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 2011)

Practice Sheet 4 (Hints and Comments)

Exercise 1 Questions

Exercise 2 Super-Calls

```
public class A {
        public void m() { System.out.println(''A''); }
}
public class B {
        public void m() { super.m(); }
}
public class C {
        public static void main(String ... arg) {
                new C().m();
        }
}
```

With static binding new C().m() results in a call to the method m of class A. With dynamic binding it would end in an endless recursion, because the supertype of the current this would be B, so the call would again be dispatched to m in B.

Exercise 3 Tiny Web Server

a)

b) See enclosed source. To be able to compile the complete webserver without changing the whole source, the old code was marked as deprecated but remained in the implementation. It can be removed if all classes use the delegation pattern instead of subtyping.

Exercise 4 University Administration System - Reloaded

a) Advantages:

- Easy to change the current role.
- Easy to extend with new roles, as that does not affect any of the non-role specific parts, whereas subtypes accidentally could modify the behavior of the role independent parts unintended.
- ...

Disadvantages:

- No code for free, everything has to be written for each role
- Guarantees of the type system cannot be used anymore. E.g. no static guarantee anymore, that professors don't register for exams (every one is just a person)
- ...

The given scenario would do something like:

```
public static void main(String... argv) {
                Student s = new Student("Max␣Mustermann");
                Assistant a = new Assistant(s); // copy constructor, i.e. copy all information that remain valid
                                // from the existing student object to the new assistant object
                // invalidate s, remove s from all collections, ...
                Professor p = new Professor (a); // same as for assistant
}
```

b) If the set of types/roles can be given at compile time and it will not change for an object during its lifecycle, it is usually a good idea to use subtyping and, if appropriate, inheritance. If the role of an object is likely to change, it may be better to take a delegation approach. In general, the decision depends on a) the language support, b) the used libraries (some favor delegation, some are designed with subtyping in mind) c) the knowledge of the programmer, etc. Bigger software projects usually use both, e.g. parts of the software that have strong dependencies between them may use subtyping and parts that are not integrated so much may use delegation or forwarding.

Exercise 5 *StoJas* Extension

See the solution to sheet 5.