Variance

class Double<out T>({t1 : T, t2 : T}) {
    fun first() : T = t1;
    fun second() : T = t2;
}

x := Double<String>("Hello", "World");
fun append<T>({d : Double<Iterable<T>>}) :
    Iterable<T> = d.first() ++ d.second();
return [String(append<Character>(x))];

Anonymous Classes

interface Property<T> {
    fun holds(t : T) : Boolean;
}

fun lessThans(i : Int) : Iter<Prop<Int>> {
    iter := [];
    while (i > 0) {
        iter := iter ++ [new Prop<Int>() {
            fun hold(t : Int) : Bool = t < i;
        }];
        i := i - 1;
    }
    return iter;
}

Comprehensions

fun lessThans(i : Int) : Iter<Prop<Int>> = [for (j in 0..<i) lessThan(j)];
fun locations(i : Int, j : Int) : Double<Int> = [for (x in 0..<i)
    for (y in 0..<j)
        Double<Int>(x, y)];
Yields

```scala
class Append[T](f : Iter<T>, s : Iter<T>)
  extends Iterable<T> {
  yielder {
    for (t in f)
      yield t;
    for (t in s)
      yield t;
  }
}
```

Laziness

```scala```
fun sum(ints : Iter<Int>) : Int {
  sum := 0;
  for (int in ints)
    sum := sum + int;
  return sum;
}

infinity := sum(0…);
return [toString(0 * infinity)];
```

Assembly

ENTER AT YOUR OWN RISK