Software Testing, Quality Assurance and Maintenance Winter 2017

Lecture 34 — March 29, 2017

Patrick Lam version 1

JML

The Java Modelling Language allows developers to specify structured design-by-contract properties for Java code. In particular, you can specify preconditions and postconditions for methods as well as class invariants. These are all in a Java-like expression language. It is possible to verify these predicates either statically or dynamically.

Static verification. ESC/Java2 is a research tool that reads JML specifications and enforces them statically using theorem-proving technology. Here's some sample code that ESC/Java2 can verify.

```
1  class Bag {
2    int[] a; //@ invariant a != null;
3    int   n; //@ invariant 0 <= n && n <= a.length;
4
5    //@ requires n > 0;
6    int extractMin() {
7      int m = Integer.MAX_VALUE;
8      int mindex = 0;
9      for (int i = 0; i < n; i++) {
10       if (a[i] < m) { mindex = i; m = a[i]; }
11     }
12     n--;
13     a[mindex] = a[n];
14     return m;
15   }
16 }
```

ESC/Java2 reads the Java code and produces verification conditions for the theorem prover, which in turn uses a backtracking search to assign values to variables. This tool thus statically guarantees that the code satisfies its specifications. It's also possible to check these properties at run-time by adding assertions to the code.

Dynamic verification. It's easier to perform dynamic verification for many JML specifications. In the above example, it suffices for the compiler to insert an assertion that `n > 0` upon entry to `extractMin`, as well as assumptions of the invariants. Then upon exit, the compiler inserts assertions for any `ensures` clauses as well as the invariants. So we know that the invariant always holds when the class is not actively working on updating its state.