

Assignment 12

Anowarul Kabir

Clayton Mottley

Xavier Gitiaux

Contributions:

1. As always we came up with our own implementation first.
2. Xavier had the first implementation of *IntSet* in C#
3. Clayton added contracts to support the rep invariant.
4. Anowarul wrote some tests to check the working of the contracts.

Visual Studio includes a Code Contracts mechanism which allows users to easily translate contract specifications (preconditions and postconditions) into executable code. To explore this functionality, we took Liskov's basic *IntSet*, a very common reference for contracts throughout this semester, and implemented Code Contracts. First, we translated the Java *IntSet* to C# (also adding a *repOk()* check for us to validate the rep-invariant) in order to make it compatible with Visual Studio and for us to use the Code Contracts library.

```
/*
 * Overview: LiskovIntSets are unbounded, mutable sets of integers.
 * AF(c) = c.x0, c.x1, c.x2 ... .. c.xn where 0 <= n < size-1
 * rep-inv:
 * 1. c.els != null
 * 2. all elements in c.els are integers
 * 3. no duplicates in c.els
 *   for all i,j where 0 <= i,j < size-1
 *     if i!=j then c.els[i] != c.els[j]
 */
public class LiskovIntSet{
    private readonly List<int> els;

    /*
     * Initializes this to be empty
     */
    public LiskovIntSet(){els = new List<int>();}

    /*
     * Modifies: this
     * Effects: adds x into this
     */
}
```

```

    * PostConditions: x is in this and rep-inv holds
    */
    public void Insert(int x){...}

    /*
    * Modifies: this
    * Effects: removes x into this
    * PostConditions: x is not in this and rep-inv holds
    */
    public void Remove(int x){...}

    /*
    * Effects: returns true if x is in this else returns false
    */
    public bool IsIn(int x){...}

    /*
    * Effects: returns the cardinality of this
    */
    public int Size(){...}

    /*
    * Effects: if this is empty, throw InvalidOperationException else
    * returns an arbitrary element of this.
    */
    public int Choose(){...}

```

A full implementation in C# is at the end of our solution.

Next, we turned the contract specifications (written in Doc comments) into runnable code using the Code Contracts library. With this library, preconditions map to *Contract.Requires()* and postconditions translate to *Contracts.Ensures()*. Fortunately, our *IntSet* has a total design and all preconditions are replaced with defined behavior. However, we were still able to translate the postconditions to *Contracts* calls.

After inserting an integer into the *IntSet*, the contract states that the newly added element should be added to the set. We verified this postcondition by checking that the element *IsIn()* the *IntSet* at the end of the *Insert()* completion.

```

public void Insert(int x)
{
    if (!els.Contains(x))
        els.Add(x);
    Contract.Ensures(els.Contains(x), "LiskovIntSet.Insert: item is not properly inserted");
    Contract.Ensures(RepOk(), "LiskovIntSet.Insert: rep-inv is broken");
}

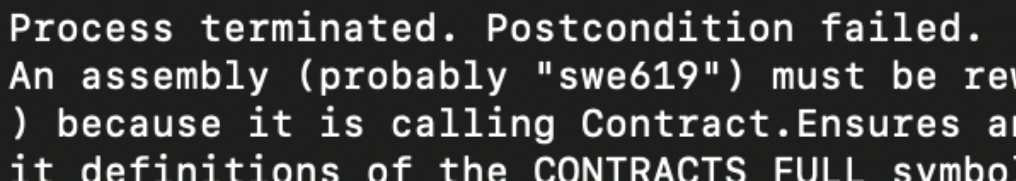
```

Similarly, after removing an integer from the *IntSet*, the contract states that the element should no longer be in the set. We checked this postcondition by checking that the integer is not *IsIn()* the *IntSet* at the end of *Remove()* completion.

```
public void Remove(int x)
{
    els.Remove(x);
    Contract.Ensures(!els.Contains(x), "LiskovIntSet.Remove: item is not properly removed");
    Contract.Ensures(RepOk(), "LiskovIntSet.Remove: rep-inv is broken");
}
```

For any data abstraction, the rep-invariant must always hold at the end of a method completion. We managed to turn this implicit postcondition into executable code by verifying *repOk()* as the last stage of every mutator call with Contracts.

Placing these Contract assertions for postconditions allows us to easily ensure that our method implementations are in line with their specifications. We ran a test by changing our *Insert()* functionality to only add an integer to the set if it already exists in the set. When we called *Insert()* within our test code we immediately received an error that the postcondition failed.



```
Process terminated. Postcondition failed.
An assembly (probably "swe619") must be reloaded
because it is calling Contract.Ensures at a location
not defined in its metadata. Please check the definitions
of the CONTRACTS_FULL symbol.
```

During the actual implementation phase this feedback would be extremely useful for automatically checking the correctness of an abstraction with complicated contracts and/or implementations as a form of test-driven development. We would be confident in always being able to say that our code satisfies the contract without having to do a full manual analysis after every change.

```
using System;
using System.Collections.Generic;
using System.Diagnostics.Contracts;
using System.Text;
```

```
namespace Question1
{
    /*
    * Overview: LiskovIntSets are unbounded, mutable sets of integers.
    * AF(c) = c.x0, c.x1, c.x2 ... .. c.xn where 0 <= n < size-1
    * rep-inv:
    * 1. c.els != null
    * 2. all elements in c.els are integers
    */
}
```

```

* 3. no duplicates in c.els
*   for all i,j where  $0 \leq i,j < \text{size}-1$ 
*     if  $i \neq j$  then  $c.\text{els}[i] \neq c.\text{els}[j]$ 
*/
public class LiskovIntSet
{
    private readonly List<int> els;

    /*
    * Initializes this to be empty
    */
    public LiskovIntSet()
    {
        els = new List<int>();
    }

    /*
    * Modifies: this
    * Effects: adds x into this
    * PostConditions: x is in this and rep-inv holds
    */
    public void Insert(int x)
    {
        if (!els.Contains(x))
            els.Add(x);
        Contract.Ensures(els.Contains(x), "LiskovIntSet.Insert: item is not properly inserted");
        Contract.Ensures(RepOk(), "LiskovIntSet.Insert: rep-inv is broken");
    }

    /*
    * Modifies: this
    * Effects: removes x into this
    * PostConditions: x is not in this and rep-inv holds
    */
    public void Remove(int x)
    {
        els.Remove(x);
        Contract.Ensures(!els.Contains(x), "LiskovIntSet.Remove: item is not properly removed");
        Contract.Ensures(RepOk(), "LiskovIntSet.Remove: rep-inv is broken");
    }

    /*
    * Effects: returns true if x is in this else returns false
    */
    public bool IsIn(int x)
    {
        return els.Contains(x);
    }
}

```

```

/*
 * Effects: returns the cardinality of this
 */
public int Size()
{
    return els.Count;
}

/*
 * Effects: if this is empty, throw InvalidOperationException else
 * returns an arbitrary element of this.
 */
public int Choose()
{
    Contract.Requires<InvalidOperationException>(els.Count != 0, "LiskovIntSet.Choose: LiskovIntSet
is empty.");
    return els[0];
}

/*
 * Effects: returns string representation of this
 */
public override string ToString()
{
    return els.ToString();
}

/*
 * Effects: returns true if rep-inv is ok, else returns false
 */
[ContractInvariantMethod]
public bool RepOk()
{
    if (els == null)
        return false;
    for (int i = 0; i < els.Count; i++)
        for (int j = 0; j < els.Count; j++)
            if (i != j && els[i] == els[j])
                return false;

    return true;
}
}
}

```