Formal Methods in Software Development
Exercise 1 (April 21)

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

March 14, 2005

The exercise is to be submitted by **April 21** (hard deadline)

1. either as a single paper report (cover page with full name and Matrikelnummer, pages stapled) which is handed out to me in class,
2. or as a single PDF file sent to me per email.

Questions can be asked per email or in the class on April 14 latest.

1 Array

Take the attached file `Array.java` and write a small test program `Main` in the style of the program presented in class that allows to test the method `subarray(a,i)` with arrays of various lengths.

`Main` should check the command line arguments such that they do not let the program crash in any case.

Annotate the methods in `Array` and the auxiliary methods in `Main` with lightweight specifications (method contracts) that as strongly as possible capture the expected behavior of the methods. As a minimum, make sure that `escjava2` does not complain on these files. Please note that `Array` contains a bug that has to be fixed for this purpose.

Compile the test program with the runtime assertion checking tool `jmlc` and let it run with `jmlrac`.

As a result of this exercise, deliver

1. the source of `Main.java` and of the JML annotated (buggy) `Array.java`;
2. the output of an execution of `Main` with the buggy class `Array` using `jmlrac` such that an assertion exception demonstrates the bug;
3. the output of `escjava2` on `Main` and the buggy `Array`;

4. the source of the JML annotated `Array.java` after fixing the bug;
5. the output of a correct execution of `Main` with `jmlrac`;
6. the output of of `escjava2` on `Main` and the correct `Array`.

2 Integer Queue

An *integer queue* is an abstract datatype Q with operations n, e, d, i, f (nil, enqueue, dequeue, isnil, first) obeying the following laws, for $q, q' \in Q, j, j' \in \mathbb{Z}$:

- $n \neq e(j, q)$.
- $e(j, q) = e(j', q') \Rightarrow j = j' \wedge q = q'$.
- $i(n) = \text{true}, i(e(j, q)) = \text{false}$.
- $d(e(j, n)) = n, d(e(j, e(j', q))) = e(j, d(e(j', q)))$.
- $f(e(j, n)) = j, f(e(j, e(j', q))) = f(e(j', q))$.

The attached file `Queue.java` contains a Java class that implements a queue. Specify the private behavior of this class as strongly as possible; as a minimum `escjava2` shall not complain. Please note that `Queue` contains a bug that has to be fixed for this purpose.

Write a program `Main` that tests the queue in a simple way.

Compile the test program with the runtime assertion checking tool `jmlc` and let it run with `jmlrac`.

Then also specify the public behavior of the class in a JML specification file `Queue.jml` using a model type `QueueModel`.

As a result of this exercise, deliver

1. the source of `Main.java` and of the JML annotated (buggy) `Queue.java` specifiying the private behavior;
2. the output of an execution of `Main` with the buggy class `Queue` using `jmlrac` such that an assertion exception demonstrates the bug;
3. the output of `escjava2` on `Main` and the buggy `Queue`;
4. the source of the JML annotated `Queue.java` after fixing the bug;
5. the output of a correct execution of `Main` with `jmlrac`;
6. the output of of `escjava2` on `Main` and the correct `Queue`.
7. the source of the corrected `Queue.java`, `Queue.jml` and `QueueModel.java` specifying the public behavior and the output of `escjava2` on these files.

Exercise Bonus (25%): implement (the methods in) `QueueModel` such that runtime assertions can be generated from the public behavior specification. Use for this purpose the JML class `org.jmlspecs.models.ObjectSequence` that implements an unbounded sequence of objects (you can encapsulate `int` values as `Integer` objects before putting them into such a sequence). Remove the specification of the private behavior from `Queue` and demonstrate by the use of `jmlc` and `jmlrac` that an assertion exception corresponding to the public behavior specification is triggered by the buggy implementation of `Queue`.