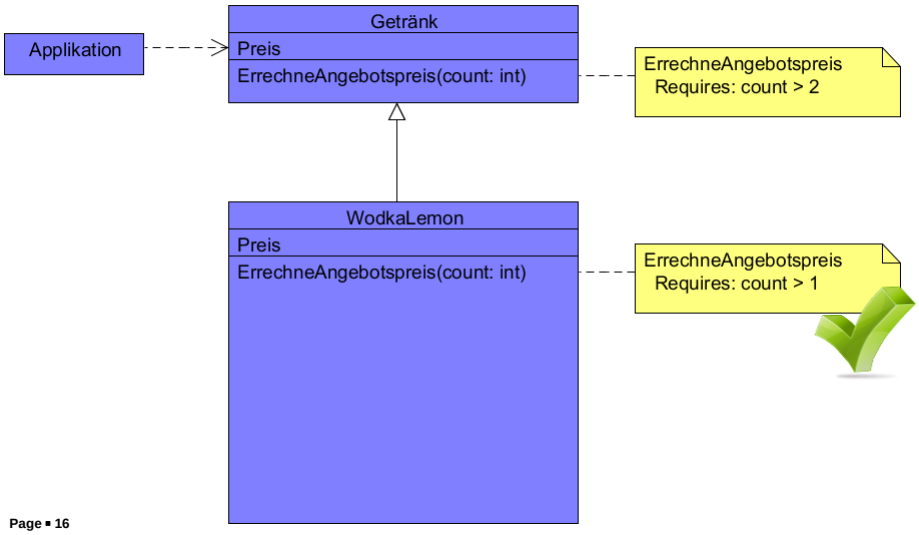
Minibar mit Code Contracts

VERERBUNG

Contracts in Subtypen



Contracts in Subtypen



Realisierung in Code Contracts

- Subtypen dürfen **keine weiteren Vorbedingungen** einführen
- Subtypen dürfen **jedoch weitere Nachbedingungen** einführen

Demo

Contracts in Subtypen

INTERFACES UND ABSTRAKTE KLASSEN

Interfaces und Abstrakte Klassen

```
public class Getraenk
{
    public int CalcPreis(int count)
    {
        Contract.Requires(count > 0);
        return 0;
    }
}
```

Interfaces und Abstrakte Klassen

```
public interface Getraenk
{
    int CalcPreis(int count);
}
```


Contract-Klassen

```
[ContractClass(typeof(GetraenkContracts))]
public interface Getraenk
{
    int CalcPreis(int count);
}

[ContractClassFor(typeof(Getraenk))]
public class GetraenkContracts : Getraenk
{
    public int CalcPreis(int count)
    {
        Contract.Requires(count > 0);
        return 0;
    }
}
```

Demo

Interface mit Contracts

ZUSÄTZLICHE STATISCHE PRÜFUNGEN

Zusätzliche statische Prüfungen

- Non-Null Obligations
- Array Bounds Obligations
- Arithmetic Obligations

Demo

EXCEPTIONS UND UNIT-TESTING

Verhalten bei Vertragsverletzung

- **Assert**verletzung (Konfigurierbar)
- **Exception** (Standard)
- Contract.Requires<**T**>(…)
- **Event** auslösen (ContractFailed)

Demo

ContractFailed

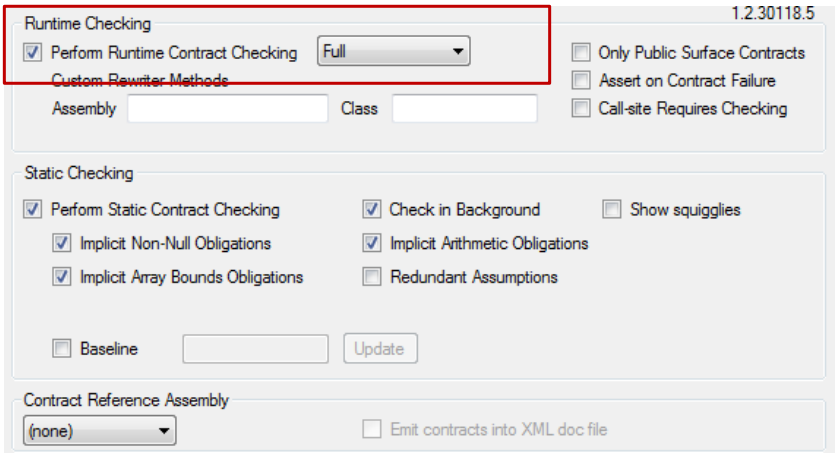
LAUFZEITVERHALTEN

Runtime Checking Level

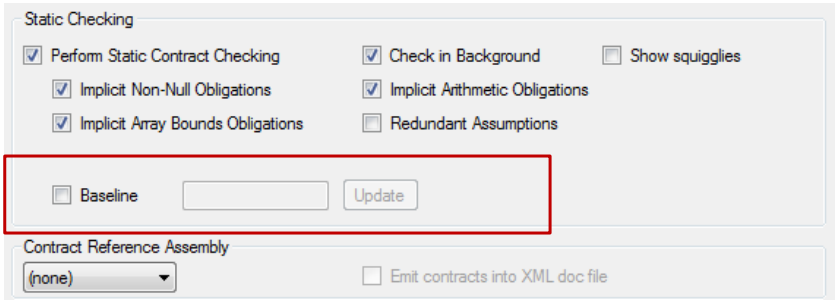
	Requires<E>	Requires	Ensures	Invariants
Full	x	x	x	x
Pre and Post	x	x	x	
Preconditions	x	x		
ReleaseRequires	x			
None				

BESTEHENDE PROJEKTE

Runtime Checking Level

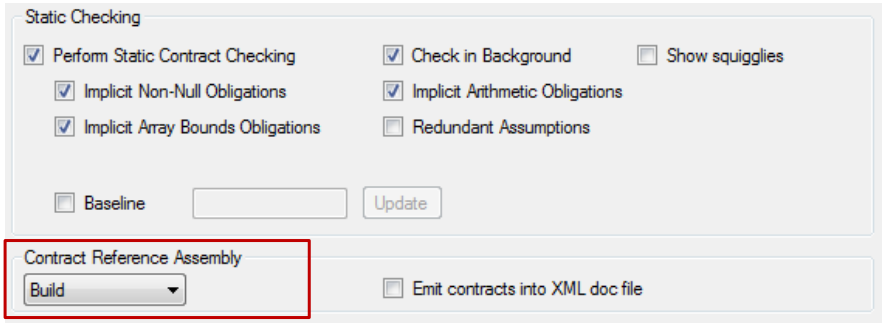


Bestehende Projekte



WEITERGABE VON LIBS

Weitergabe von Libs



ZUSAMMENFASSUNG

Zusammenfassung

- **Contracts:** Fehler bereits im Zuge der Implementierung finden
- **Bestehen aus:** Vorbedingungen, Nachbedingungen, Invarianten

Zusammenfassung

- **Subklassen:** Keine weiteren Vorbedingungen erlaubt!
- **Interfaces:** ContractClass
- **Statische Prüfungen:** Non-Null, Arithmetic, Array Bounds

BASTA! SQLCON 2010 Share Connect 2010

Zusammenfassung

- **Vertragsvertragsverletzungen**
  - Assertverletzungen vs. Exceptions
  - ContractFailed

Zusammenfassung

- **Laufzeitverhalten:** Runtime Checking Level
- **Bestehende Projekte:** BaseLine
- **Ausliefern:** Contract Reference Assembly

BASTA! SQLCON 2010 Share Connect 2010

Kontakt und weitere Infos

- **Manfred Steyer**
- www.softwarearchitekt.at
- manfred.steyer@softwarearchitekt.at