J. Hoenicke
J. Christ

06.11.2012
Hand in solutions via email to
`christj@informatik.uni-freiburg.de`
until 13.11.2012 (only Java sources and PDFs accepted)

Tutorials for "Formal methods for Java"
Exercise sheet 3

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 be given in terms of a model field of type `JMLValueToObjectMap`, so make yourself familiar with the methods provided by this class.

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

(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 methods 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: You need to specify a property about all nodes in the tree. There are multiple ways to do that, but all need at least one pure function.