

Higher Order Programming in C# with LINQ

CIS399-005 Code Appendix

Jeff Vaughan

April 9, 2008

1 An enumerator for a constant sequence

```
1 using System;
  using System.Collections.Generic;

  class TenEnumerator : IEnumerator<int>
  {
6     private int myCurrent;

    public TenEnumerator()
    {
11     Reset();

    object System.Collections.IEnumerator.Current
    {
16     get { return myCurrent; }

    public int Current
    {
21     get { return myCurrent; }

    public bool MoveNext()
    {
26     if (Current == 10)
        return false;

        myCurrent = myCurrent + 1;
        return true;
31     }

    public void Reset()
    {
36     myCurrent = 0;

    public void Dispose() { }
```

```
    }  
41  class Printer  
    {  
  
        static void Main()  
46    {  
            var e = new TenEnumerator();  
  
            while (e.MoveNext())  
            {  
51                Console.WriteLine(e.Current);  
            }  
        }  
    }  
}
```

2 An enumerator for a sequence with settable range

```
1 using System;
  using System.Collections.Generic;

  class SeqEnumerator : IEnumerator<int>{

6   private int myMin;
     private int myMax;
     private int myCurrent;

     public SeqEnumerator(int min, int max)
11  {
        myMin = min; myMax = max; Reset();
    }

     object System.Collections.IEnumerator.Current{
16  get{return myCurrent;}
    }

     public int Current{
21  get{return myCurrent;}
    }

     public bool MoveNext(){
26     if (Current == myMax)
         return false;

        myCurrent = myCurrent + 1;
        return true;
    }

31  public void Reset(){
        myCurrent=myMin-1;
    }

36  public void Dispose() {}
    }

  class Printer{
41
     static void Main(){
         var e = new SeqEnumerator(1, 10);

         while(e.MoveNext()){
46             Console.WriteLine(e.Current);
         }
     }
 }
```

3 Lifting the enumerator to a collection

```
1 using System;
using System.Collections.Generic;
using System.Collections;

class SeqEnumerator : IEnumerator<int>{
6
    private int myMin;
    private int myMax;
    private int myCurrent;

11 public SeqEnumerator(int min, int max)
    {
        myMin = min; myMax = max; Reset();
    }

16 object IEnumerator.Current{
    get{return myCurrent;}
}

21 public int Current{
    get{return myCurrent;}
}

    public bool MoveNext(){
26     if (Current == myMax)
        return false;

        myCurrent = myCurrent + 1;
        return true;
31 }

    public void Reset(){
        myCurrent=myMin-1;
    }

36 public void Dispose() {}
}

class SeqCollection : IEnumerable<int>
41 {
    private int myMin;
    private int myMax;

    public SeqCollection(int min, int max){myMin = min; myMax = max;}

46 public IEnumerator<int> GetEnumerator(){
        return new SeqEnumerator(myMin, myMax);
    }

51 IEnumerator IEnumerable.GetEnumerator() {
        return null;
    }
}
```

```
56 class Printer{  
    static void Main(){  
        var c = new SeqCollection(1, 10);  
        var e = c.GetEnumerator();  
61  
        while(e.MoveNext()){  
            Console.WriteLine(e.Current);  
        }  
    }  
66 }
```

4 Introducing yield

```
using System;
using System.Collections.Generic;
using System.Collections;
4
class SeqCollection : IEnumerable<int>
{
    private int myMin;
9    private int myMax;

    public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
14        for(int i = myMin; i <= myMax; i++)
            yield return i;

    }
19    IEnumerator IEnumerable.GetEnumerator() {
        return null;
    }
24 }
class Printer{

    static void Main(){
        var c = new SeqCollection(1, 10);
29        var e = c.GetEnumerator();

        while(e.MoveNext()){
            Console.WriteLine(e.Current);
        }
34 }
}
```

5 Introducing foreach

```
using System;
using System.Collections.Generic;
using System.Collections;

5
class SeqCollection : IEnumerable<int>
{
    private int myMin;
    private int myMax;
10
    public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
15
        for(int i = myMin; i <= myMax; i++)
            yield return i;

        }

20
    IEnumerator IEnumerable.GetEnumerator() {
        return null;
    }
}

25
class Printer{

    static void Main(){
        var c = new SeqCollection(1, 10);
30
        foreach(int i in c){
            Console.WriteLine(i);
        }
    }
}
```

6 Defining a filter transformation by iterating

```
1 using System;
  using System.Collections.Generic;
  using System.Collections;

6 class SeqCollection : IEnumerable<int>
  {
    private int myMin;
    private int myMax;

11 public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
      for(int i = myMin; i <= myMax; i++)
16        yield return i;
    }

    IEnumerator IEnumerable.GetEnumerator() {
21        return null;
    }
  }

  static class HelperMethods{
26
    public static bool IsOdd(int x){ return (x % 2 == 1); }

    public static IEnumerable<int> FilterOdds(IEnumerable<int> input){
31
      foreach(int i in input)
      {
        if (IsOdd(i))
          yield return i;
      }
36
    }
  }

41 class Printer{

    static void Main(){
      var c = new SeqCollection(1, 10);
      var cc = HelperMethods.FilterOdds(c);
46
      foreach(int i in cc){
        Console.WriteLine(i);
      }
    }
51 }
```


7 Generalizing filter as a higher order function

```
using System;
using System.Collections.Generic;
using System.Collections;
4

class SeqCollection : IEnumerable<int>
{
    private int myMin;
    9    private int myMax;

    public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
14        for(int i = myMin; i <= myMax; i++)
            yield return i;
    }

    19    IEnumerator IEnumerable.GetEnumerator() {
        return null;
    }
}

24 static class HelperMethods{

    public static bool IsOdd(int x){ return (x % 2 == 1); }
    public static bool IsEven(int x){ return (x % 2 == 0); }
    public static bool IsLarge(int x){ return (x > 5); }
    29

    public delegate bool IntPredicate(int x);

    public static IEnumerable<int>
        Filter(IEnumerable<int> input, IntPredicate f){
34        foreach(int i in input)
        {
            if (f(i))
                yield return i;
39        }
    }

}

44 class Printer{

    static void Main(){
        var c = new SeqCollection(1, 10);
        var cc = HelperMethods.Filter(c, HelperMethods.IsLarge);
49

        foreach(int i in cc){
            Console.WriteLine(i);
        }
    }
54 }
```

8 Calling filter with an anonymous lambda

```
1 using System;
  using System.Collections.Generic;
  using System.Collections;

6 class SeqCollection : IEnumerable<int>
  {
    private int myMin;
    private int myMax;

11 public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
        for(int i = myMin; i <= myMax; i++)
16         yield return i;
    }

    IEnumerator IEnumerable.GetEnumerator() {
21         return null;
    }
  }

  static class HelperMethods{
26
    public delegate bool IntPredicate(int x);

    public static IEnumerable<int>
        Filter(IEnumerable<int> input, IntPredicate f){
31
        foreach(int i in input)
        {
            if (f(i))
36             yield return i;
        }
    }
  }

41 class Printer{

    static void Main(){
        var c = new SeqCollection(1, 10);
        var cc = HelperMethods.Filter(c, (x => x > 5));
46
        foreach(int i in cc){
            Console.WriteLine(i);
        }
    }
51 }
```

9 Map: another higher-order function

```
using System;
using System.Collections.Generic;
using System.Collections;
4
class SeqCollection : IEnumerable<int>
{
    private int myMin;
    9    private int myMax;

    public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
14        for(int i = myMin; i <= myMax; i++)
            yield return i;

    }

19    IEnumerator IEnumerable.GetEnumerator() {
        return null;
    }
}
24
static class HelperMethods{

    public delegate bool IntPredicate(int x);
    public delegate int IntOperator(int x);
29

    public static IEnumerable<int>
        Filter(IEnumerable<int> input, IntPredicate f){

        foreach(int i in input)
34        {
            if (f(i))
                yield return i;
        }
    }
39

    public static IEnumerable<int>
        Map(IEnumerable<int> input, IntOperator f){

44        foreach(int i in input)
            {
                yield return (f(i));
            }
    }
49
}

class Printer{
54    static void Main(){
```

```
var c = new SeqCollection(1, 10);
var cc = HelperMethods.Filter(c, (x => x > 5 ));
var ccc = HelperMethods.Map(cc, (x => x*x));

59  foreach(int i in ccc){
      Console.WriteLine(i);
    }
  }
```

10 Generalizing using generics

```
using System;
2 using System.Collections.Generic;
using System.Collections;

class SeqCollection : IEnumerable<int>
7 {
    private int myMin;
    private int myMax;

    public SeqCollection(int min, int max){myMin = min; myMax = max;}
12
    public IEnumerator<int> GetEnumerator(){
        for(int i = myMin; i <= myMax; i++)
            yield return i;
17
    }

    IEnumerator IEnumerable.GetEnumerator() {
        return null;
22
    }
}

static class HelperMethods{
27
    public delegate S Function<T,S>(T x);

    public static IEnumerable<T>
        Filter<T>(IEnumerable<T> input, Function<T, bool> f){
32
        foreach(T i in input)
        {
            if (f(i))
                yield return i;
        }
37
    }

    public static IEnumerable<S>
        Map<T,S>(IEnumerable<T> input, Function<T,S> f){
42
        foreach(T i in input)
        {
            yield return (f(i));
        }
47
    }
}

class Printer{
52
    static void Main(){
        var c = new SeqCollection(1, 10);
```

```
var cc = HelperMethods.Filter(c, (x => x > 5 ));
var ccc = HelperMethods.Map(cc, (x => x*x));
57 var cccc = HelperMethods.Map(ccc, (x => x.ToString()
    + "is_a_cool_number"));

foreach(string i in cccc){
62     Console.WriteLine(i);
    }
}
```

11 Rewriting with extension methods

```
1 using System;
  using System.Collections.Generic;
  using System.Collections;

6 class SeqCollection : IEnumerable<int>
  {
    private int myMin;
    private int myMax;

11 public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
        for(int i = myMin; i <= myMax; i++)
16         yield return i;
    }

    IEnumerator IEnumerable.GetEnumerator() {
21         return null;
    }
  }

  static class HelperMethods{
26 public delegate S Function<T,S>(T x);

    public static IEnumerable<T>
        Filter<T>(this IEnumerable<T> input, Function<T, bool> f){
31         foreach(T i in input)
            {
                if (f(i))
36                 yield return i;
            }
        }

    public static IEnumerable<S>
41         Map<T,S>(this IEnumerable<T> input, Function<T,S> f){

            foreach(T i in input)
            {
46                 yield return (f(i));
            }
        }

    }

51 class Printer{

    static void Main(){
        var c = new SeqCollection(1, 10)
```

```
56         .Filter(x => x > 5 )
           .Map(x => x*x)
           .Map(x => x.ToString() + "is_a_cool_number");

    foreach(string i in c){
61         Console.WriteLine(i);
    }
}
```


12 Replacing custom code with LINQ library calls

```
using System;
2 using System.Collections.Generic;
using System.Collections;
using System.Linq;

7 class SeqCollection : IEnumerable<int>
{
    private int myMin;
    private int myMax;

12 public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
        for(int i = myMin; i <= myMax; i++)
17         yield return i;
    }

    IEnumerator IEnumerable.GetEnumerator() {
22         return null;
    }
}

class Printer{
27
    static void Main(){
        var c      = new SeqCollection(1, 10)
                    .Where(x => x > 5 )
                    .Select(x => x*x)
32         .Select(x => x.ToString() + "is a cool number");

        foreach(string i in c){
            Console.WriteLine(i);
        }
37 }
}
```

13 Using special LINQ syntax

```
using System;
2 using System.Collections.Generic;
using System.Collections;
using System.Linq;

7 class SeqCollection : IEnumerable<int>
{
    private int myMin;
    private int myMax;

12 public SeqCollection(int min, int max){myMin = min; myMax = max;}

    public IEnumerator<int> GetEnumerator(){
        for(int i = myMin; i <= myMax; i++)
17         yield return i;
    }

    IEnumerator IEnumerable.GetEnumerator() {
22         return null;
    }
}

class Printer{
27
    static void Main(){
        var c = new SeqCollection(1, 10);
        var cc = from x in c
                where x > 5
32         select ((x*x).ToString() + "is a cool number");

        foreach(string i in cc){
            Console.WriteLine(i);
        }
37 }
}
```