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 6

Date of Issue: 21.05.13
Deadline: 28.05.13
(before the lecture as PDF via E-Mail)

Exercise 1 Immutable Classes in the JDK

Select five classes from the JDK which conform to the definition of immutability as given in the lecture and explain the reason for their immutability.

Exercise 2 Confined Types

Examine the code, available with this practice sheet on the web, with respect to confinedness. The classes `ProofTreeNodeIt` und `ProofContainer` should provide the externally visible interfaces and are thus not confined.

a) Examine the class `ConfinedList`. Can we declare this class as confined? Can we modify the implementation to make it confined?

b) Examine the class `ProofTreeNode` and answer the same questions as above.

c) Examine the class `PTNIterator` and answer the same questions as above.

d) Download and install the tool “kacheck” from the lecture homepage. This tool is able to check the confinedness of Java classes based on the bytecode of these classes.

   Compile Java source files mentioned in this exercise using the java compiler. Afterwards run the command `kacheck -violations` on the folder the class files are in.

   Compare the output of the tool with the results you expected. Try to explain unexpected results.

   Hint: The tool outputs violations for all classes of the Java runtime that are indirectly used. These outputs can be ignored.

Exercise 3 Encapsulation on Object Level

Consider the class `ConfinedList` from the exercise above. The array stored in the field `data` should be part of the representation of a `ConfinedList`-object, i.e. we annotate it with `rep`.

```
class ConfinedList<T> extends AbstractList<T> implements List<T>, RandomAccess, Cloneable, Serializable {
...
 private /*rep*/ transient Object[] data;
 ...
}
```

a) Is it possible to annotate the remaining parts of the class `ConfinedList`, such that it conforms to the programming discipline presented in the lecture (slide 4.27)? If not, what is the problem with the implementation?

b) Which properties can the programming discipline guarantee? Compare them with the guarantees given by confined types.