What Mike is doing in Hawai‘i
• Four papers:
  – Inference of field initialization
  – Building and using pluggable type-checkers
  – Always-available static and dynamic feedback
  – Speculative program analysis

• Plus more UW papers:
  – Identifying program, test, and documentation changes that affect behavior
Inference of field initialization

```java
public class MyWindow extends JWindow {
    private final String name;
    public MyWindow(String name) {
        super();
        this.name = name;
    ...
    
    Accuracy >98% (by far the world’s best)
```
Pluggable type-checking

**Problem:** Even if the type-checker succeeds, your program can still crash
  
  Example: null pointer exception

**Idea:** create optional, stronger type systems

**Tool:** the Checker Framework

**Results:**
- Finds lots of real bugs
- Little annotation overhead
- Easy to get started using
- Easy to build new type systems
Complementary verification technologies

Static type-checking is useful
not always the most important goal

Dynamic testing is useful
not always the most important goal

Idea: let the programmer choose the best approach, at any moment during development

– Fast, flexible development, as with dynamic types
– Reliable, maintainable applications, as with static types
Program analysis

Informs you about your program
• Type-checking
• Testing
• Profiling
• Bug-finding
• Verification
• Collaboration

Idea: run program analysis on programs you have not yet written