Prof. Dr. A. Poetzsch-Heffter
Mathias Weber, M.Sc.

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Advanced Aspects of Object-Oriented Programming (SS 2013)

Practice Sheet 7

Date of Issue: 28.05.13
Deadline: 04.06.13
(before the lecture as PDF via E-Mail)

Exercise 1 Introduction to JML

The *Java Modeling Language* allows to specify properties of Java software by using special annotations. The JML homepage (`http://www.jmlspecs.org`) provides tutorials, papers, and tools you can use to solve the following exercises.

a) Summarize advantages of using a formal specification technique such as *JML* in comparison to informal approaches. Could you think of any drawbacks caused by the usage of formal specification techniques?

b) Make yourself familiar with the conceptual framework provided by JML. Use the paper „Design by Contract with JML“ by Leavens and Cheon (`http://www.eecs.ucf.edu/~leavens/JML//jmldbc.pdf`) as a starting point.

c) Download a version of the common JML tools (e.g. `http://www.eecs.ucf.edu/~leavens/JML/OldReleases/JML.5.5.tar.gz`), follow the included install instructions and make yourself familiar with their usage. Use the tools for the following exercises to check your specifications and to execute your annotated programs. *Note: The tools only support Java 1.4, so you cannot use generics and the syntax extensions of Java 5 to 7.*

d) Specify the following class. Give pre- and postconditions for the constructor and methods and a non-trivial class and loop invariant.

```
class Container {

    int[] a;
    int n;

    Container( int[] input ){
        n = input.length;
        a = new int[n];
        System.arraycopy(input, 0, a, 0, n);
    }

    int extractMin() {
        int m = Integer.MAX_VALUE;
        int mindex = 0;
        for (int i = 0; i < n; i++) {
            if (a[i] < m) {
                mindex = i;
                m = a[i];
            }
        }
        n--;
        a[mindex] = a[n];
        return m;
    }
}
```

e) It is obvious that the value `n` can only decrease over time. How can you specify this using JML?

f) Use the API documentation of the class `java.io.ByteArrayInputStream` to write a *JML* specification for it.

Exercise 2 Pre- and Postconditions using Assertions

To use pre- and postconditions you do not have to use a complex tool such as JML. You can simply use assertions in your code which check the conditions during execution. These assertions can even be switched on and off, for example when deploying the program. Assertions are a nativ feature of the Java Programming Language since version 1.4.

a) Make yourself familiar with Java assertions, for example using the Java Language Specification §14.10.

b) Replace the JML annotations by Java assertions that check the same properties in the following code:

```
public class Person {
   private /*@ spec_public non_null @*/
     String name;
   private /*@ spec_public @*/
     int weight;

   /*@ public invariant !name.equals("")
    *@         && weight >= 0; @*/

   //@ also
   //@ ensures \result != null;
   public String toString() {
     return "Person(\"" + name + "\","
       + weight + ")";
   }

   //@ also
   //@ ensures \result == weight;
   public int getWeight() {
     return weight;
   }

   /*@ also
     @ requires n != null && !n.equals("");
     @ ensures n.equals(name)
     @   && weight == 0; @*/
   public Person(String n) {
     name = n; weight = 0;
   }

   /*@ also
    @ requires kgs >= 0;
    @ requires weight + kgs >= 0;
    @ ensures weight == \old(weight + kgs);
    @*/
   public void addKgs(int kgs) {
     if (kgs >= 0) {
       weight += kgs;
     } else {
       throw new IllegalArgumentException();
     }
   }
}
```