CS 4120
Introduction to Compilers

Ross Tate
Cornell University

Lecture 36: Extensions

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))];
```

2

Variance

```
interface Comparator<T> {
    fun lessEq(left : T, right : T) : Boolean;
}
class IntegerComparator()
    extends Comparator<Integer> {
    fun lessEq(left : Integer, right : Integer)
        : Boolean = left <= right;
}
Comparator<Object> c := IntegerComparator();
```

3

Anonymous Classes

```
interface Property<T> {
    fun holds(t : T) : Boolean;
}
fun lessThan(i : Integer) : Property<Integer>
    = new Property<Integer>() {
        fun holds(t : Integer) : Boolean
            = t < I;
    };
```

4

Anonymous Classes

```
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;
}
```

5

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)];
```

6

Yields

```
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;
      }
}
```

7

Laziness

```
fun sum(ints : Iter<Int>) : Int {
      sum := 0;
      for (int in ints)
            sum := sum + int;
      return sum;
}
infinity := sum(0…);
return [toString(0 * infinity)];
```

8

Assembly

9