The Moama Functional Language Design and Implementation (and quite a bit about the Monto Disintegrated Development Environment)

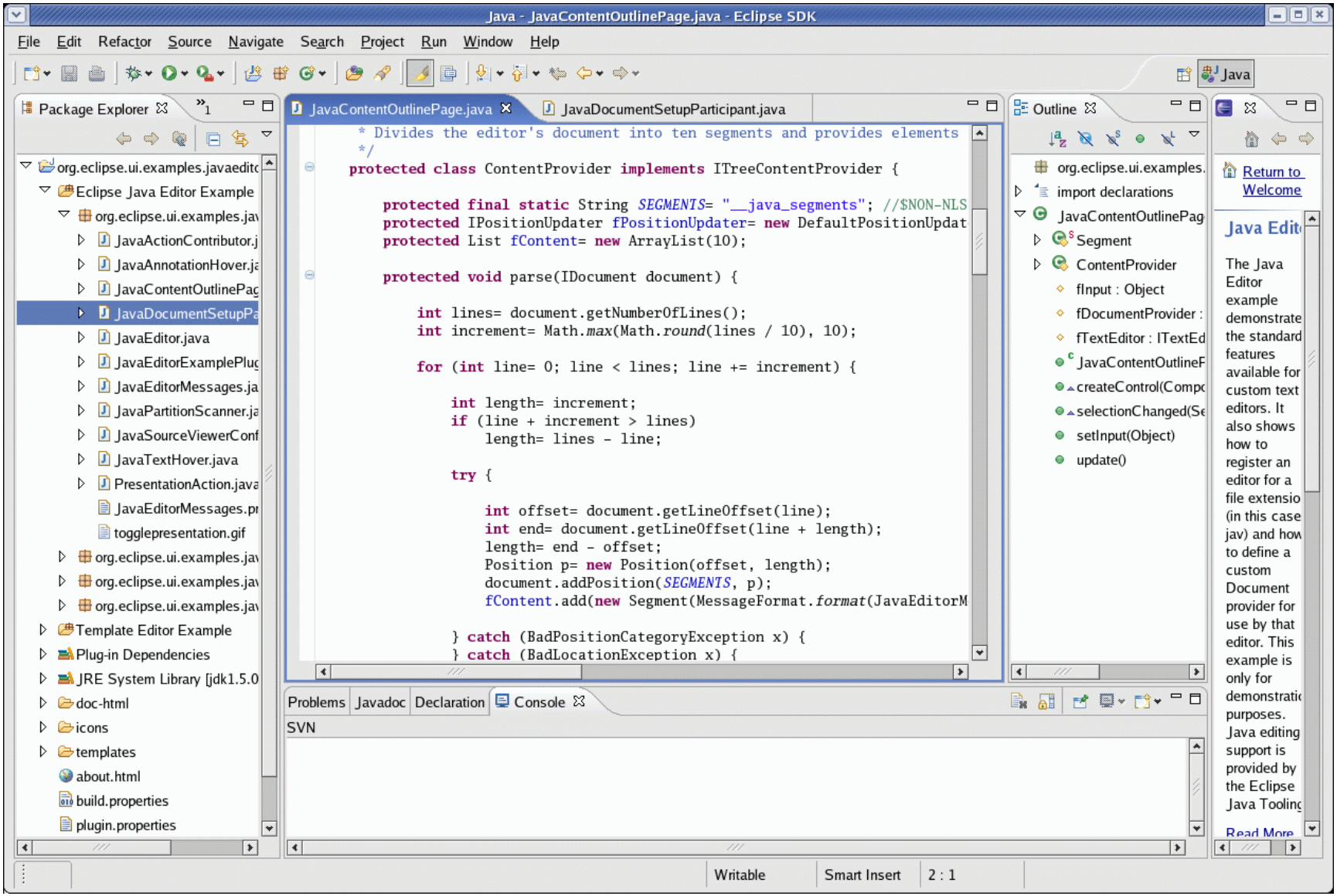
Anthony M. Sloane

Programming Languages and Verification Research Group
Department of Computing
Macquarie University
@plvmq

Anthony.Sloane@mq.edu.au
@inkytonik

MACQUARIE
University
SYDNEY·AUSTRALIA

Integrated Development Environments



Extending IDEs

- http://www.vogella.com/tutorials/EclipsePlugIn/article.html

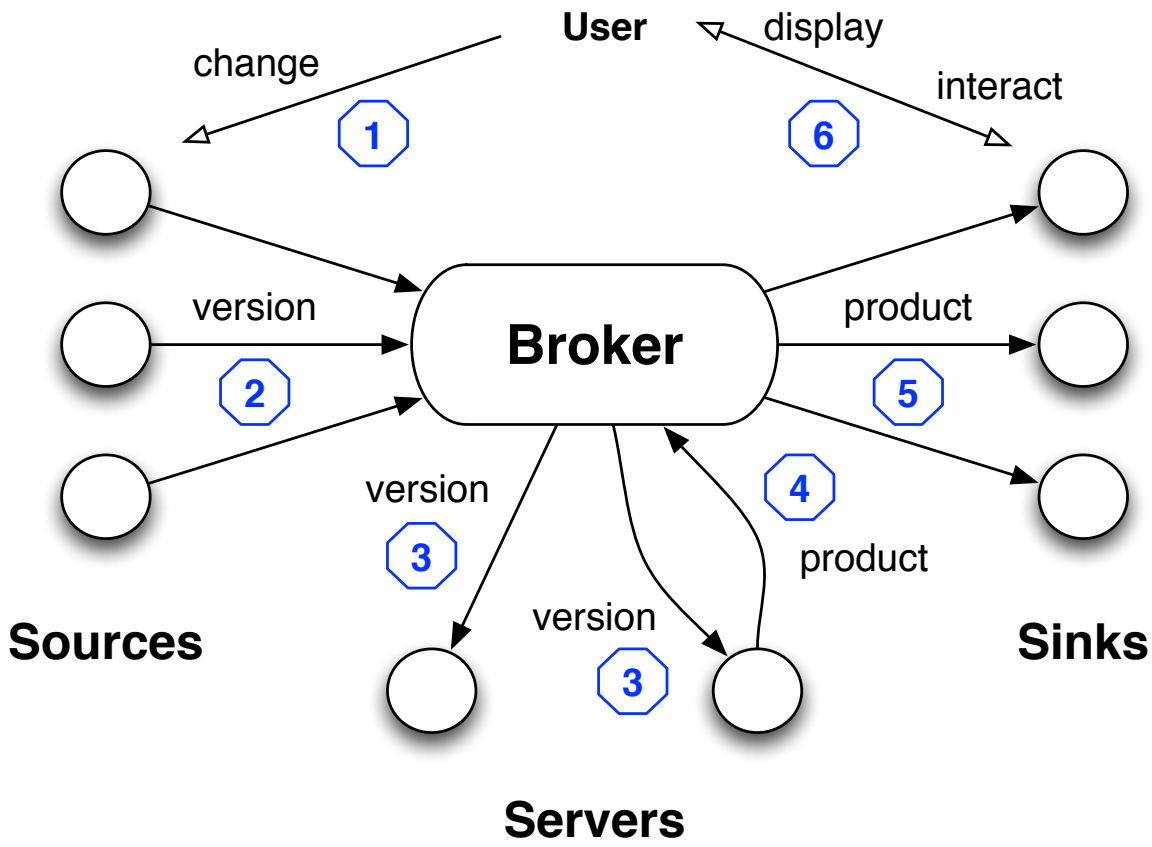
7. Exercise: Add a e4 menu and toolbar to the Eclipse IDE

- **7.1. Target of this exercise**
- **7.2. Creating a plug-in project**
- **7.3. Starting an Eclipse IDE with your plug-in**
- **7.4. Adding the plug-in dependencies for the e4 API**
- **7.5. Creating the handler class**
- **7.6. Creating a model contribution**
- **7.7. Adding a toolbar contribution**
- **7.8. Validating the presence of the menu and toolbar contribution**

Disintegrated Development Environments

- Joint work with Matt Roberts, Scott Buckley, Shaun Muscat
- Inspiration
  - Difficulty of integrating new functionality into established IDEs
  - Editor-based approaches to language-specific support
  - Work on tool integration: e.g., ToolBus, Linda, ENSIME
- Philosophy
  - Simplify, simplify, simplify
  - Separate components as much as possible
  - Text is the common denominator
- Monto
  - Python-based infrastructure
  - Simple JSON messages sent using ZeroMQ
  - Front-ends: Sublime Text 3 (Macquarie), Eclipse (TU Darmstadt)
  - Web-based experiments

Monto Architecture



SublimeMonto plugin

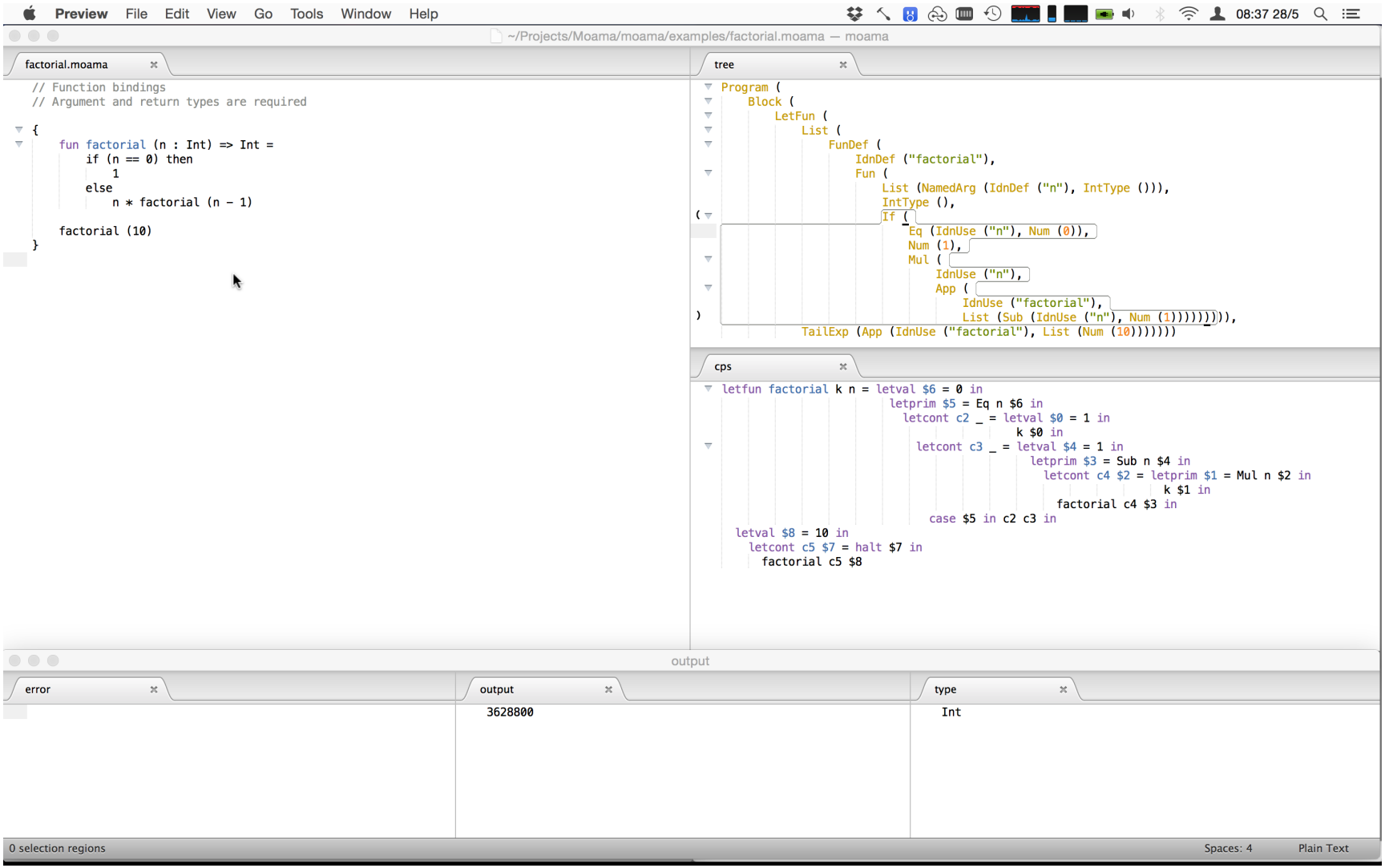
- Extends Sublime Text 3
- Source
  - A version is published each time a "change" happens in a file view
  - Changes include opening, focussing, typing, and moving selection
- Sink
  - Users interactively create views on products
  - Product views are updated when new products arrive
- In the works:
  - Two-way mapping between source and product views

Moama

- Simple, strict, pure functional language
  - Scala-inspired syntax, ML-inspired semantics
  - Translation to continuation-passing style (CPS)
  - Evaluate in batch mode, via REPL or using Monto
  - Missing lots of stuff, including
    - user-defined types
    - input/output
- Implementation in Scala
  - About 3000 lines of code
  - Parsing using sbt-rats parser generator
  - Uses Kiama language processing library
    - rewrite rules for desugaring
    - attribution for name and type analysis
    - pretty-printing
  - Monto server wrapper

Demo

Demo: SublimeMonto while editing Moama program



Demo program: Simplest

```
// A program is an expression
// Int and Bool basic types

42
// 31 + 11
// true
// false || true
// 5 <= 10
```

Demo program: Values

```
// Blocks contain definitions and
//   one final expression
// Value definitions have inferred types
// Values are visible to end of scope (let)

{
    val x = 1
    // val z = y
    val y = x + 1

    x
    // z
    // { val x = 2 x * 3 }
    // x + y
    // x + y * { val z = 3 y + z }
}
```

Demo program: Factorial

```
// Function bindings
// Argument and return types are required

{
    fun factorial (n : Int) => Int =
        if (n == 0) then
            1
        else
            n * factorial (n - 1)

    factorial (10)
}
```

Demo program: Lambda expressions

```
// Return type is inferred
// Partial application is allowed
// Over-application is not allowed
// How to print functions?

fun (x : Int) = x + 1
// (fun (x : Int) = x + 1) (42)
// (fun (a : Int, b : Int) = a + b) (4, 5)
// (fun (a : Int, b : Int) = a + b) (4, 5, 6)
// (fun (a : Int, b : Int) = a + b) (4) (5)
// (fun (a : Int, b : Int) = a + b) (4)
```

Demo program: First-class functions

```
{
    fun twice (f : (Int) => Int, x : Int) => Int =
        f (f (x))

    fun add (a : Int) => (Int) => Int =
        fun (b : Int) = a + b

    // twice
    // add
    // add (2)
    // add (2, 3)
    // add (2) (3)
    // twice (add (2))
    // twice (add (2), 3)
}
```

Demo program: Mutually recursive functions

```
// Adjacent function definitions form a letrec

{
    fun even (n : Int) => Bool =
        if (n == 0) then
            true
        else
            odd (n - 1)

    fun odd (n : Int) => Bool =
        if (n == 0) then
            false
        else
            even (n - 1)

    even (1670)
}
```

Questions?

- Moama
  - `bitbucket.org/inkytonik/moama`
- Disintegrated Development Environments
  - Monto: `bitbucket.org/inkytonik/monto`
  - SublimeMonto: `bitbucket.org/inkytonik/sublimemonto`
  - Sublime Text: `www.sublimetext.com/3`
- Software Language Engineering
  - Kiama: `kiama.googlecode.com`
  - sbt-rats: `bitbucket.org/inkytonik/sbt-rats`
- Twitter
  - @plvmq
  - @inkytonik