ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

J. Hoenicke
J. Christ

18.01.2012
Hand in solutions via email to
christj@informatik.uni-freiburg.de
until 25.01.2012 (only Java sources, KeY
proofs, and PDFs accepted).
Paper submissions possible after the lecture.

Tutorials for "Formal methods for Java"
Exercise sheet 10

Exercise 1: Jahob Syntax
Translate the JML annotations in the following program into Jahob syntax.

```
class IntKey {
    int value;
}

public class BubbleSort {
    /*@ requires arr.length > 0 && (\forall int i; i >= 0 && i < arr.length; arr[i] != null);
      @ ensures (\forall int k, l; 0 <= k && k <= l && l < arr.length;
      @                            arr[k].value <= arr[l].value);
      @*/
    public void sort(/*@ non_null */ IntKey[] arr) {
        /*@ loop_invariant i >= 0 && i < arr.length;
          @ loop_invariant (\forall int i; i >= 0 && i < arr.length; arr[i] != null);
          @ loop_invariant (\forall int k, l; i <= k && k <= l && l < arr.length;
                                arr[k].value <= arr[l].value);
          @ loop_invariant (\forall int k, l;
          @              0 <= k && k <= i && i < l && l<arr.length;
          @              arr[k].value <= arr[l].value);
          @
          @*/
        for (int i = arr.length-1; i > 0; i--) {
            /*@ loop_invariant i >= 0 && i < arr.length;
              @ loop_invariant (\forall int i; i >= 0 && i < arr.length; arr[i] != null);
              @ loop_invariant j >= 0 && j <= i;
              @ loop_invariant (\forall int k, l; i <= k && k <= l && l < arr.length;
              @                                   arr[k].value <= arr[l].value);
              @ loop_invariant (\forall int k, l;
              @              0 <= k && k <= i && i < l && l<arr.length;
              @              arr[k].value <= arr[l].value);
```

```
            @ loop_invariant (\forall int k; 0 <= k && k < j;
            @                                arr[k].value <= arr[j].value);
            @*/
          for (int j = 0; j < i; j++) {
            if (arr[j].value >= arr[j+1].value) {
              IntKey tmp = arr[j];
              arr[j] = arr[j+1];
              arr[j+1] = tmp;
            }
          }
        }
    }
}
```

Exercise 2: Sets, Lambda Functions, and Closure

JML uses special model classes to reason about sets while Jahob has builtin support for these operations. Consider a binary tree implementation that has nodes of the following type:

```
class TreeNode {
  TreeNode left;
  TreeNode right;
}
```

For the verification of properties of this class we often need the set of all nodes of a tree.

```
class Tree {
  TreeNode root;
  //: specvar content :: objset
  /*: vardefs content == ... */
  ...
}
```

Complete the definition of content of class Tree such that it corresponds to the set of all tree nodes in this tree. Use builtin sets, lambda expressions, and the closure predicate *rtrancl_pt*. You may assume the tree is acyclic.