Invariants Runtime Checking Example

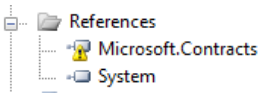
Abstract

This example shows the runtime checking of invariants. It also points out a subtle issue of when an invariant check is *not* done. The example consists of three classes: a base class BasicBankAccount, a subtype WallStreetAccount, and a client, Client that creates bank accounts and calls methods on them.

1 Adding the Contract Library Reference

1.1 Visual Studio 2008

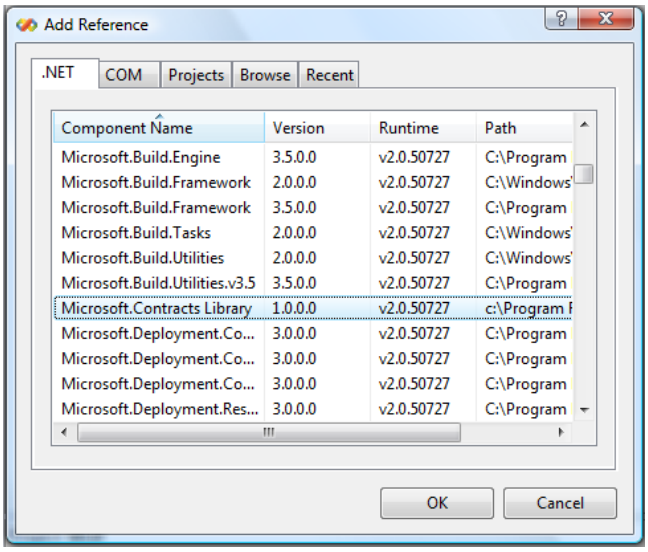
Before getting started with the sample, we need to make sure we have the proper reference for Microsoft.Contracts.dll. Look at the references of project Invariants:



If Visual Studio cannot find the Microsoft.Contracts.dll (indicated by the yellow warning), delete the reference and follow the steps below. Otherwise, you are ready to go to Section 2.

For new projects to use contracts, we need to add a reference to the Microsoft.Contracts library to the project.

Assuming you have installed the code contract tools, open the Invariants solution and right-click on References in the Invariants project and select Add Reference. Find the Microsoft.Contracts library in the .NET tab as shown below and click OK.



1.2 Visual Studio 2010

If you want to try the sample with Visual Studio 2010 you have two choices:

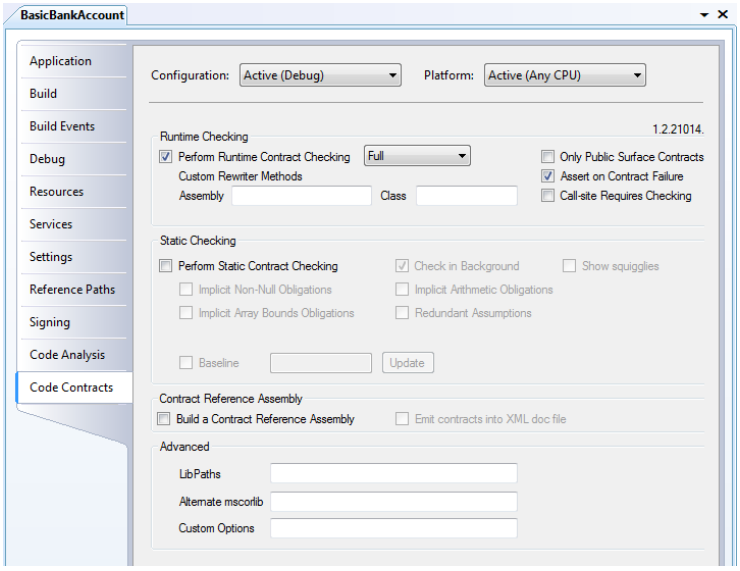
- Target the v3.5 framework (the sample default). In this case, follow the instructions under Visual Studio 2008 as this mode will require the Microsoft.Contracts library to be referenced.
- Target the v4.0 framework. In this case, the contract class is defined in mscorlib and *no* reference to Microsoft.Contracts.dll is necessary.

Thus, to target v4.0 you need to perform the following steps:

- Delete the Microsoft.Contracts.dll reference from all projects
- Change the target framework of all projects to v4.0.

2 Sample Walkthrough

After adding the proper reference, go to the Properties of project BasicBankAccount, select the Code Contracts pane (at the bottom), and enable runtime checking by clicking on the runtime checking box, as shown in this screenshot:

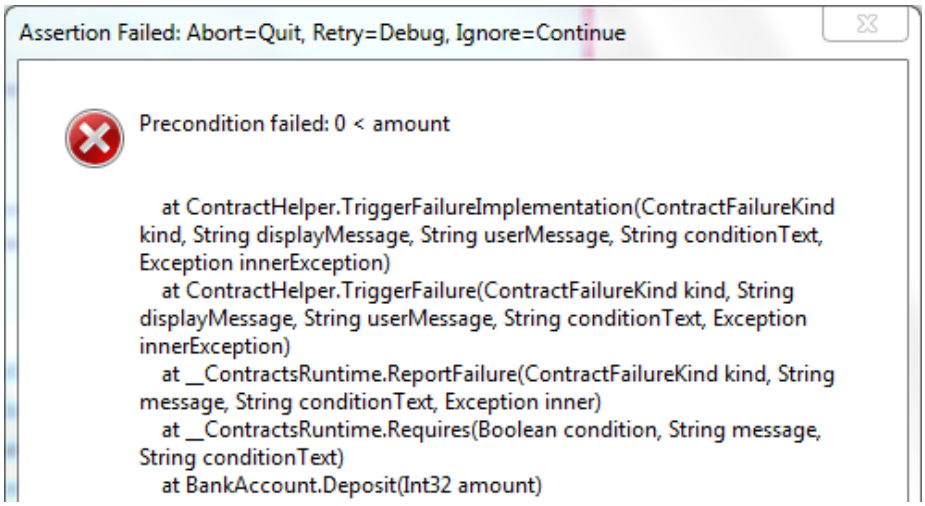


Then build the entire solution. The build should succeed. Now let's look at some of the code in the solution.

First, take a look at MainStreet.cs which contains the very simple class BankAccount. It has one method for putting money into the account, Deposit, another for removing money from the account, Withdraw, and a property, Balance, that has only a getter for returning the amount of money that is in the account.

What is different (aside from the contracts in each of those methods) is that an object invariant has been added to the class, a method named GoodAccount. (The full details about invariant methods and how to write them can be found in the general documentation.) This method gets checked at the end of every public method in the class.

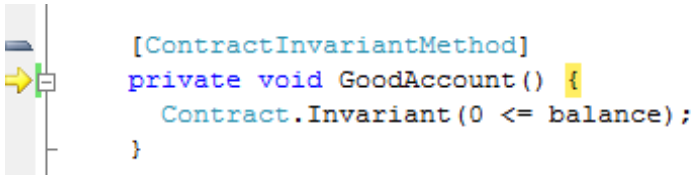
Now run the program by, e.g., pressing F5 to start the debugger. You should see the following "assert" dialog box:



This failed contract is easily explained if you look at the code in Deposit:

```
public virtual void Deposit(int amount) {
  Contract.Requires(0 < amount);
  this.balance += amount;
}
```

and then notice that the client was calling it with a negative number. Fix that in Program.cs by changing −5 to some positive number. To see it in action, press F10 to start the debugger and have it start at the first line in the program. Press F10 twice to get to the call to Deposit. Now press F11 to step into the method call. Pres F11 four more times and you'll see that you are now entering the object invariant:



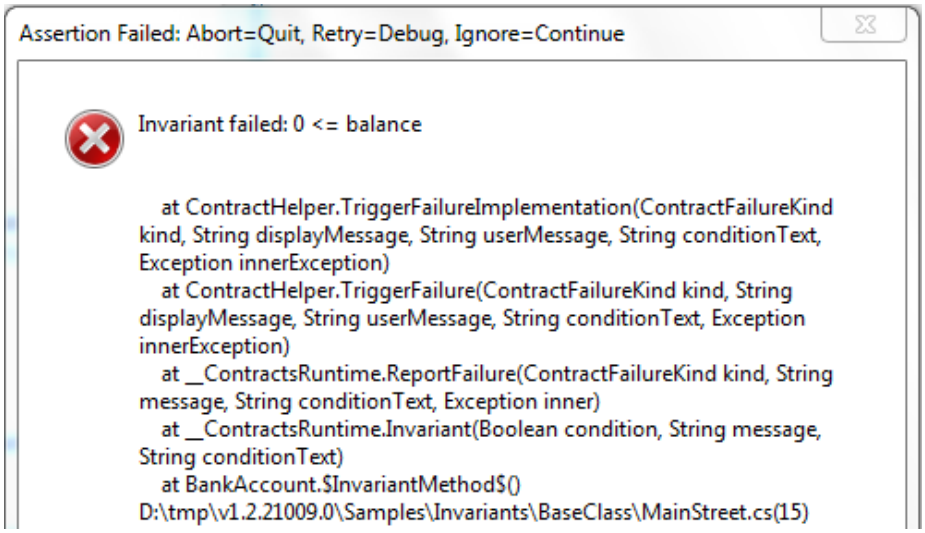
Continue execution by pressing F5 and the program will run to completion without an error.

However, if you look at the code in WallStreet.cs, you'll see that the override it has for Deposit does not maintain the invariant of its supertype.

```
public override void Deposit(int amount) {
  if (100 < amount) {
    this.slushFund += amount;
    this.balance -= amount;
  } else {
    base.Deposit(amount);
  }
```

```
}
```

The problem is that runtime checking was enabled for the base class, but not for the subtype. So now turn on the runtime checking for the project WallStreet and press F5 to run the program again. This time you should see another error dialog, this time for the invariant violation:



Note that the invariant check is injected into the subtype even though it has no contracts at all — not even a reference to the contract library!

The subtle point that is important to realize is that if you go back to the settings for the BasicBankAccount project and now turn *off* the runtime checking, then even though runtime checking is turned on for WallStreet, the invariant check will not be injected into the code and no invariant violation will be found.