ProcessGenerators

For example this code:

```
list = ("one", "two", "three")
newlist = [item.ToUpper() for item as string in list]
```

is transformed into this:

```
[Boo.Lang.ModuleAttribute]
public final transient class TempModule(System.Object):

 private static def Main(argv as (System.String)) as System.Void:
  ___locals = TempModule.___locals2()
  ___locals.___list_0 = ('one', 'two', 'three')
  newlist = Boo.Lang.List(TempModule.___generator1(___locals))

 private def constructor():
  super()

 [Boo.Lang.EnumeratorItemTypeAttribute(System.String)]
 private final class ___generator1(Boo.Lang.AbstractGenerator):

  public virtual def GetEnumerator() as System.Collections.IEnumerator:
   return Enumerator(self._____locals4)

  internal _____locals4 as TempModule.___locals2

  public def constructor(_____locals4 as TempModule.___locals2):
   super()
   self._____locals4 = _____locals4

  class Enumerator(System.Object, System.Collections.IEnumerator,
System.ICloneable):

   internal _____locals3 as TempModule.___locals2

   public def constructor(_____locals3 as TempModule.___locals2):
    super()
    self._____locals3 = _____locals3
    self.Reset()

   protected ____enumerator as System.Collections.IEnumerator

   protected ____current as System.Object

   public virtual def Reset() as System.Void:
    self.____enumerator = self._____locals3.___list_0.GetEnumerator()

   public Current as System.Object:
    public virtual get:
```

```
      return self.____current

    public virtual def MoveNext() as System.Boolean:
     if self.____enumerator.MoveNext():
      item = self.____enumerator.get_Current()
      self.____current = item.ToUpper()
      return true
     return false

    public virtual def Clone() as System.Object:
     return self.MemberwiseClone()

  class ___locals2(System.Object):

   internal ___list_0 as (System.String)
```

```
    public def constructor():
     super()
```

So, thank Rodrigo for saving you all that typing 🙂