



Tutorials for “Formal methods for Java” Exercise sheet 2

Exercise 1: Operational semantics

Consider the following Java class:

```
class C {  
    private boolean b = true;  
    public int m(int x) {  
        return (b = !b) ? x-1 : x+1;  
    }  
}
```

Use the rules defining the operational semantics of Java to compute the result of the method call: $c.m(4)$. Assume that c is an instance of class C which has just been initialized.

Exercise 2: Loops with breaks

Java provides the **break** statement that when executed within a loop causes the execution of the loop to be stopped immediately. Execution is then continued with the first statement after the corresponding loop. We can model **break** statements by extending the flow component of program states:

$$Flow ::= Norm | Ret | Exc \langle \langle Address \rangle \rangle | Break .$$

Use this extension to define the operational semantics of **break** statements and **while** loops with breaks.

Exercise 3: Operational equivalence

We say that two Java statements c_1 and c_2 are operationally equivalent if

$$\forall flow, heap, lcl, flow', heap', lcl'. (flow, heap, lcl) \xrightarrow{c_1} (flow', heap', lcl') \iff (flow, heap, lcl) \xrightarrow{c_2} (flow', heap', lcl')$$

Are the following pairs of Java statements operationally equivalent? Give a proof or a counter-example.

- (a) **if**(e) c_1 **else** c_2 and **if**($!e$) c_2 **else** c_1
- (b) **if**(e) c **else** c and c
- (c) **while**(e) c and **while**(**true**) {**if**(e) c **else break**}
(use your rules from Exercise 2)