Formal Methods in Software Development
Exercise 3 (December 3)

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

October 14, 2007

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 (.java) files developed in the exercise.

3a(all): JML function specifications

Write a Java class `Exercise3a` that implements the following functions:

```
// returns minimum of non-empty array a
static int minimum(int[] a)

// returns concatenation of a and b
static int[] append(int[] a, int[] b)

// returns array derived from a by removing a[p..p+l]
static int[] cut(int[] a, int p, int l)
```

In all three cases the input arguments are not modified.

Annotate the functions 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.

3b(all): JML procedure specifications

Write a Java class `Exercise3b` that implements the following procedures:

```
// replaces in a every occurrence of x by y
static void replace(int[] a, int x, int y)

// overwrites a[p..p+n-1] by b[0..n-1] where n is the length of b
static void paste(int[] a, int[] b, int p)

// updates a by the assignments a[p[i]] := v[i] for every index i in p
// p does not contain the same value in different positions
static void update(int[] a, int[] p, int[] v)
```

In all three cases the content of array a is modified.

Annotate the procedures 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).

Hint: the fact that above functions modify the content of argument a can be specified by the JML clause `assignable a[*]`.

3c(all): JML exception specifications

Write a Java class `Exercise3c` that implements the following procedure:

```
// updates a by the assignments a[p[i]] := v[i] for every index i in p
// raises an IllegalArgumentException, if p holds the same value
// in different positions
static void update(int[] a, int[] p, int[] v)
  throws IllegalArgumentException
```

Annotate the procedures 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).

Hint: the exception class `IllegalArgumentException` is part of the Java standard API; a corresponding exception ccan be raised by the Java command `throw new IllegalArgumentException("`*`message`*`")`.