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 4

Date of Issue: 29.04.10
Deadline: 05.05.10
(until 10 a.m. as PDF via E-Mail)

## Exercise 1 Tiny Web Server

You can find the sources of a simple Java-based web server on the lecture's web page. *Important notice: The sources are meant to illustrate / practice concepts of the lecture. Students will refactor / redesign / extend parts of the software system, so currently existing deficiencies are intentional.*

a) Download the ZIP file, unzip it, and start the server via `ant clean compile run`. Check if everything works by requesting the URL `http://localhost:8080/public_html/HelloWorld.html` using a web browser.

b) Analyse the classes `SimpleWebServer` and `LoggableClass`. Introduce an appropriate interface and refactor the existing code of these two classes so forwarding / delegation is used instead of inheritance.

## Exercise 2 University Administration System - Reloaded

The delegation pattern can be used to simulate inheritance, but it is a more general design pattern (see `http://en.wikipedia.org/wiki/Delegation_(programming)`). Note, the meaning of the terms delegation and forwarding varies in the literature, each author has a slightly different notion of them.

Until now the UAS used inheritance to model the different persons at a university, look at the listing below to see a different implementation, which uses delegation to establish the link between a person and the role, it currently has at the university. In this implementation `Person`-objects delegate some calls to their `Role`-object. Figure 1 shows the architecture of the implementation.

```
package persons;

class Person {
    public String name;
    public Role role;

    public Person(String name) {}
    public void assignRole(Role r) {role = r;}
    public void print() {
        if (role == null)
            System.out.println("Not much known about " + name);
        else
            role.print(this);
    }

    public static void main(String... argv) {
        Person p = new Person("Max Mustermann");
        // Max starts his career
        p.assignRole(new Student());
        // Max graduates and starts working at the university
        p.assignRole(new Assistant());
        // and finally he manages to become a professor
        p.assignRole(new Professor());
    }

}

interface Role {
    public void print (Person p);
}

class Professor implements Role {
    String room;
    String institute;

    public void print(Person p) {
        System.out.print("Professor " + p.name + "'s office is in room " + room);
    }
}
```

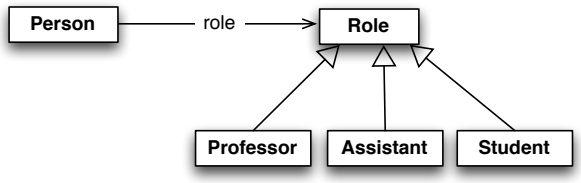


Figure 1: Architecture of the Delegation-based UAS

```
class Student implements Role {
    int reg_num;

    public void print(Person p) {
        System.out.print(p.name + " has the registration number " + reg_num);
    }
}

class Assistant implements Role {
    boolean phDStudent;

    public void print(Person p) {
        System.out.print(p.name + " is a PhD student: " + phDStudent);
    }
}
```

a) Can you think of advantages and disadvantages of the delegation based implementation compared to an inheritance based implementation ? How does the scenario of the main method looks like in a inheritance based system?

b) Formulate a general guideline, when to favor inheritance over delegation and vice versa.

## Exercise 3 Super-Calls

Write a program, that would behave differently under the assumption of dynamically bound `super`-calls than it does with statically bound `super`-calls.