Formal Methods in Software Development
Exercise 6 (December 12)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

November 18, 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.

Exercise 6: JML Specifications

Formalize the following method specifications in the JML *heavy-weight* format by a precondition (`requires`), frame condition (`assignable`), and postcondition (`ensures`) and attach the specification to the method implementations provided in file `Exercise6.java`. For this purpose, extract the implementation of each method into a separate class `Exercise6_`*I* (where *I* is the number of the method in the list below) and give this class a `main` function that allows you to test the implementation by some calls of the corresponding method.

For each method, first use `jml` to type-check the specification. Then use the runtime assertion compiler `jml4c` and the corresponding execution script `jml4crun` to validate the specification respectively implementation by at least 3 calls of each method; the calls shall include valid and (if possible) also invalid inputs. Finally use the extended static checker `escjava2` to further validate your implementation.

Please note that various of the given specifications/implementations may contain ambiguities or even errors. If you detect such, explain them, fix them and re-run your checks.

The deliverables of this exercise consist

- a nicely formatted copy of the JML-annoated Java code for each class,
- the output of running `jml -Q` on the class,
- the output of running `jml4crun` on the class,
- the output of running `escjava2` on the class,

both for the original and for the modified implementation of the method (if the implementation was modified) including an explanation of the detected error and how you fixed it.

Please note that the fact that `escjava2` does not give a warning 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 its correctness).

1. Specify the method

   ```
   public static int maximumPosition1(int[] a)
   ```

   that takes a non-null and non-empty integer array a and returns the position of the largest element in the array.

2. Specify the method

   ```
   public static int maximumPosition2(int[] a)
   ```

   that takes an integer array a and returns the position of the largest element in the array (-1, if the array is null or empty).

3. Specify the method

   ```
   public static int maximumElement(int[] a)
   ```

that takes an integer array a and returns the largest element in the array.

4. Specify the method

   ```
   public static int[] insert(int[] a, int p, int n, int x)
   ```

   that takes a non-null array a, a position p and a length n and returns a copy of a with n copies of value x inserted at position p (n may be also 0 and elements may be also inserted at the beginning/end of a).

5. Specify the method

   ```
   public static int[] remove(char[] a, char x)
   ```

   that takes a non-null array a and returns an array that is identical to a except that all occurrences of value x have been removed.

6. Specify the method

   ```
   public static boolean substitute(int[] a, int[] p, int[] x)
   ```

   that takes an array a and an array p of different positions. The function modifies a by replacing the value at position $p[i]$ by $x[i]$. If some value of p is detected not to be a position in a, then this value is ignored and the function returns `false`; otherwise the function returns `true`.

7. Specify the method

   ```
   public static void substitute2(int[] a, int[] p, int[] x)
     throws InvalidArgument
   ```

   that takes an array a and an array p of different positions. The function modifies a by replacing the value at position $p[i]$ by $x[i]$. If some value of p is detected not to be a position in a, then the function throws an exception that contains the violating value; this value and all subsequent ones are ignored.