Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inform. Kathrin Geilmann

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Advanced Aspects of Object-Oriented Programming (SS 2010)

Practice Sheet 5

Date of Issue: 06.05.10
Deadline: 12.05.10
(until 10 a.m. as PDF via E-Mail)

Exercise 1 *StoJas* Extension

In this exercise, we are going to build extensions to the Java subset *StoJas* that was presented in the lecture.

a) Extend the syntax as well as the semantics (static & dynamic) of *StoJas* to support a "**for** (...) { ... }" statement and give a detailed explanation for the necessary adjustments.

b) Extend the syntax as well as the semantics (static & dynamic) of *StooJas* to support static methods and give a detailed explanation for the necessary adjustments.

c) Extend the syntax as well as the semantics (static & dynamic) of *StooJas* to support static variables and give a detailed explanation for the necessary adjustments.

Exercise 2 Covariance & Contravariance

a) Discuss the concept of Java's *Checked Exceptions* with respect to covariance and contravariance.

b) Which kind of guarantee is given by *Checked Exceptions*?

Exercise 3 Generics

a) Write a generic static method flip which takes an object of class Pair (see the slides of the lecture) and flips the elements of the *given* Pair object. *Hint: In order to flip the elements, both need to be of the same type.*

b) Write a method equals which only allows to compare two objects using the compareTo Method of the Comparable interface.

c) Implement a generic method toArray having the following signature:

```
public static <T> T[] toArray(List<T> l) { ... }
```

d) What is the difference between a Collection<?> and a Collection<Object> ?

e) Explain the output of the following program:

```
public final class GenericClass<T> {
   public void overloadedMethod(Collection<?> o) {
       System.out.println("overloadedMethod (Collection<?>)");
   }
   public void overloadedMethod(List<Number> s) {
       System.out.println("overloadedMethod (List<Number>)");
   }
   public void overloadedMethod(ArrayList<Integer> i) {
       System.out.println("overloadedMethod (ArrayList<Integer>)");
   }

   private void method(List<T> t) {
       overloadedMethod(t); // which method is called?
   }

   public static void main(String[] args) {
       GenericClass <Integer> test = new GenericClass<Integer>();
       test.method(new ArrayList<Integer>());
   }
}
```