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

Date of Issue: 03.05.11
Deadline: 10.05.11
(until 10 a.m. as PDF via E-Mail)

Remark

We set up an online document `http://bit.ly/lGbs2v`, in which you can enter questions, which you think could appear in the exam. You can formulate the questions in German or English as you like. You could also try to give an answer to the question, but please do not start large discussions in this document. If there are open questions, please discuss them in the exercise courses. It is OK to reformulate questions, but try to ensure that the meaning of the question is not changed. If in doubt add a new question.

Exercise 1 Questions

a) Add one or more questions to the document at `http://bit.ly/lGbs2v`.

Exercise 2 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.

Exercise 3 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.*

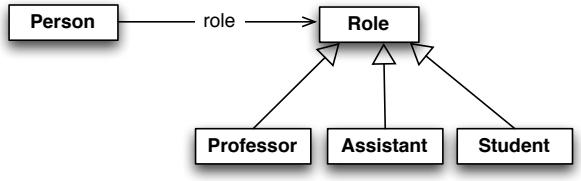
a) Download the ZIP file, unzip it, and start the server via `ant clean compile run`. Check if everything works by requesting the URL `http://localhost:8080/public_html/HelloWorld.html` using a web browser.

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 4 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_pattern`). 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. The figure 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");
        p.assignRole(new Student());     // Max starts his career
        p.assignRole(new Assistant());   // Max graduates and starts working at the university
        p.assignRole(new Professor());   // and finally he manages to become a 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);
    }
}

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 5 *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.