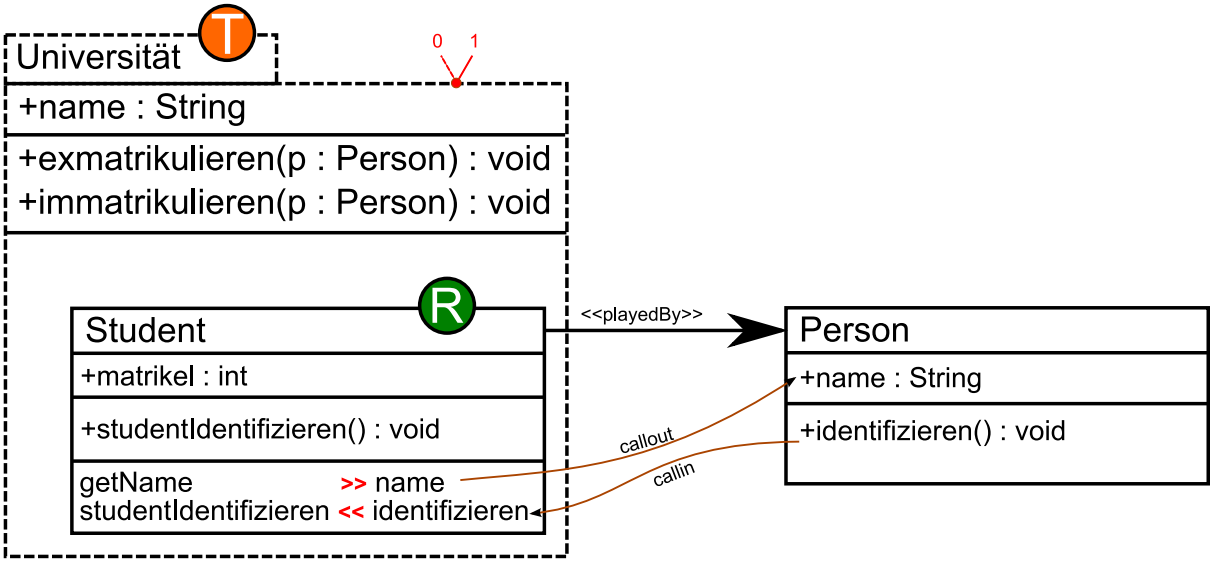
U13 Rollen-basierte Programmierung mit ObjectTeams/Java

Inhalt der Übung

- Einführung in die Konzepte von ObjectTeams/Java

Übungsaufgaben

Personen können innerhalb der Universität durch Immatrikulation die Rolle Student aufnehmen und sie durch Exmatrikulation wieder ablegen. Eine Person wird mit ihrem Namen identifiziert. Ein Student hingegen über seine Matrikelnummer.
Das folgende modifizierte Klassendiagramm stellt diesen Sachverhalt in der Sprache ObjectTeams/Java dar.



Laden Sie sich die Beispielanwendung aus der Einführung herunter und analysieren Sie sie.

Aufgabe 1 (Professor)

Erweitern Sie die Beispielanwendung um die Rolle Professor. Ein Professor wird über seinen Lehrstuhl (String) identifiziert. Hans und Anja bleiben weiterhin Studenten, Peter jedoch wird Professor. Die Ausgabe des Testprogramms soll ungefähr so aussehen:

```
---Uni aktiv-----
Matrikelnummer: 123
Hans
Matrikelnummer: 345
Anja
Peter ist Professor am Lehrstuhl Softwaretechnologie.
```

Überlegen Sie wie Sie innerhalb der Rolle Professor an den Namen der Person kommen.

Hinweis: Wenn Sie bei einen callin „`replace`“ nutzen, muss die Rollenmethode statt der Sichtbarkeit das Schlüsselwort „`callin`“ verwenden.

Aufgabe 2 (Lehrveranstaltungen)

Erweitern Sie das Beispiel um *Lehrveranstaltungen*, die von Studenten besucht und von Professoren gehalten werden. Beschränken Sie sich dabei auf *Vorlesungen* und lassen Sie Übungen außen vor.

- Das Team `Universität` soll um die Methoden `vorlesungAnbieten(…)` und `einschreiben(…)` erweitert werden.
- Die Rolle `Student` soll eine Methode `alleBesuchtenVorlesungen()` enthalten, welche sämtliche von dem jeweiligen Studenten besuchten Lehrveranstaltungen zusammen mit dem Namen des zuständigen Professors ausgibt.
- Die Rolle `Professor` soll eine Methode `alleStudentenMeinerVorlesungen()` enthalten, die die Namen sämtlicher Studenten die eine (oder mehrere!) seiner Lehrveranstaltungen besuchen ausgibt.
- Erweitern Sie das Team `Universität` um die Methoden `besuchteVorlesungenVonStudent(Person p)` und `alleStudentenEinesProfessors(Person p)` um über das Team vom Testprogramm aus, auf die vorher angelegten Rollenmethoden zugreifen zu können.
- Erweitern Sie das Testprogramm um die neu hinzugefügten Methoden zu überprüfen.

Hinweise:

- Betrachten Sie den Typ Vorlesung als Rollentyp, mit dem Kern Lehrveranstaltung.
- Verwenden Sie die reflektive Methode getRole(Basisobjekt, Rollentyp) der Klasse Team um in Team-Methoden zu überprüfen, ob ein Basisobjekt eine Rolle spielt.
- Um die Erzeugung von Rolleninstanzen zu verhindern, können die Parameter von Team-Methoden mit Basisklassen getypt sein und innerhalb der Methode mit Hilfe der reflektiven Team-Methode getRole(Parameter, Rollentyp) die entsprechenden Rolleninstanzen geholt werden.

Literaturempfehlung:

Stephan Herrmann, Christine Hundt, Marco Mosconi: *ObjectTeams/Java Language Definition Version 1.2*. TU Berlin. 2008 (http://objectteams.org/def/1.2/OTJLDv1.2-current.pdf)

Stephan Herrmann: *A Precise Model for Contextual Roles: The Programming Language ObjectTeams/Java.* Applied Ontology, Volume 2, Number 2 / 2007, pp. 181-207, IOS Press. 2007 (http://objectteams.org/publications/JAO07.pdf)

Links:

Eclipse-Download: http://eclipse.org/downloads
ObjectTeams: http://www.objectteams.org

Lösung zur 1. Übungsaufgabe:

1. Eine neue Rolle `Professor` innerhalb des Teams `Universität` anlegen:

```
public class Professor playedBy Person base when (University.this.hasRole(base,
                                                    Professor.class)) {

        professorIdentifizieren <- replace identifizieren;
        String getName() -> String getName();

        public String lehrstuhl;

        callin void professorIdentifizieren() {
                System.out.println(getName() + " ist Professor am Lehrstuhl "+lehrstuhl);
        }
}
```

   Die Rolle enthält ein Attribut `String lehrstuhl`. Mit Hilfe des Callouts auf `getName()` kann auf den Namen der Person zugegriffen werden. Das Callin realisiert die Ersetzung der Methode `identifizieren()`.

2. Eine neue Methode im Team Universität:

```
public void professorEinstellen(Person as Professor prof, String lehrstuhl) {
        prof.lehrstuhl = lehrstuhl;
}
```

   Mit Hilfe des ersten Parameters wird per Declared Lifting die Rolleninstanz von `Professor` für die `Person` erzeugt. Zusätzlich wird der Lehrstuhl des Professors gesetzt.

3. Im Testprogramm Peter zum Professor ernennen:

```
...
Universität uni = new Universität();
...
uni.professorEinstellen(peter, „Softwaretechnologie“);
...
```

Lösung zur 2. Übungsaufgabe:

4. Eine neue Basisklasse `Lehrveranstaltung` anlegen.
    a. Ein Attribut `name` mit Getter und Setter anlegen.
    b. Eine Methode `String asString()` anlegen, die den Namen der Vorlesung zurückgibt.
    c. Die Methode `String toString()` überladen und auf `asString()` umleiten. Die Methode `asString()` ist notwendig, da in einem Rollentyp die Methode `toString()` der Basis nicht mit einem `replace`-Callin unterbrochen werden kann.
5. Einen neuen Rollentyp `Vorlesung` im Team `Universität` anlegen.
    a. Ein Attribut `Professor professor` mit Getter und Setter anlegen.
    b. Ein Attribut `List<Student> students` mit Getter, Setter und Adder anlegen.
    c. Konstruktor `public Vorlesung(Lehrveranstaltung lv) {…}` für Vorlesung anlegen und die Liste `students` initialisieren.
    d. Ein callout `String getName() -> String getName()` anlegen.
    e. Eine Methode String `asString()` anlegen, die den Namen der Vorlesung, den Namen des Professors und die Namen der eingeschriebenen Studenten zurückgibt.
    f. Ein Callin `asString <- replace asString` anlegen.
    g. Die Methode String `toString()` überladen und den Namen der Vorlesung zusammen mit dem Namen des Professors zurückgeben. Das wird später für die Auflistung der von einem Studenten besuchten Vorlesungen genutzt.
6. Dem Team ein Attribut `List<Vorlesung> vorlesungen` hinzufügen, welches später für die Methoden `alleBesuchtenVorlesungen()` und `alleMeineStudenten()` benötigt wird.
7. Einen Standard-Konstruktor für das Team einführen, in dem das Attribut `vorlesungen` initialisiert wird.
8. Dem Team Universität die Methode `vorlesungAnbieten(…)` hinzufügen.

```
public void vorlesungAnbieten(Lehrveranstaltung as Vorlesung vorlesung, Person professor)
{
    Object o = getRole(professor,Professor.class);
    if(o != null) {
            Professor prof = (Professor)o;
            vorlesung.setProfessor(prof);
            vorlesungen.add(vorlesung);
    } else {
            System.err.println("Die Person "+professor+" ist kein Professor!");
    }
}
```

Mit Hilfe des ersten Parameters wird per Declared Lifting für Lehrveranstaltungen die eine Vorlesung-Rolleninstanz erzeugt. Der zweite Parameter ist vom Basistyp Person, da hier keine Rolleninstanz von Professor erzeugt werden soll. Stattdessen wird mit Hilfe der reflektiven Methode getRole überprüft, ob die übergebene Person-Instanz ein Professor ist. Wenn es sich um einen Professor handelt, liefert diese Methode die Rolleninstanz zurück, welche dann auf den Rollentyp gecastet werden kann. Nach dem Cast, wird der Professor der Vorlesung zugewiesen und die Vorlesung der Liste von Vorlesungen hinzugefügt. Handelt es sich nicht um einen Professor, liefert die Methode `getRole(…)` `null` zurück, was durch eine Fehlerausschrift behandelt wird.

9. Dem Team die Methode `einschreiben(…)` hinzufügen.

```
public void einschreiben(Person student, Lehrveranstaltung vorlesung) {
    Object stud = getRole(student, Student.class);
    Object vl   = getRole(vorlesung, Vorlesung.class);
    if(stud != null && vl != null) {
            Student s = (Student)stud;
            Vorlesung v = (Vorlesung)vl;
            v.addStudent(s);
    }
}
```

   Beide Parameter dieser Methode sind Basistypen. Mit Hilfe der Methode `getRole(…)` wird überprüft, ob die übergebene Person ein Student und die übergebene Lehrveranstaltung eine Vorlesung ist. Wenn dies der Fall ist, werden die Person und die Lehrveranstaltung auf den Rollentyp Student, bzw. Vorlesung gecastet und der Student der Vorlesung hinzugefügt.

10. Der Rolle Professor die Methode `alleStudentenMeinerVorlesungen()` hinzufügen.

```
public void alleStudentenMeinerVorlesungen() {

    List<Vorlesung> gehalteneVorlesungen = new ArrayList<Vorlesung>();
    for(Vorlesung v : vorlesungen) {
            if(v.getProfessor().equals(this)) {
                    gehalteneVorlesungen.add(v);
            }
    }

    Set<Student> meineStudenten = new HashSet<Student>();
    for(Vorlesung v : gehalteneVorlesungen) {
            meineStudenten.addAll(v.getStudents());
    }

    for(Student s : meineStudenten) {
            System.out.println(s);
    }
}
```

   Zunächst werden alle von der Universtät angebotenen Vorlesungen untersucht und diejenigen Vorlesungen, die von dem jeweiligen Professor gehalten werden, in eine Hilfs-`List` aufgenommen. Anschließend werden die so gefundenen Vorlesungen durchlaufen und die in diese Vorlesungen eingeschriebenen Studenten in einem `Set` zusammengefasst. Die Verwendung von `Set` eliminiert die möglichen Duplikate. Abschließend werden alle so gefundenen Studenten durchlaufen und auf die Konsole ausgegeben.

11. Der Rolle Student die Methode `alleBesuchtenVorlesungen()` hinzufügen.

```
public void alleBesuchtenVorlesungen() {
    List<Vorlesung> besuchteVorlesungen = new ArrayList<Vorlesung>();
    for(Vorlesung v : vorlesungen) {
            for(Student s : v.getStudents()) {
                    if(s.equals(this)) {
                            besuchteVorlesungen.add(v);
                            break;
                    }
            }
    }
    System.out.println("Besuchte Vorlesungen von "+this);
    for(Vorlesung v : besuchteVorlesungen) {
            System.out.println(v);
    }
}
```

Zunächst werden alle von der Universität angebotenen Vorlesungen und für jede Vorlesung wiederum alle eingeschrieben Studenten durchlaufen. Wird der jeweilige Student in einer Vorlesung gefunden, so wird die Vorlesung in eine Hilfs-`List` aufgenommen. Abschließend wird diese Hilfsliste durchlaufen und jede Vorlesung auf die Konsole ausgegeben.

12. Dem Team Universität die Methode `besuchteVorlesungenVonStudent(Person p)` hinzufügen.

```
public void besuchteVorlesungenVonStudent(Person p) {
    Object o = getRole(p,Student.class);
    if(o != null) {
            Student s = (Student)o;
            s.alleBesuchtenVorlesungen();
    }
}
```

Zunächst wird mit der Methode `getRole` überprüft, ob die übergebene Person ein Student ist. Wenn das der Fall ist, wird die Person-Instanz auf den Rollentyp gecastet und die Methode `alleBesuchtenVorlesungen()` darauf aufgerufen.

13. Dem Team Universität die Methode `alleStudentenEinesProfessors(Person p)` hinzufügen.

```
public void alleStudentenEinesProfessors(Person p) {
    Object o = getRole(p,Professor.class);
    if(o != null) {
            Professor prof = (Professor)o;
            prof.alleStudentenMeinerVorlesungen();
    }
}
```

Zunächst wird mit der Methode `getRole` überprüft, ob die überprüft, ob die übergebene Person ein Professor ist. Wenn das der Fall ist, wird die Person-Instanz auf den Rollentyp gecastet und die Methode `alleStudentenMeinerVorlesungen()` aufgerufen.

14. Das Testprogramm kann wie folgt angepasst werden:

```
public static void main(String[] args) {
            //Personen anlegen
            Person hans = new Person("Hans");
            Person anja = new Person("Anja");
            Person peter = new Person("Peter");
            Person klaus = new Person("Klaus");
            Person ina = new Person("Ina");

            //Lehrveranstaltungen anlegen
            Lehrveranstaltung st1 = new Lehrveranstaltung("Softwaretechnologie I");
            Lehrveranstaltung dpf = new Lehrveranstaltung("Design Patterns");
            Lehrveranstaltung mathe = new Lehrveranstaltung("Mathe");

            //Team instanziieren und aktivieren
            University u = new University();
            u.activate();

            //Studenten immatrikulieren
            u.immatrikulieren(hans, 123); //Hans immatrikulieren
            u.immatrikulieren(anja, 345); //Anja immatrikulieren
            u.immatrikulieren(ina, 567); //Ina immatrikulieren
```

```
            //Professoren ernennen
            u.professorEinstellen(peter, "Softwaretechnologie"); //Peter wird Professor
            u.professorEinstellen(klaus, "Mathematik"); //Klaus wird Professor

            //Vorlesungen anbieten
            u.vorlesungAnbieten(st1, peter); // "ST1“, gehalten von Peter, anbieten
            u.vorlesungAnbieten(dpf, peter); // "DPF“, gehalten von Peter, anbieten
            u.vorlesungAnbieten(mathe, klaus); // "Mathe“, gehalten von Klaus, anbieten

            //Studenten in Vorlesungen einschreiben
            u.einschreiben(hans, st1); //Hans schreibt sich für die Vorlesung "ST1" ein
            u.einschreiben(anja, st1); //Anja schreibt sich für die Vorlesung "ST1" ein
            u.einschreiben(anja, mathe); //Anja schreibt sich für die Vorlesung "Mathe" ein
            u.einschreiben(ina, mathe); //Ina schreibt sich für die Vorlesung "Mathe" ein
            u.einschreiben(anja, dpf); //Anja schreibt sich für die Vorlesung "DPF" ein
            u.einschreiben(ina, dpf); //Ina schreibt sich für die Vorlesung "DPF" ein

            u.besuchteVorlesungenVonStudent(anja);
            System.out.println();
            u.besuchteVorlesungenVonStudent(ina);
            System.out.println();

            System.out.println("Studenten von Professor Peter");
            u.alleStudentenEinesProfessors(peter);
            System.out.println();
            System.out.println("Studenten von Professor Klaus");
            u.alleStudentenEinesProfessors(klaus);
}
```

Das obige Testprogramm erzeugt folgende Ausgabe auf der Konsole:

```
Besuchte Vorlesungen von Student Anja
Softwaretechnologie I gehalten von Professor Peter
Design Patterns and Frameworks gehalten von Professor Peter
Mathe gehalten von Professor Klaus

Besuchte Vorlesungen von Student Ina
Design Patterns and Frameworks gehalten von Professor Peter
Mathe gehalten von Professor Klaus

Studenten von Professor Peter
Student Ina
Student Hans
Student Anja

Studenten von Professor Klaus
Student Ina
Student Anja
```

Aufgabe 3 (Tutor)

Unterscheiden Sie bei Lehrveranstaltungen zwischen Vorlesungen und Übungen. Vorlesungen werden von Professoren gehalten. Übungen von Tutoren. Überlegen Sie sich wie Sie die Rolle Tutor realisieren können.
Hinweis: Rollen dürfen nicht an Rollen desselben Teams gebunden werden. Es werden mehrere weitergehende Konzepte für die Lösung dieser Aufgabe benötigt. (Team-Anker, Externalized Roles, ...)