J. Hoenicke
A. Nutz

17.05.2017
Please hand in your solution until
24.05.2017, via email to
nutz@informatik.uni-freiburg.de.

Tutorials for "Formal methods for Java"
Exercise sheet 4

Exercise 1: Map implementation

On the lecture's webpage you find the skeleton for a Map data structure that implements functions mapping keys to values using sorted binary trees. Your task is to implement and specify this data structure. The specification should use a `ghost` field of type `JMLValueToObjectMap`.[1]

(a) Implement and specify a method
`private model pure JMLValueToObjectMap computeContent(Node n)`
that computes the content of the binary tree in terms of a `JMLValueToObjectMap`.

(b) Implement and specify a method `public Object add(Key k, Object v)` that inserts a key/value pair into the tree. The tree should remain sorted. If a node with that key already exists, the old mapping will be replaced. The methods returns `null` if the key was not mapped to a value before this method has been called, and the old value otherwise.

(c) Implement and specify a method `public /*@ pure @*/ Object get(Key k)` that returns the value associated with key `k`. If the key is not associated to any value, a `NoSuchElementException` should be thrown. Remember to include the exception in your specification.

(d) Write a small test program that creates an instance of your map and inserts some key/value pairs where keys are of type `IntKey`. Compile and run your program with the JML tools.

(e) Give a class invariant that states sortedness of the tree. Test your invariant with your example program.

Hint: The state of the support for `forall`-quantifiers in OpenJML is somewhat unclear. This exercise can be solved without the use of quantifiers by just using a pure recursive function (or more).

[1] The field declaration is already given in skeleton. For an API description see e.g. `http://www.cs.ucf.edu/~leavens/JML-release/javadocs/org/jmlspecs/models/JMLValueToObjectMap.html`.

Miscelleanous notes:

- Messages from the OpenJML compiler of the form
  `(...) openjml.jar(specs18/java/lang/Object.jml):108: Note: Not implemented for runtime assertion checking: ensures clause containing fresh`
  can be ignored.
- Reference variables, parameters, return types, etc. that might be `null` during run-time need to be declared so (via `/*@ nullable @*/`).
- To compile several files together, just give them all as arguments to the compiler.