Formal Methods in Software Development
Exercise 8 (January 16)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

December 4, 2011

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

1. a PDF file with
   - a cover page with the course title, your name, Matrikelnummer, and email address,
   - the deliverables requested in the description of the exercise,
2. the JML-annotated Java files developed in the exercise,

Email submissions are *not* accepted.

6a (50 points): A Private JML Class Specification

Take the attached source code of a class `BoundedQueue` which implements a bounded queue of integers and extend it by a *private* specification in the *heavy-weight* JML format that is as expressive as possible.

Use `jml -Q` to check the specification (which must not yield an error). Run `escjava2` on the specification. If the tool gives warnings, check them, and *if you are very confident* that everything is fine, insert the *minimal* set of `//@nowarn Post` (respectively `Invariant` or `Post, Invariant`) annotations required to switch them off.

However, give an explicit explanatation/interpretation of the warnings and why you felt justified to switch them off.

The result of this exercise contains the JML-annotated file `BoundedQueue.java` and the output of `jml -Q` and `escjava2` on this file (in the last case, once without and once with the `nowarn` annotations).

6b (50 points): A Public JML Class Specification

Take the previously JML-annotated file `BoundedQueue.java` and modify it for an appropriate *public* specification of class `BoundedQueue`; this public specification is to be written into file `BoundedQueue.jml` and shall be based on the abstract datatype `QueueModel` specified in the attached file `QueueModel.java`. Some hints:

- The basic strategy is the same as shown in class for the model-based public specification of class `IntStack`.
- Introduce in `BoundedQueue.jml` a model field of type `QueueModel` which receives its value from a model function `toModel()`.
- Give in `BoundedQueue.jml` public specifications of the public functions using the model field and the corresponding operations on `QueueModel`.
- Annotate `BoundedQueue.java` by a `refines` annotation that indicates that the definition of class `BoundedQueue` in this file is a refinement of the class declared in `BoundedQueue.jml`. Add the keyword `also` to the private behavior specifications of all public methods.
- Give a specification-only definition of the abstraction function `toModel` as

```
/*@ public pure model QueueModel toModel() {
  @   QueueModel q = new QueueModel();
  @   int index = head;
  @   for (int i=0; i<count; i++)
  @   {
  @     q = q.enqueue(a[index]);
  @      index = index+1;
  @      if (index == a.length) index = 0;
  @    }
  @    return q;
  @ }
  @*/
```

Annotate this definition with a *private* behavior specification that relates the constructed `QueueModel` to the current `BoundedQueue` object.

- Add the private object variables to the data group of the model variable; thus whenever an assignment on the model variable in the public specification is allowed, also an assignment to the private variables in the implementation is allowed.

First use `jml -Q` to type-check `BoundedQueue.jml` in a directory that contains also `BoundedQueueModel.java` but does *not* contain `List.java` (otherwise also this file will be immediately type-checked). As soon as the type-check succeeds, also add the file `BoundedQueue.java` from the previous exercise to this directory and extend it as indicated above.

Now use `jml -Q` again to type-check the files. As soon as everything is fine, try `escjava2` which may complain again. Check these warnings; if you are confident that everything is fine, turn them off by `nowarn` annotations.

However, give an explicit explanatation/interpretation of the warnings and why you felt justified to switch them off.

The exercise result contains the files `BoundedQueue.jml`, `BoundedQueue.java`, and also `QueueModel.java`, and the output of `jml -Q` and `escjava2` (in the last case, once without and once with the `nowarn` annotations).