12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vertragssicher
Code Contracts

Michael Wiedeking

MATHEMA Software GmbH

Der Wächter

```
double sqrt(double x) {

    if (x < 0) {
        throw new IllegalArgumentException();
    }

    // Berechnung der Quadratwurzel

}
```

Vorbedingung

```
double sqrt(double x) {
    assert (x ≥ 0);
    // Berechnung der Quadratwurzel
}
```

Nachbedingung

```
double sqrt(double x) {

    double result = … // Berechnung der Quadratwurzel

    assert (abs((result * result) – x) < ε);
    assert (result ≥ 0);

    return result;

}
```

Vor- und Nachbedingungen

```
void increment() {

    assert (this.i > 0);

    int old_i = this.i;
    this.i += 1;

    assert (this.i == old_i + 1);

}
```

Invarianten

void f (double a, double b) {

assert (***this***.$\underline{v} \geq 0$);

this.$\underline{v}$ = ***this***.$\underline{v} - a + b$;

assert (***this***.$\underline{v} \geq 0$);

}

Beispiel: Code Contracts …

```
using System;
using System.Diagnostics.Contracts;

namespace ContractExample1 {

    class Rational {
        int numerator;
        int denominator;
        …
    }

}
```

Beispiel: Code Contracts …

```
public Rational(int n, int d) {

    Contract.Requires(d != 0);

    this.numerator = n;
    this.denominator = d;

}
```

Beispiel Code Contracts …

```
public int Denominator {
    get {
        Contract.Ensures(Contract.Result<int>() != 0);
        return this.denominator;
    }
}
```

Beispiel Code Contracts

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

    Contract.Invariant(this.denominator != 0);

}
```

Vorbedingung

- Contract.Requires(*x* != null);
- Contract.Requires<ArgumentNullException>(*x* != null, "x");

Vorbedingung

```
double sqrt(double x) {

    if (x < 0) {
        throw new IllegalArgumentException();
    }

    if (x == 0) {
        throw new DegeneratedResultException();
    }

    // Berechnung der Quadratwurzel

}
```

Abgrenzung der Vorbedingungen

```
double sqrt(double x) {

    if (x < 0) {
        throw new IllegalArgumentException();
    }

    Contract.EndContractBlock()

    if (x == 0) {
        throw new DegeneratedResultException();
    }

    // Berechnung der Quadratwurzel
}
```

Nachbedingung …

- Contract.Ensures(***this***.*value* > 0);
- Contract.EnsuresOnThrow<T>(***this***.*value* == 0);
- Contract.Ensure(Contract.Result<int>() > 0);
- Contract.Ensures(Contract.OldValue(*xs*.Length) > *xs*.Length)

Aber:

- Nachbedingung muss berechenbar sein (*xs* != **null** || *E*)
- Contract.Result<T>() und *out*-Parameter können nicht in einer Contract.Old-Anweisung benutzt werden

Nachbedingung

```
public void OutParam(out int x) {
    Contract.Ensures(Contract.ValueAtReturn(out x) == 0);
    x = 0;
}
```

Invarianten

```
[ContractInvariantMethod]
protected void ObjectInvariant() {
    Contract.Invariant(velocity > 0);
    Contract.Invariant(direction != null);
}
```

- nur nach Aufruf einer *public*-Methode
- ggf. Aussetzen der Prüfung (reentrant)
- nicht bei *finalize* oder *dispose*

Annahmen …

f = createByName("x");

Contract.Assert(*f* != **null**)

Annahmen

f = createByName("x");

Contract.Assume(*f* != null)

Bedingungen für Collections …

```
public void findMultiples<T>(IEnumerable<T> collection) {

    Contract.Requires(
        Contract.ForAll(collection, (T x) => x != null)
    );

}
```

Bedingungen für Collections …

```
public int[] Squares() {

    Contract.Ensures(
        Contract.ForAll(
            0,
            Contract.Result<int[]>().Length,
            i => Contract.Result<int[]>()[i] > 0
        )
    );

}
```

Bedingungen für Collections

- Contract.ForAll
- Contract.Exists
- System.Linq.Enumable.All
- System.Linq.Enumable.Any

Schnittstellen …

```
[ContractClass(typeof(IFooContract))]
interface IFoo {
    int Count { get; }
    void Put(int value);
}
```

Schnittstellen …

```
[ContractClassFor(typeof(IFoo))]
sealed class IFooContract : IFoo {
    int IFoo.Count {
        get {
            Contract.Ensures(Contract.Result<int>() >= 0);
            return default(int);
        }
    }
    void IFoo.Put(int value){
        Contract.Requires(value ≥ 0);
    }
}
```

Schnittstellen

```
[ContractClassFor(typeof(IFoo))]
sealed class IFooContract : IFoo {

    void IFoo.Put(int value){
        IFoo this = this;
        Contract.Requires(value >= 0);
        Contract.Requires(this.Count < 10);
    }

}
```

Abstrakte Klassen …

```
[ContractClass(typeof(FooContract))]
abstract class Foo {

    public abstract int Count { get; }

    public abstract void Put(int value);

}
```

Abstrakte Klassen

```
[ContractClassFor(typeof(Foo))]
abstract class FooContract : Foo {
    public override int Count {
        get {
            Contract.Ensures(Contract.Result<int>() ≥ 0);
            return default(int);
        }
    }
    public override void Put(int value){
        Contract.Requires(value ≥ 0);
    }
}
```

Referenzen

- Microsoft Corporation
  Code Contracts User Manual

  http://download.microsoft.com/download/

  C/2/7/C2715F76-F56C-4D37-9231-EF8076B7EC13/userdoc.pdf
- Microsoft Research
  Code Contracts

  http://research.microsoft.com/en-us/projects/contracts/

Fragen?

Vielen Dank!

michael.wiedeking@mathema.de