An introduction to formal specifications and JML

Concluding remarks

Yves Ledru
Université Grenoble-1
Laboratoire d'Informatique de Grenoble
Yves.Ledru@imag.fr

2013

Missing elements

- More should be said about:
  - How JML handles inheritance
  - How to specify interfaces
  - How to specify interfaces of classes with a hidden state
  - How to support different abstraction levels in specifications
- Moreover JML provides its own implementation of some data structures (e.g. JMLcollection)

Quality vs cost

- JML assertions are a way to improve quality of java programs, by associating them to precise specifications.
- This comes at the cost of writing specifications: should we write the same things twice (in Java and JML)?
  - No, if the JML spec is too close to the Java code
  - Yes if they are at different abstraction/complexity levels
  - Yes if you write a property once (e.g. invariant) and it is checked at lots of places in the program!

Multi-person developments

- Most successful sofware are built by several distinct developers:
  - Either working simultaneously (development team)
  - Or working on subsequent versions in time
- It is difficult to make sure that everybody is aware of the expected properties of the software or its data structures.
- JML assertions provide an active documentation of these properties, which can be checked by a conventional testing activity.

Subcontractors

- The JML specification provides a precise reference which can be unambiguously checked by both parties!
- When a problem arises:
  - It means that specification and code disagree
  - It is often easier to decide whether the code or the spec is wrong.

Difficulties in the expression of a specification

- Find the right abstraction level for the specification, in order to keep implementation freedom.
- Write cost-efficient specifications, e.g. write invariants and history constraints which apply to all methods of the class.
- An objective could be to produce 5 to 10% of JML code in the program. (this shows that everything was not written twice!)

Use assertions and contract-based specifications!

- JML 5.6 is old, but it is a nice language to illustrate numerous constructs for contract-based specifications.
- Other languages support the same approach, for Java and for other programming languages (e.g. Code Contracts for C#)
- It is also possible to use the Java assert mechanism, but it makes it more difficult to express invariants, or to refer to the old state.
- Let us hope that the lightweight specification mechanisms of JML will eventually find their way in the daily practice of software engineers!