Prof. A. Podelski, **Prof. P. Thiemann**,
M. Mehlmann and **S. Wehr**

Summer term 2008

Software Engineering

http://swt.informatik.uni-freiburg.de/node/94
http://proglang.informatik.uni-freiburg.de/teaching/swt/2008/

Exercise Sheet 3

2008-05-23

Exercise 1 (Merging of linksets; 2 Points)
Given the two linksets

$$L_1 \equiv x : \texttt{int} \mid (b \approx y : \texttt{int} \vdash x > y : \texttt{bool}), (y \approx \emptyset \vdash 5 : \texttt{int})$$
$$L_2 \equiv b : \texttt{bool}, z : \texttt{int} \mid (x \approx \emptyset \vdash \texttt{if } b \texttt{ then } z \texttt{ else } 0 : \texttt{int})$$

Merge L_1 and L_2; that is, compute $L_1 + L_2$.

Exercise 2 (Linking; (3+3) Points)
(a) Link the following linkset L; that is, execute link steps $\rightsquigarrow$ as long as possible.

$$\begin{aligned} L \equiv z : \texttt{int} \mid\ & (b \approx y : \texttt{bool}, x : \texttt{int} \vdash \texttt{if } y \texttt{ then } x \texttt{ else } z : \texttt{int}) \\ & (y \approx x : \texttt{int} \vdash x > 5 : \texttt{bool}) \\ & (x \approx \emptyset \vdash 6 : \texttt{int}) \end{aligned}$$

(b) Show that the link step relation $\rightsquigarrow$ does not preserve the intra-checked property. That is, find a linkset L with *intra-checked*(L), $L \rightsquigarrow L'$, but not *intra-checked*(L').

Exercise 3 (Interfaces for Featherweight Java; 12 points)
Extend Featherweight Java with interfaces. To help you getting started, here is the syntax of the extended language:

$$\begin{aligned} CL ::=\ & \textbf{class } C \textbf{ extends } D \textbf{ implements } E_1, \dots \ \{C_1\ f_1; \dots\ K\ M_1 \dots\} \\ & \mid \textbf{interface } C \textbf{ extends } D_1, \dots \ \{S_1; \dots\} \\ S ::=\ & C\ m(C_1\ x_1, \dots) \end{aligned}$$

(K, M, t, and v are defined as in the lecture.)

We use the metavariables C, D, and E to range over class *and* interface names. A class declaration **class** C **extends** D **implements** $E_1, \dots\ \{C_1\ f_1; \dots\ K\ M_1 \dots\}$ specifies, in addition to the superclass C, the interfaces $E_1, \dots$ that C implements. It is possible that the sequence $E_1, \dots$ is empty; in this case, C does not implement any interfaces.

An interface declaration **interface** C **extends** $D_1, \dots\ \{S_1; \dots\}$ introduces a new interface C. The sequence $D_1, \dots$ (which may be empty) specifies the superinterfaces of D.

The metavariable S ranges over method signatures. A method signature only gives the return and the argument types of a method; it does not define the method body.

You must now extend the typing rules and, if necessary, the operational semantics of Featherweight Java to support interfaces. Use your experience with interfaces in Java when designing the extension. You should keep your extension as small as possible by reusing the rules presented in the lecture. (Note: in these rules, class declarations do not specify the interfaces that a class implements. Nevertheless, you can reuse these rules by assuming that these interfaces are $E_1, \ldots$, where $E_1, \ldots$ are names that are not used anywhere else in the rule.)

Submission: 2008-05-30, 12pm **before** the exercise session in HS 00-036, building 101.