Formal Methods in Software Development
Exercise 4 (December 7)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

November 18, 2009

The result is to be submitted by the deadline stated above via the Moodle interface as a .zip or .tgz file which contains

- A PDF file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - for each exercise, a section with the number and name of the exercise, the JML-annotated Java code, and a copy of the output of an `escjava2` check of that code,
  - optionally any explanations or comments you would like to make;
- the JML-annotated Java files developed in the exercise.

6a (50 points): JML function specifications

Annotate the functions in class `Exercise4a` with JML header specifications that are as expressive as possible.

Type-check the specifications with `jml -Q` (which must not give an error) and statically check them with `escjava2` (which may or may not give warnings).

Please take care to constrain the arguments by appropriate preconditions. The fact that above functions d not modify the content of argument a can be specified by the JML clause `assignable \nothing`.

6b (40 points): JML procedure specifications

Annotate the functions in class `Exercise4b` with JML header specifications that are as expressive as possible.

Type-check the specifications with `jml -Q` (which must not give an error) and statically check them with `escjava2` (which may or may not give warnings).

Please take care to constrain the arguments by appropriate preconditions. The fact that above functions modify the content of argument a can be specified by the JML clause `assignable a[*]`.

6c (10 points): JML exception specification

Annotate the funciton in class `Exercise4c` with a *heavy-weight* JML header specification that is as expressive as possible. Type-check the specification with `jml -Q` (which must not give an error) and statically check it with `escjava2` (which may or may not give warnings).

Warning

One of the Java functions has a (small) error. Demonstrate with the help of `escjava2` the error, fix it, and show the output of `escjava2` on the new code.

Hints

Please note that no tool can prove that a specification is *adequate* i.e. correctly describes the informal intention; please apply common sense in writing your specifications (it may be wise to check that examples of correct input/output pairs meet the specification and that examples of incorrect input/output pairs violate the specification, e.g. using the JML compiler/runtime assertion checker `jmlc`/`jmlrac`).

Furthermore please note that the fact that `escjava2` does not report an error does not prove that the function indeed satisfies the specification (only that the tool could not find a violation); on the other hand, if `escjava2` reports a warning, this does not necessarily mean that the program indeed violates its specification (only that the tool could not verify correctness).