# Contract-based Software Development

Rasmus Guldborg Pedersen

January 2015

# Overview

1. Formalism
2. Code Contracts
   - The tooling
   - Class Specification

# Formal specification of classes

How can formalism be used to make a precise specification of a class? Relate to Code Contracts.

# Informal Specification

```
public interface ISimpleDictionary {
    /* Put 'key' into the dictionary with associated
        'value'  */
    void Put(string key, object value);

    /* Remove 'key' from the dictionary */
    void Remove(string key);

    /* Does the dictionary contain 'key'? */
    bool ContainsKey(string key);
}
```

# Formal Specification

```
public interface ISimpleDictionary {
    /* Pre: key != null && !ContainsKey(key)
        Post: ContainsKey(key) */
    void Put(string key, object value);

    /* Pre: key != null && ContainsKey(key)
        Post: !ContainsKey(key) */
    void Remove(string key);

    /* Pre: key != null */
    [Pure]
    bool ContainsKey(string key);
}
```

# Code Contracts (.NET tool)

Express preconditions, postconditions and object invariants for:

# Code Contracts (.NET tool)

Express preconditions, postconditions and object invariants for:

- Static analysis

# Code Contracts (.NET tool)

Express preconditions, postconditions and object invariants for:

- Static analysis
- Documentation

# Code Contracts (.NET tool)

Express preconditions, postconditions and object invariants for:

- Static analysis
- Documentation
- Runtime checking

# Interface

```
public interface ISimpleQueue {
    void Enqueue(object item);
    object Dequeue();
    object ElementAt(int index);
    int Count();
}
```

# Contract

```
abstract class ISimpleQueueContract {
    public void Enqueue(object item) {
        Contract.Requires(item != null);
        Contract.Ensures(Count ==
            Contract.OldValue(Count()) + 1);
        Contract.Ensures(ElementAt(Count()) == item);
        // ...
    }
    // ...
}
```

# Associating Interface with Contract

```
[ContractClass(typeof(ISimpleQueueContract))]
public interface ISimpleQueue { /* ... */ }

[ContractClassFor(typeof(ISimpleQueue))]
abstract class ISimpleQueueContract { /* ... */ }
```

# The End

*"Testing shows the presence, not the absence of bugs."*
*— Edsger W. Dijkstra*