UNIKASSEL VERSITÄT

SOFTWARE ENGINEERING
UNIVERSITÄT KASSEL

Programmiermethodik

Übung 10

Wintersemester 2012 / 13

Fachgebiet Software Engineering

Tobias George

george@uni-kassel.de

Agenda

- Besprechung HA 8
- eDOBS: Live Demo
- Zetteltest
- Praktische Übung: Castle
- Vorschau HA 9

Besprechung HA8 I

- **Aufgabe 1:**
  - DiaballikGame::checkWinner()
- **Aufgabe 2:**
  - Holder::move(Field f)
- **Aufgabe 3:**
  - Ball::pass(Holder h)

eDOBS

LIVE DEMO

Methodenentwurf I - Zetteltest

- **Mögliche Implementierung**

```
1  public void moveStones ()
2  {
3        Pit nextPit = getNext();
4        for(Iterator<Stone> iter = getIteratorOfStones(); iter.hasNext();)
5        {
6              Stone stone = iter.next();
7              nextPit.addToStones(stone);
8              nextPit = nextPit.getNext();
9        }
10 }
```

- **Verifikation durch Zetteltest**

Methodenentwurf II - Zetteltest

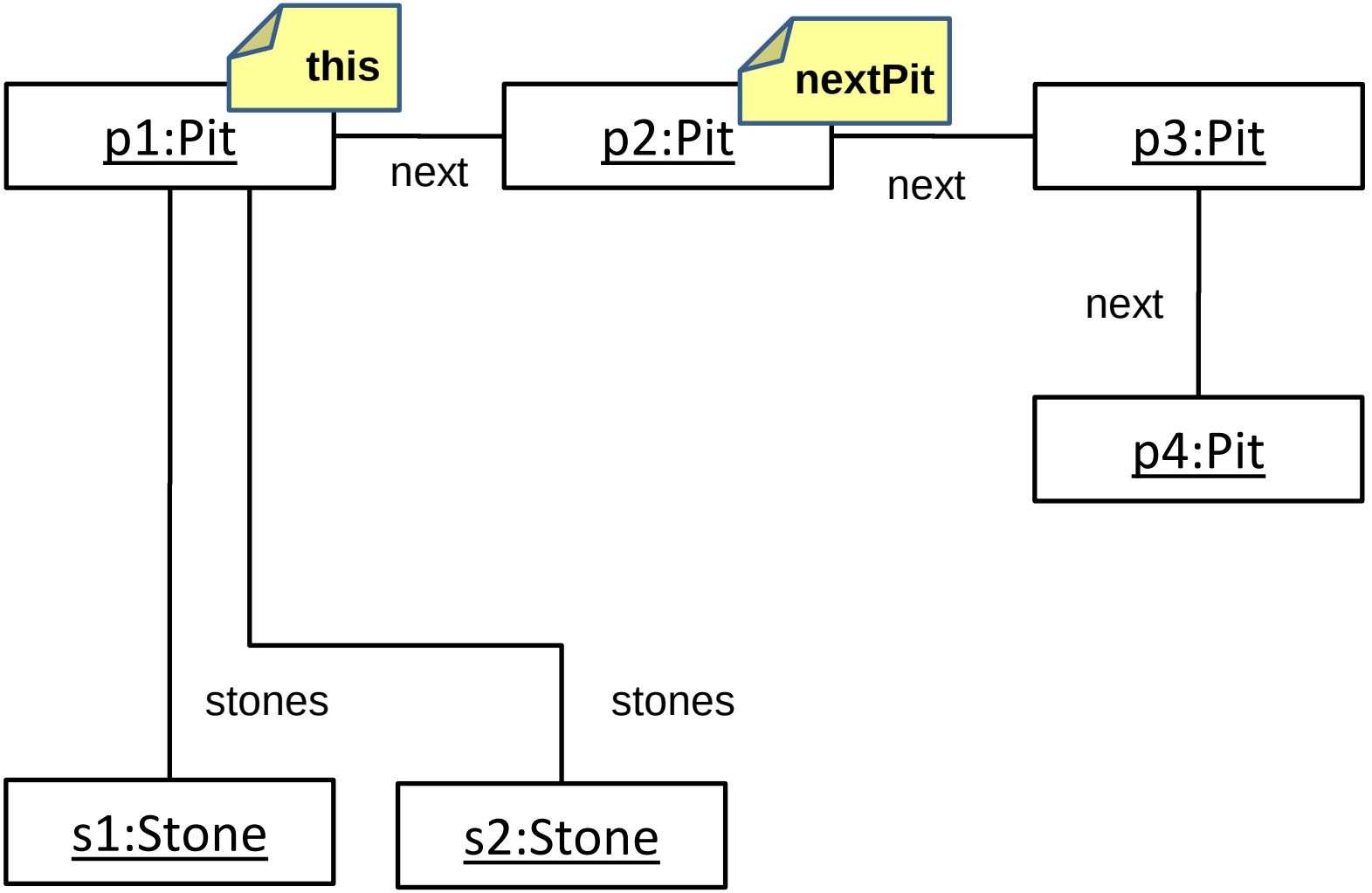
- **Zetteltest**
  - Hilft Methoden zu verstehen
  - Fehler aufzudecken
  - Ist „manuelles Debuggen“
- **Zetteltest Methode `moveStones():void` der Klasse `Pit`**

Methodenentwurf III - Zetteltest

- **Mögliche Implementierung**

```
1  public void moveStones ()
2  {
3        Pit nextPit = getNext();
4        for(Iterator<Stone> iter = getIteratorOfStones(); iter.hasNext();)
5        {
6                Stone stone = iter.next();
7                nextPit.addToStones(stone);
8                nextPit = nextPit.getNext();
9        }
10 }
```

Methodenentwurf IV - Zetteltest



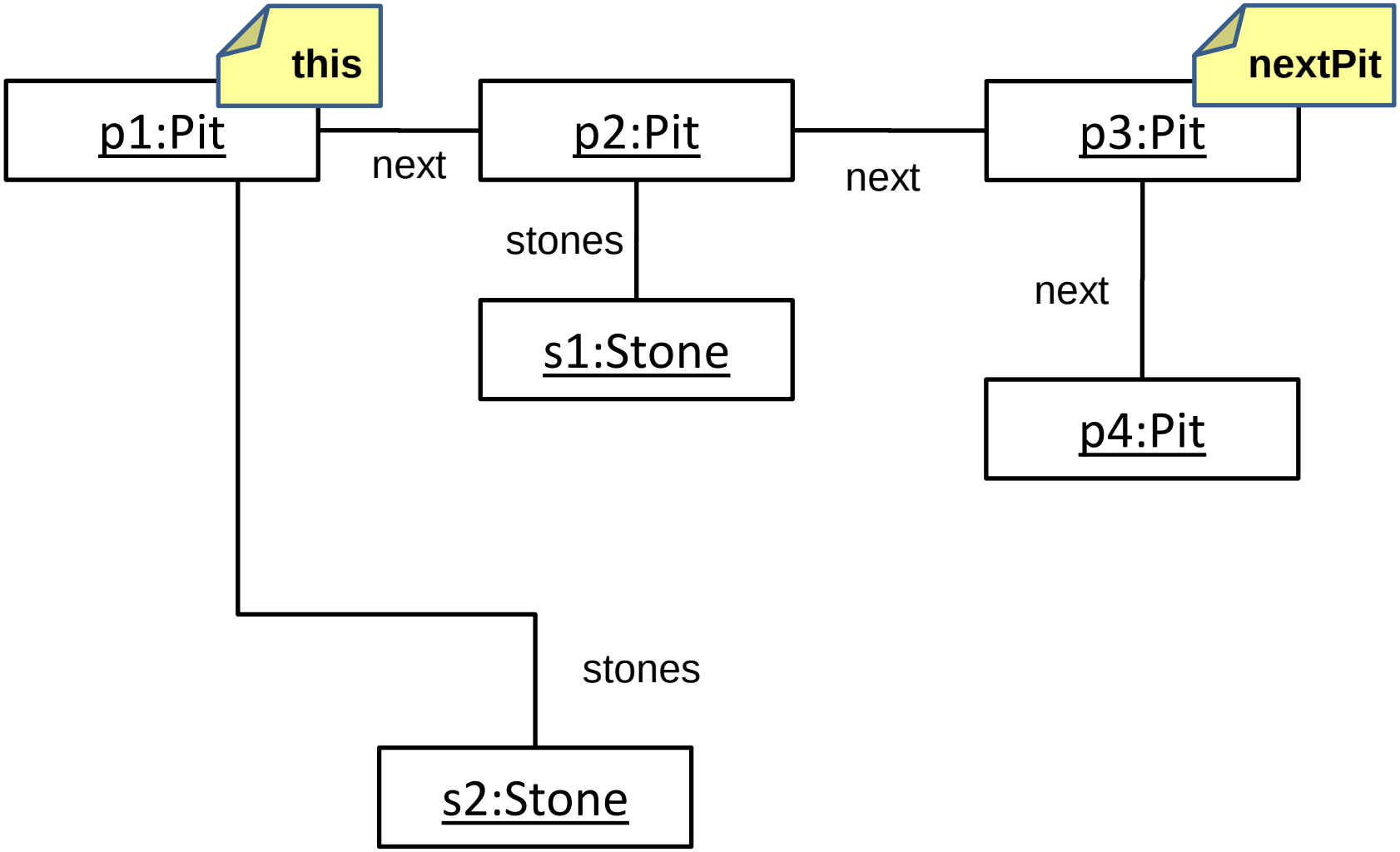
Methodenentwurf V - Zetteltest

- **Mögliche Implementierung**

```
1  public void moveStones ()
2  {
3        Pit nextPit = getNext();
4        for(Iterator<Stone> iter = getIteratorOfStones(); iter.hasNext();)
5        {
6              Stone stone = iter.next();
7              nextPit.addToStones(stone);
8              nextPit = nextPit.getNext();
9        }
10 }
```

- **Verifikation durch Zetteltest**

Methodenentwurf VI - Zetteltest



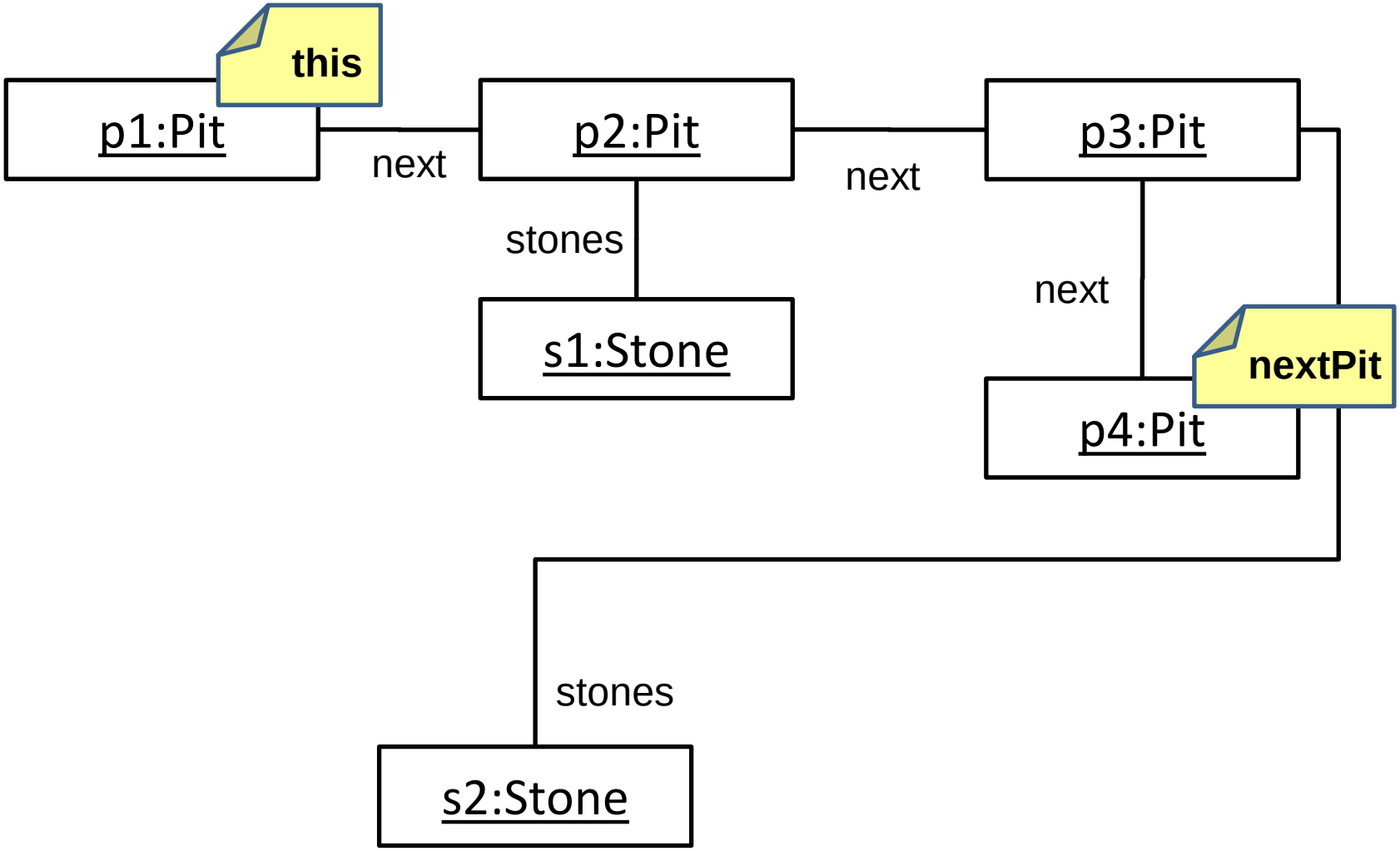
Methodenentwurf VII - Zetteltest

- **Mögliche Implementierung**

```
1  public void moveStones ()
2  {
3        Pit nextPit = getNext();
4        for(Iterator<Stone> iter = getIteratorOfStones(); iter.hasNext();)
5        {
6              Stone stone = iter.next();
7              nextPit.addToStones(stone);
8              nextPit = nextPit.getNext();
9        }
10 }
```

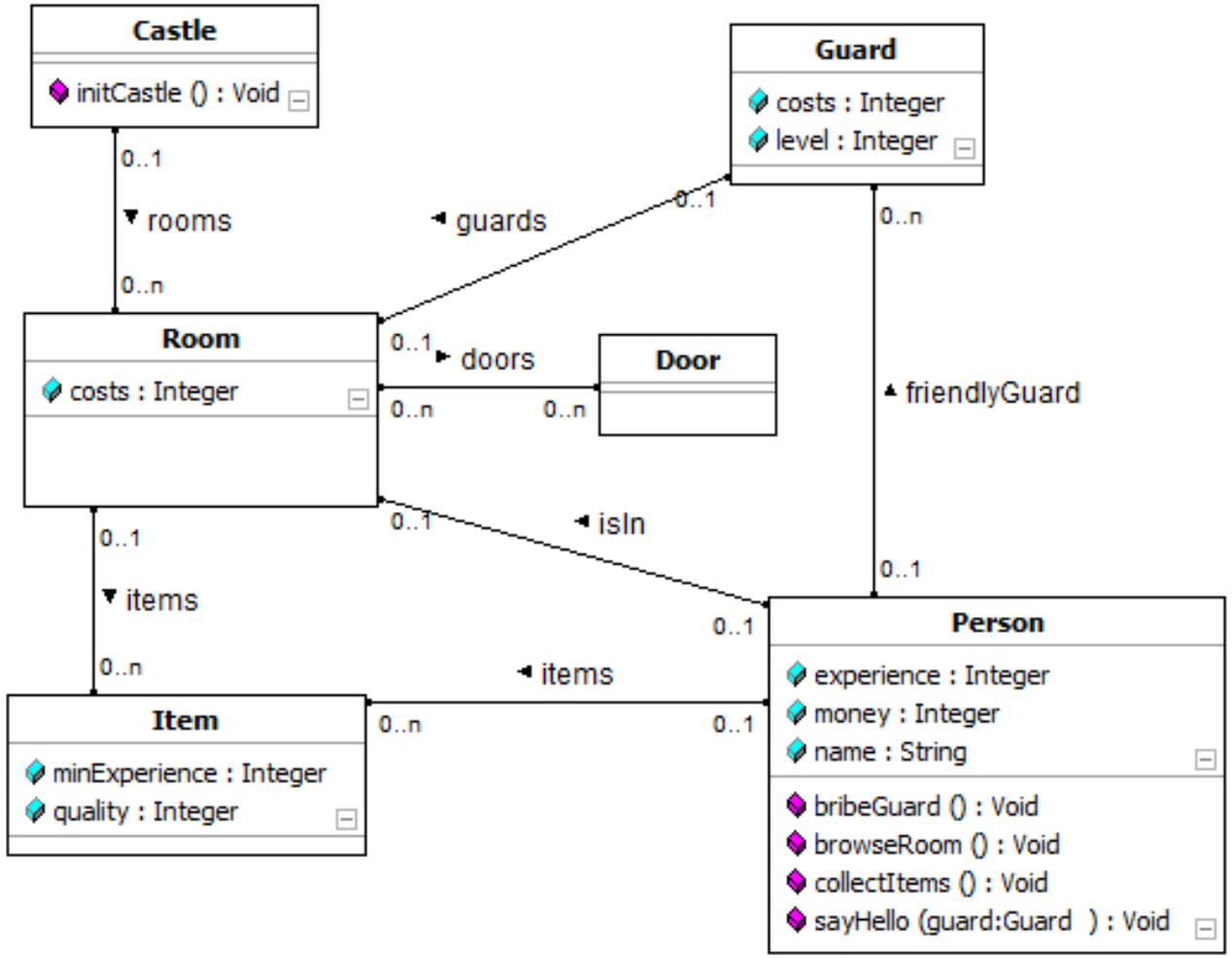
- **Verifikation durch Zetteltest**

Methodenentwurf VIII - Zetteltest



Praktische Übung

- **Castle**

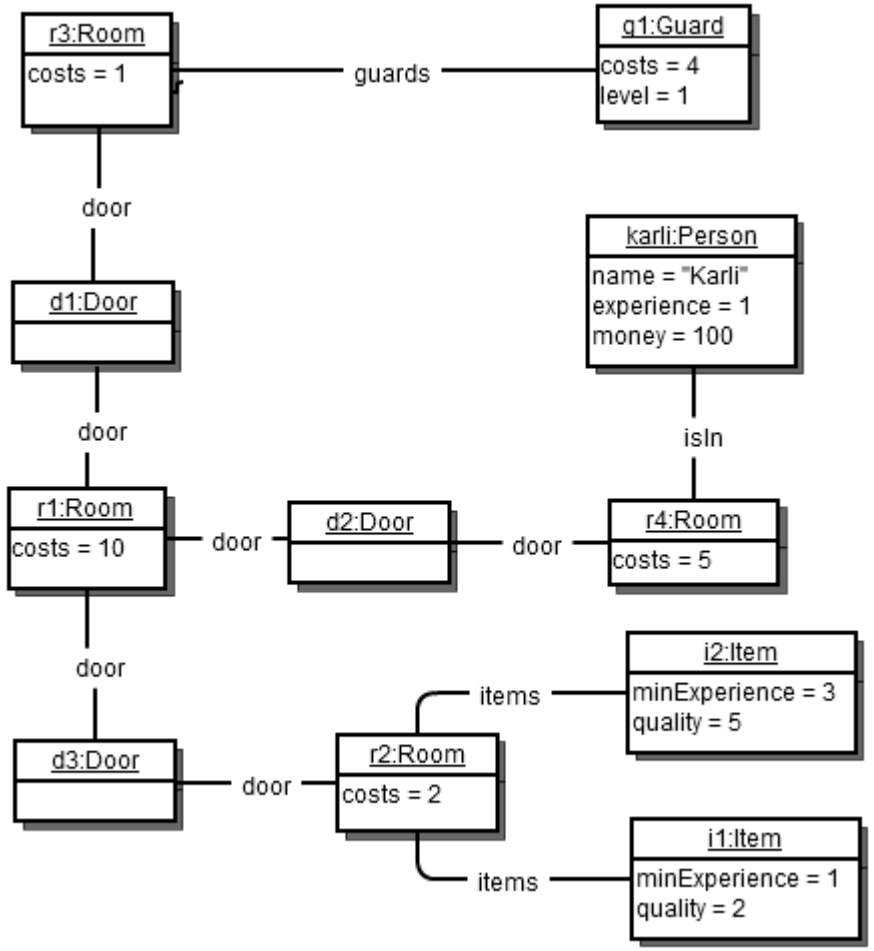


Praktische Übung

- **Castle**
  - Das Schloss hat viele Räume, die mit Türen verbunden sind
  - Räume haben Kosten die bezahlt werden müssen
  - In Räumen können Items liegen
  - Items haben einen Wert (quality) und ein Level: Nur Personen mit mindestens dem gleichen Level können den Gegenstand aufnehmen
  - Räume können von Wächtern bewacht werden
  - Wächter müssen zunächst mit Geld bestochen werden, bevor der Raum geplündert werden kann.
  - Einmal bestochene Wächter sind einer Person freundlich gesinnt (friendlyGuard)
  - Personen haben einen Namen (name), Erfahrungspunkte (experience) und Geld (money)
  - Personen können Items besitzen

Praktische Übung

- **Beispiel Objektdiagramm**



Praktische Übung

- **Aufgabe:**

  – Erstellt einen Zetteltest zur Methode Person::collectItems():void

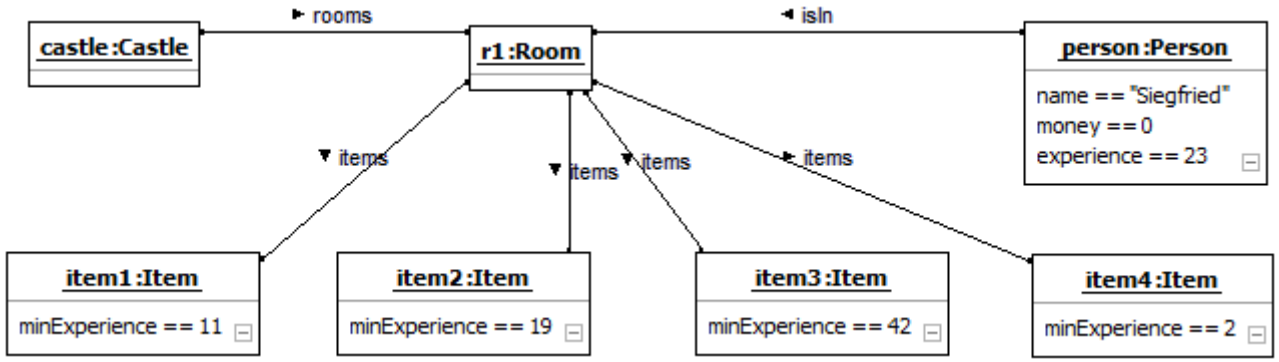
```
public void collectItems ()
{
    Room r = this.getIsIn();

    for(Item item : r.getItems())
    {
                if(this.getExperience() >= item.getMinExperience())
                {
                            item.setRoom(null);
                            item.setPerson(this);

                            int itemQuality = item.getQuality();
                            int newExperience = this.getExperience() + itemQuality;
                            this.setExperience(newExperience);
                }
    }
}
```

Praktische Übung II

- **Initiale Objektstruktur:**



Zeigt das Ergebnis einen Betreuer

Vorschau HA9 I

- **Deadline: 31.01.2012, 23:59 Uhr**
- **Aufgabe 1: Zetteltest**
- **Aufgabe 2: eDOBS**

Ende

Jetzt: Betreutes Arbeiten

Ansonsten: Schönes WE!