What is Java Modeling Language (JML)?

```java
public class Counter {
    public final static int MAX = 100;

    //@ invariant 0 <= count && count <= MAX;
    private int count;

    //@ requires count < MAX;
    //@ ensures count == old(count) + 1;
    //@ also
    //@ requires count == MAX;
    //@ ensures count == 0;
    //@*@
    public void inc() {
        count = count < MAX ? count + 1 : 0;
    }
}
```
Project Motivation

- Find JML tool(s) that we can use at work
  - Easy to use
  - Robust and fully developed
- Project plan
  - Survey all tools
  - In depth analysis of most promising tools
  - Introduce best of breed to work

Types of JML Tools

- **Run time checking** – Tests for violations of the JML assertions as Java code is executed
- **Static checking** - checking annotations prior to execution
  - **Automatic** – little developer interaction
  - **Manual** – Programmer provides proof for more sound and complete evaluation
Overview of JML tools

Runtime Tools
- JML2
- JML3
- JML5

Static Tools
- ESC/java
- LOOP
- JACK

Legend
- Inactive
- Active
- Future

JML4

JML4c

ESC/java2

FSPV

JMLEclipse

jml4c

ESC4

FSPV

JML4
JML4

JML checks for very many things. These include:

- Null pointer dereference
- Negative array index
- Array index too large
- Invalid type casts
- Array storage type mismatch
- Negative array size
- Unreachable code
- Unchecked code
- Unchecked exception

Object invariant broken
Divide by zero
Deadlock in concurrent code
Race condition
Loop invariant broken
Precondition not satisfied
Postcondition not satisfied
Assertion not satisfied

JML annotations and assertions can help with all of these.

Evaluation of the JML4

- Installed the tools along with several plugins to understand set-up constraints
- Ran code examples and several small modules to test tool checking capabilities
- Performed usability tests with experienced developers to examine issues with using the tools in a development setting
Tool Conclusions

- Tools not commercially viable; currently they are research tools
  - Tools have difficulty keeping up with changing language features
  - Usability issues - difficult to install and use
- Bottom line the tools are not actively marketed commercially – Academically driven w/o corporate sponsorship

Is JML useful?

- Assuming a production ready JML tool: Would you use JML?
  - Hard to identify invariant pre/post conditions
  - Adding specifications as complex as coding
  - No published work showing results of testing on industrial scale code
- But, probably only cost- effective for “mission critical” development
Questions

Significant References


