Higher Order Programming in C# with LINQ

CIS399-005 Code Appendix

Jeff Vaughan

April 9, 2008

1 An enumerator for a constant sequence

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

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

    public TenEnumerator()
    {
        Reset();
    }

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


    public int Current
    {
        get { return myCurrent; }
    }

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

        myCurrent = myCurrent + 1;
        return true;
    }

    public void Reset()
    {
        myCurrent = 0;
    }

    public void Dispose() { }
}

class Printer
{

    static void Main()
    {
        var e = new TenEnumerator();

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

2 An enumerator for a sequence with settable range

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

class SeqEnumerator : IEnumerator<int>{

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

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

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


  public int Current{
   get{return myCurrent;}
  }

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

    myCurrent = myCurrent + 1;
    return true;
  }

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

  public void Dispose() {}

}

class Printer{

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

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

3 Lifting the enumerator to a collection

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

class SeqEnumerator : IEnumerator<int>{

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

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

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


  public int Current{
   get{return myCurrent;}
  }

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

    myCurrent = myCurrent + 1;
    return true;
  }

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

  public void Dispose() {}
}

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

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

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

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

```
class Printer{

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

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

4 Introducing yield

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

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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

class Printer{

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

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

5 Introducing foreach

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


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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

class Printer{

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

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

6 Defining a filter transformation by iterating

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


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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

static class HelperMethods{

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

  public static IEnumerable<int> FilterOdds(IEnumerable<int> input){

    foreach(int i in input)
    {
      if (IsOdd(i))
        yield return i;
    }

  }

}

class Printer{

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

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

7 Generalizing filter as a higher order function

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

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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;
  }

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

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); }

  public delegate bool IntPredicate(int x);

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

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

}

class Printer{

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

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

8 Calling filter with an anonymous lambda

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


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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

static class HelperMethods{

  public delegate bool IntPredicate(int x);

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

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

}

class Printer{

  static void Main(){
    var c = new SeqCollection(1, 10);
    var cc = HelperMethods.Filter(c, (x => x > 5));

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

9 Map: another higher-order function

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

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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

static class HelperMethods{

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

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

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


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

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

}

class Printer{

  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));

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

10 Generalizing using generics

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


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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

static class HelperMethods{

  public delegate S Function<T,S>(T x);

  public static IEnumerable<T>
          Filter<T>(IEnumerable<T> input, Function<T, bool> f){

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


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

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

}

class Printer{

  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));
    var cccc = HelperMethods.Map(ccc, (x => x.ToString()
                                              + " is a cool number"));

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

11 Rewriting with extension methods

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


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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

static class HelperMethods{

  public delegate S Function<T,S>(T x);

  public static IEnumerable<T>
          Filter<T>(this IEnumerable<T> input, Function<T, bool> f){

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


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

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

}

class Printer{

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

```
                        .Filter(x => x > 5 )
                        .Map(x => x*x)
                        .Map(x => x.ToString() + " is a cool number");

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

12 Replacing custom code with LINQ library calls

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


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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

class Printer{

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

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

13 Using special LINQ syntax

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


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

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

  public IEnumerator<int> GetEnumerator(){

    for(int i = myMin; i <= myMax; i++)
      yield return i;

  }

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

class Printer{

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

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