sml2java

Haakon Larson • Justin Koser • Jeff Vaughan

Why SML?

In which the obscure functional language makes an appearance.

- SML is type-safe. Translated code retains these benefits.
- Translated code effectively simulates first order functions.
- SML concepts can be translated into familiar imperative syntax.

Why Java?

Here comes the Sun.

- Everyone knows Java
- Automatic garbage collection mirrors SML
- Objects ease translation of SML concepts
- Translated programs can call on Java’s libraries

Variables, Records and Tuples

In which variables that don’t vary become variables that do.

Variable declaration

```
val x = 312

Record r = (new Integer(312))
```

Record declaration

```
val r = {toast = "seven", waffle = 13}

Record r = ((new Record()).add("toast",new String("seven")))
                          .add("waffle", newString)
```

Tuples reduce to records

```
val t = ("seven", 13)

Record t = ((new Record()).add("1",new String("seven")))
                          .add("2", newString)
```

Functions and Closures

Where we will be in the next few weeks.

- A function's a function because it can be applied

```
public interface function
{ Record apply(Record r) }
```

- Function expressions translate to classes

```
fn(x) => <function body>

new function()
{ Record apply(Record r) {<translated function body>} }
```

- Function values become objects

```
val foo (x) =  <function body>

function foo = new function()
{ Record apply(Record r){<translated function body>} }
```

Conclusion

To inifinity and beyond!

- Future enhancements
  - User defined data-types
  - Pattern matching
  - Parametric polymorphism
- Uses
  - Teaching tool for COM S 312
  - Method for integrating provably stable code with Java