UNIKASSEL VERSITÄT

Research Group Software Engineering

Programmiermethodik

Übung 9

Wintersemester 2011 / 12

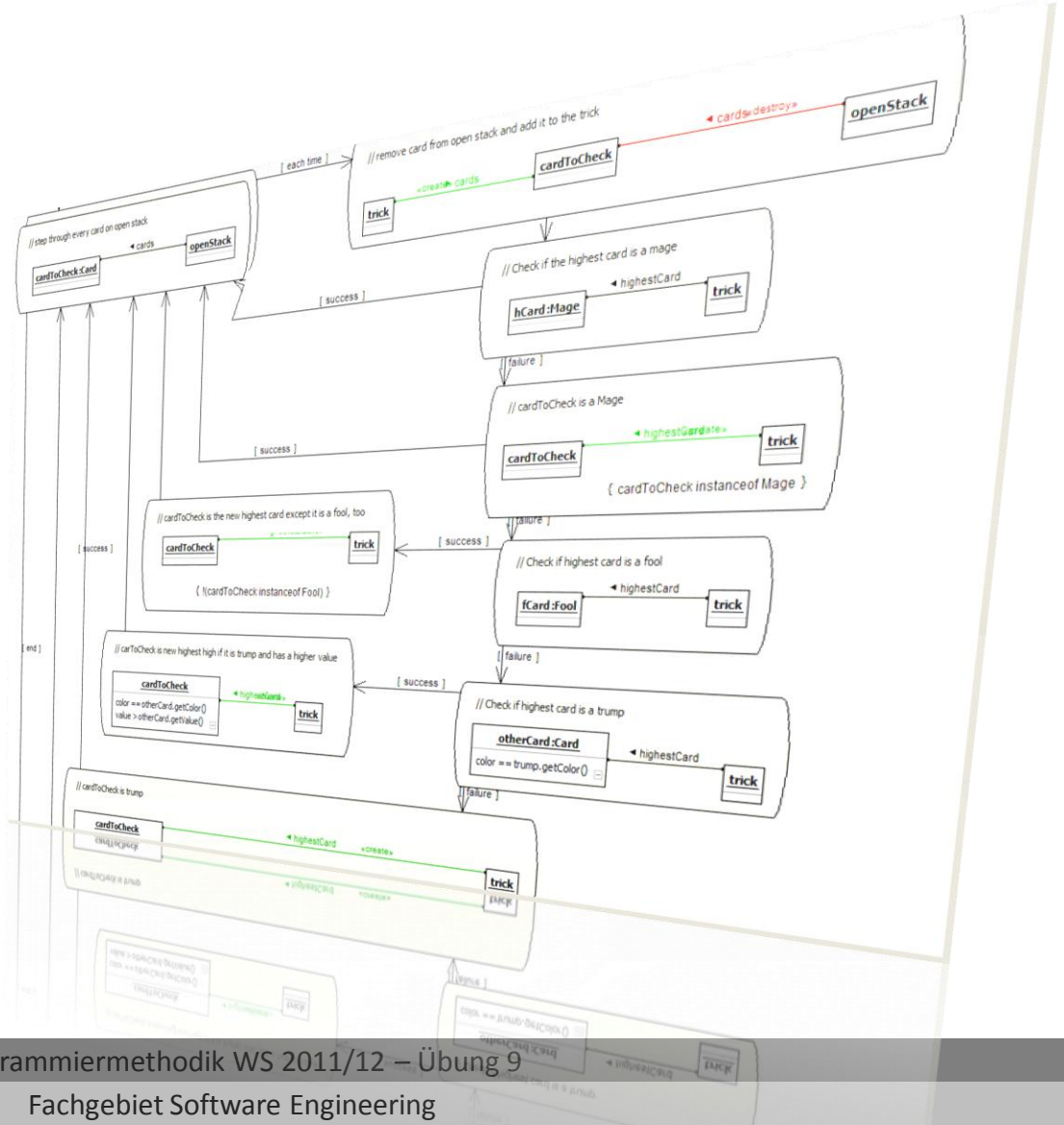
Fachgebiet Software Engineering

Tobias George

george@uni-kassel.de

Agenda

- Besprechung HA 7
- Storydiagramme
- Wiederholung Zetteltest
- Vorschau HA 8
- Praktische Übung: Castle



Besprechung HA7 I

- **1.1 LoginController**
  - Öffnet die GUI der Login Oberfläche
  - Initialisiert das Spiel
  - Startet den GameController
- **1.2 GameController**
  - Öffnet die GUI der Spielfläche
  - Startet Controller für alle Spieler, Hexen und den Würfel
  - Zeigt eine Meldung und beendet das Spiel wenn ein Spieler gewonnen hat

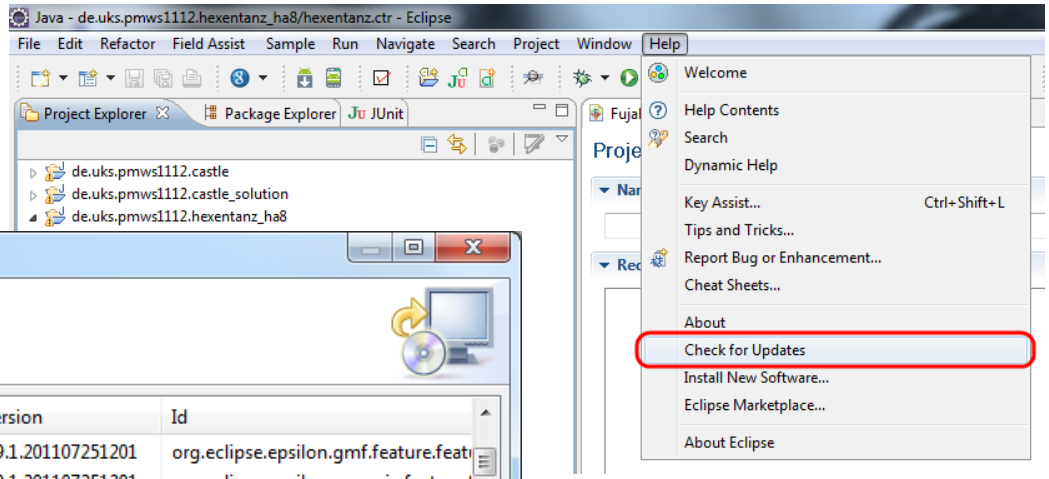
Besprechung HA7 II

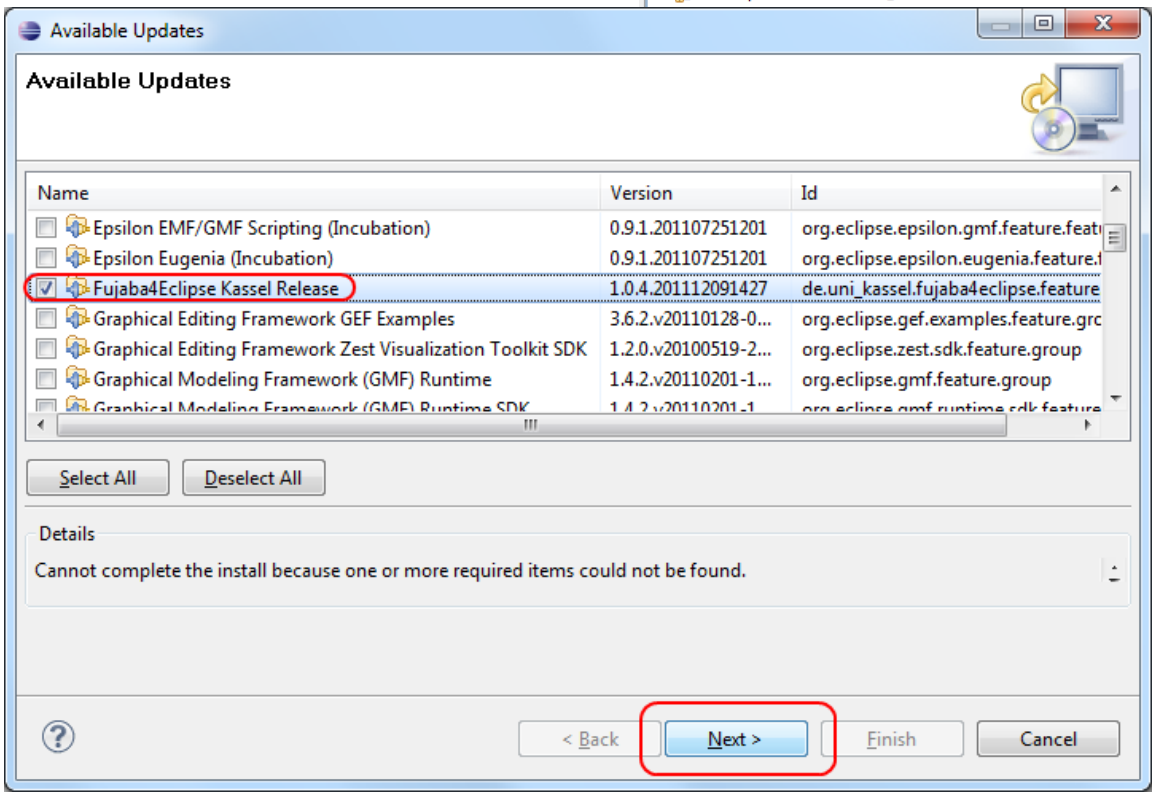
- **1.3 PlayerController**
  - Zeigt den aktuellen Spieler und die Instruktion an
- **1.4 WitchController**
  - Zeigt an welche Hexe sich rückwärts bewegt
- **1.5 DiceController**
  - Ermöglicht das Würfeln und zeigt das Ergebnis an

- **1.6 StartHexentanz**
  - Startet den LoginController

Storydiagramme

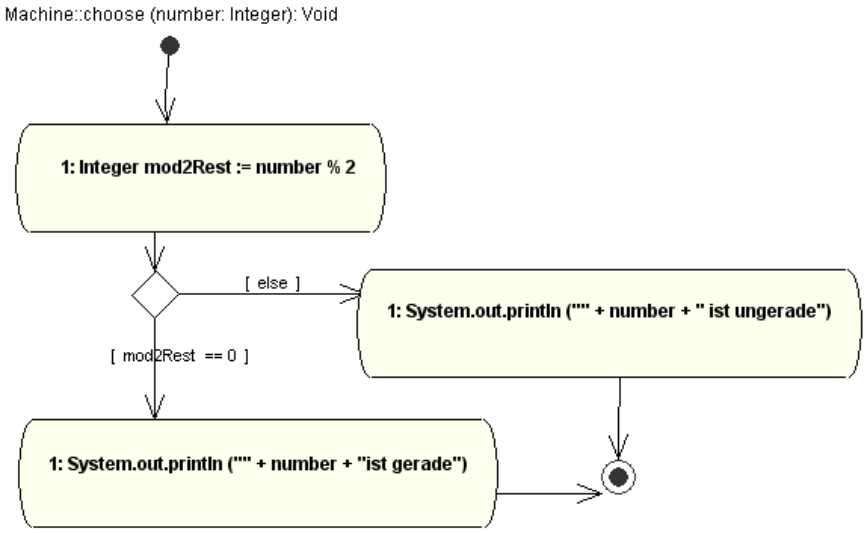
- **Update Fujaba4Eclipse:**
  - neue Features und Fixes





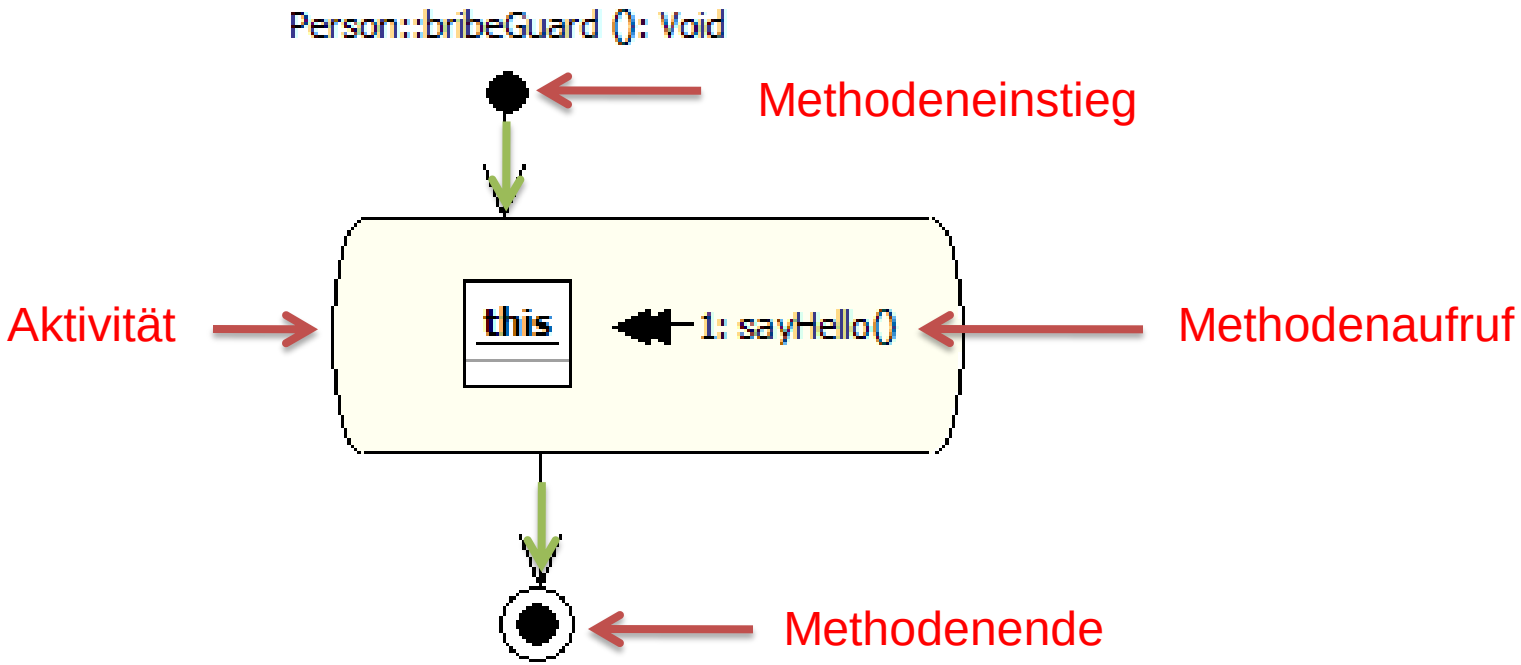
Storydiagramme

- **Warum?**
  - Erhöhen das Abstraktionsniveau
  - Man kann sich auf das Wesentliche konzentrieren
  - Änderungen in Diagrammen sind meist leichter als im Code
  - Nachvollziehbarkeit von Code geht in Diagrammen leichter
  - …



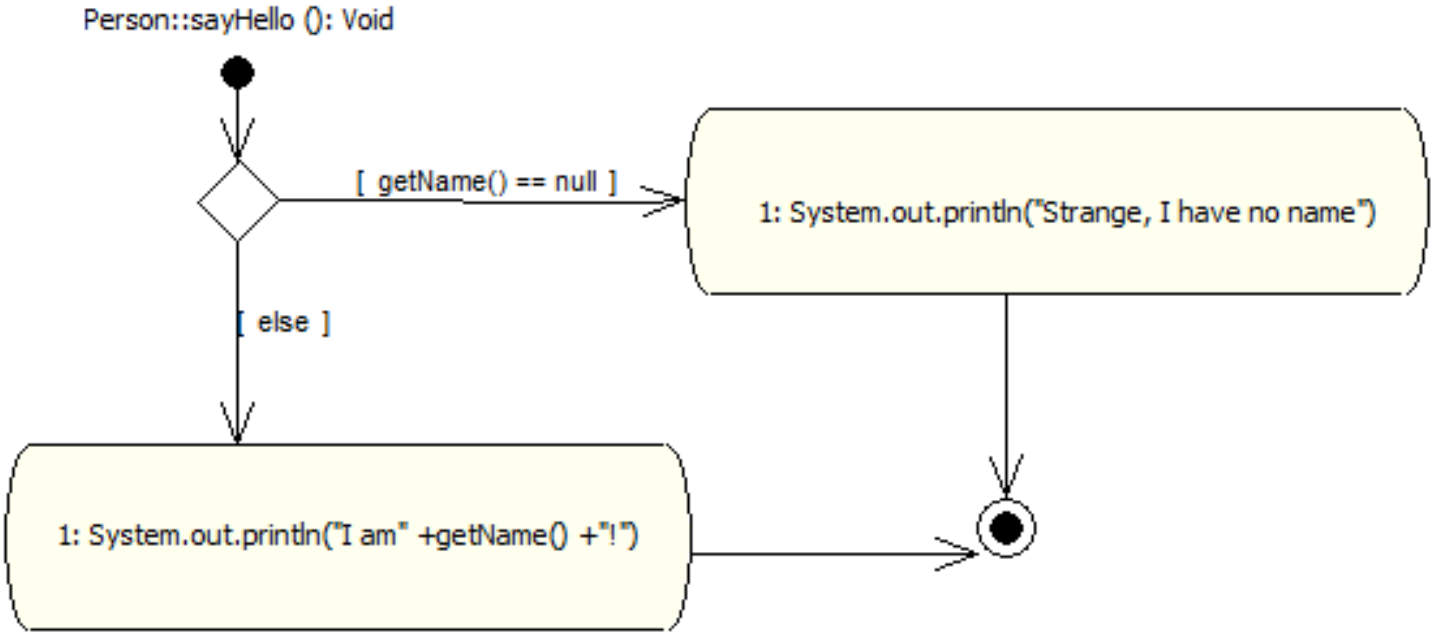
Storydiagramme

- **Kontrollfluss und Methodenaufrufe**



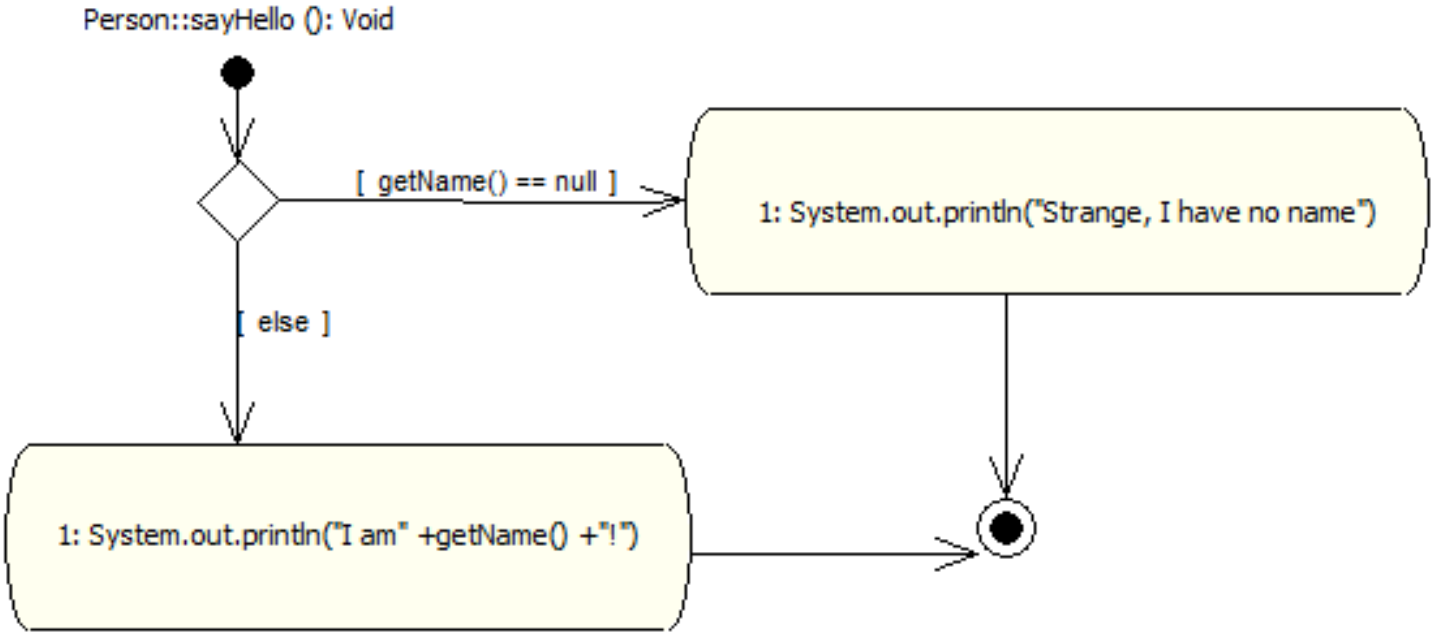
Storydiagramme

- **Fallunterscheidung**



Storydiagramme

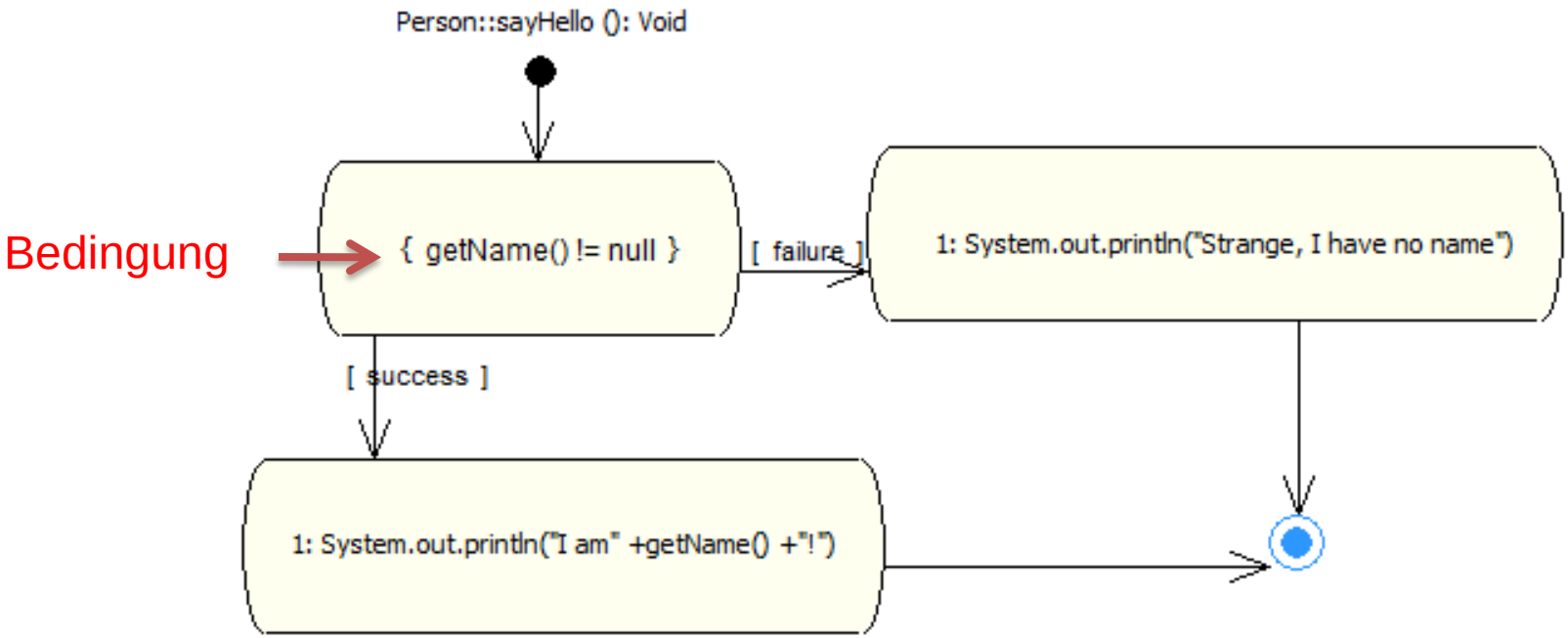
- **Fallunterscheidung**



- **oder auch…**

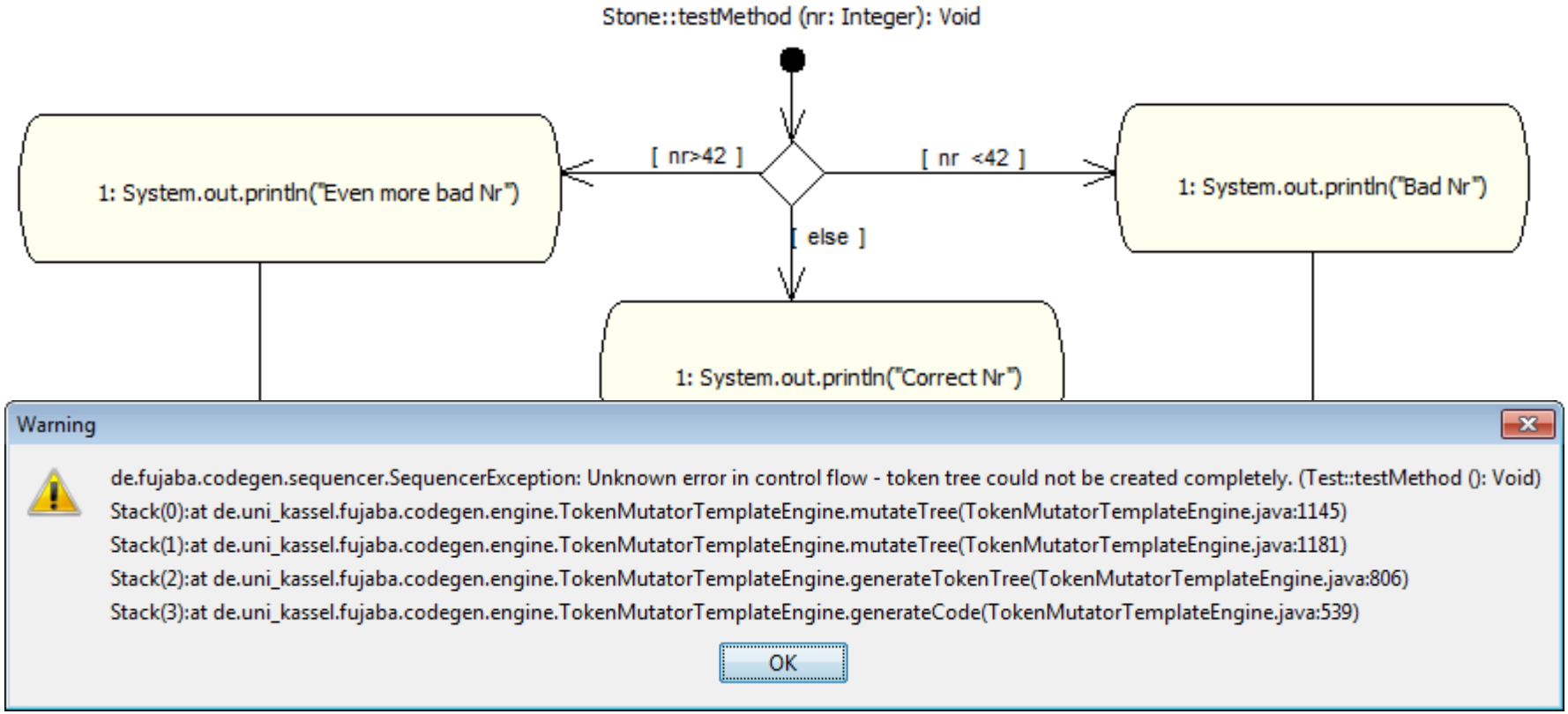
Storydiagramme

- **Fallunterscheidung**



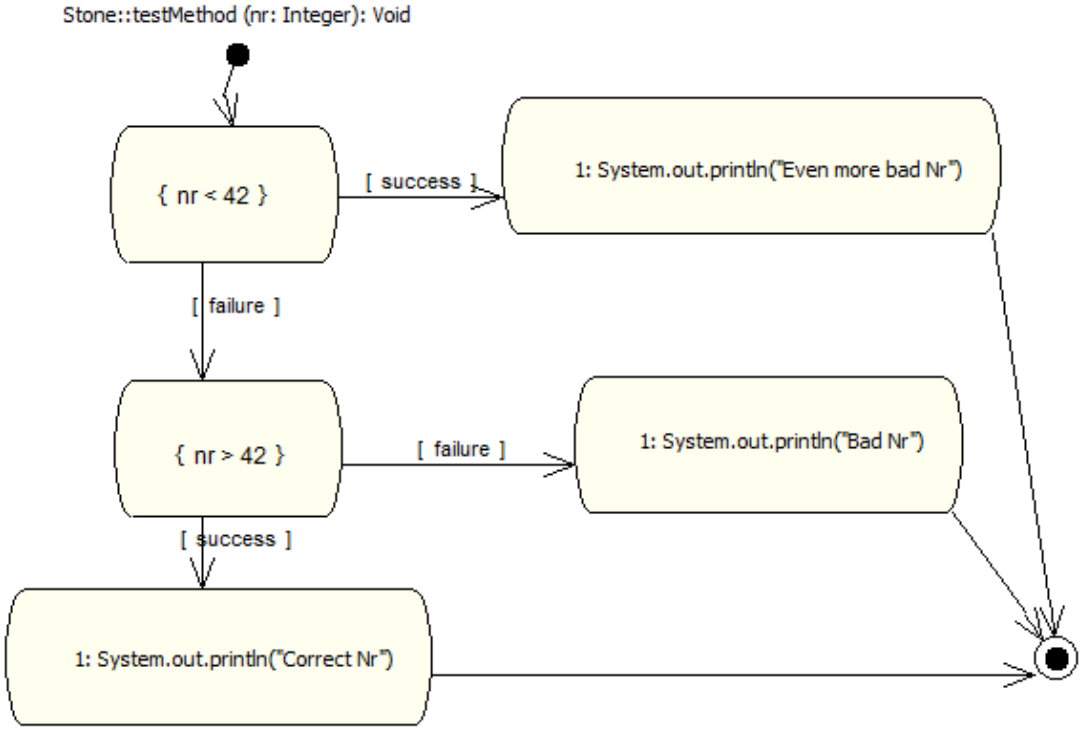
Storydiagramme

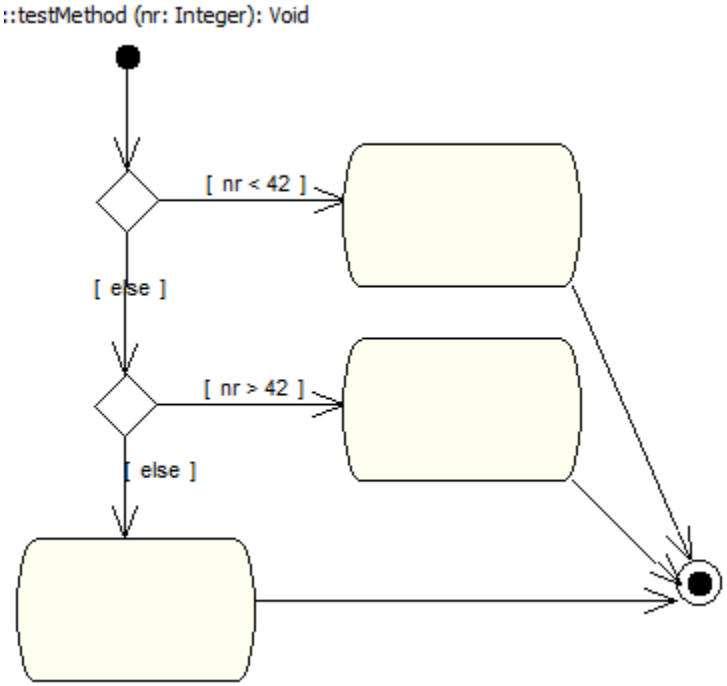
- **Achtung: „Diamanten“ scheinen im Moment fehlerhaft zu sein:**



Storydiagramme

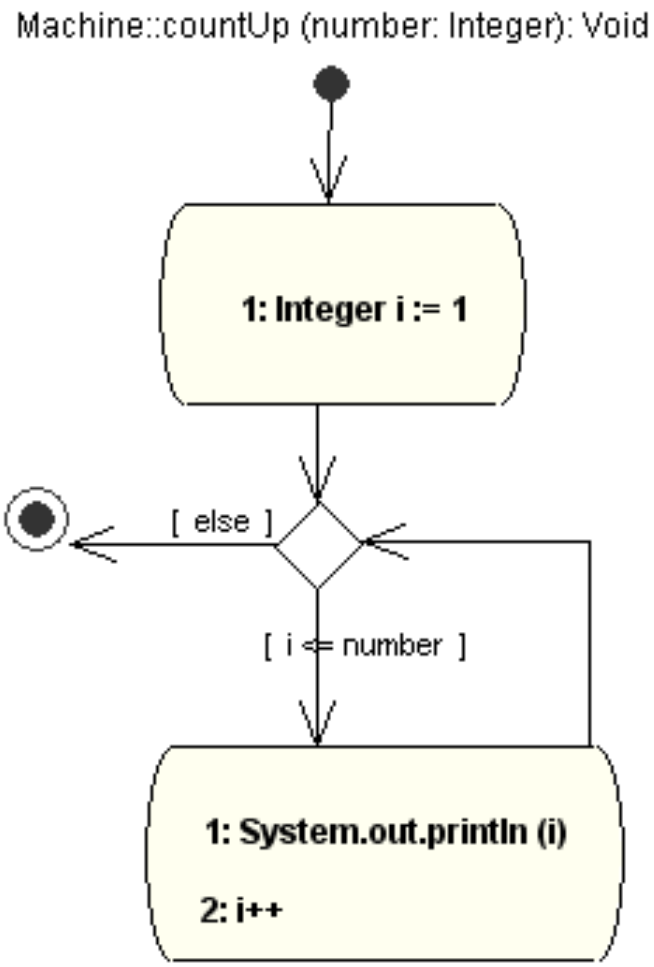
- **Workarounds:**





Storydiagramme

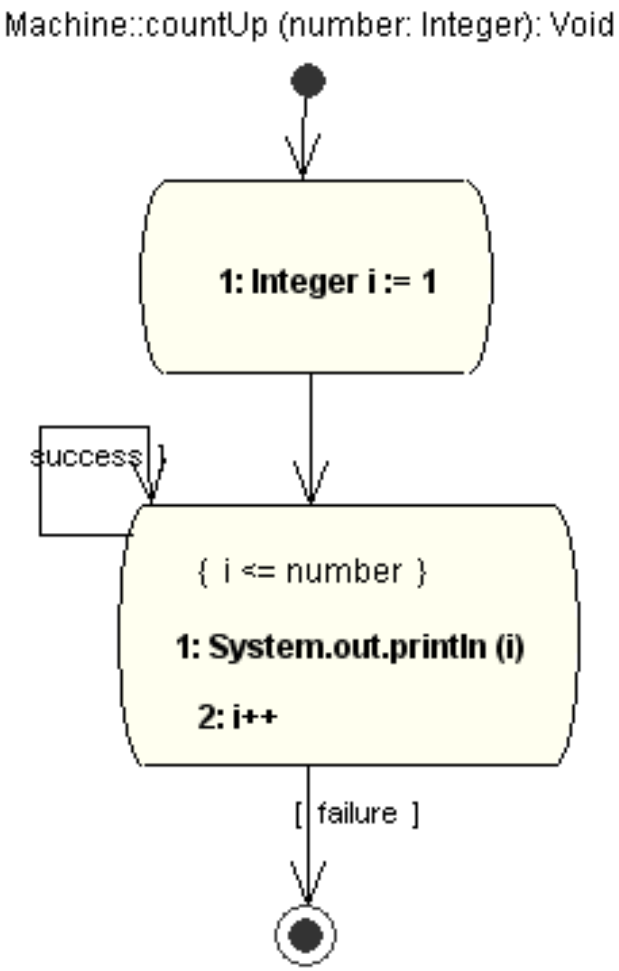
- **Zählschleife**



- **oder auch…**

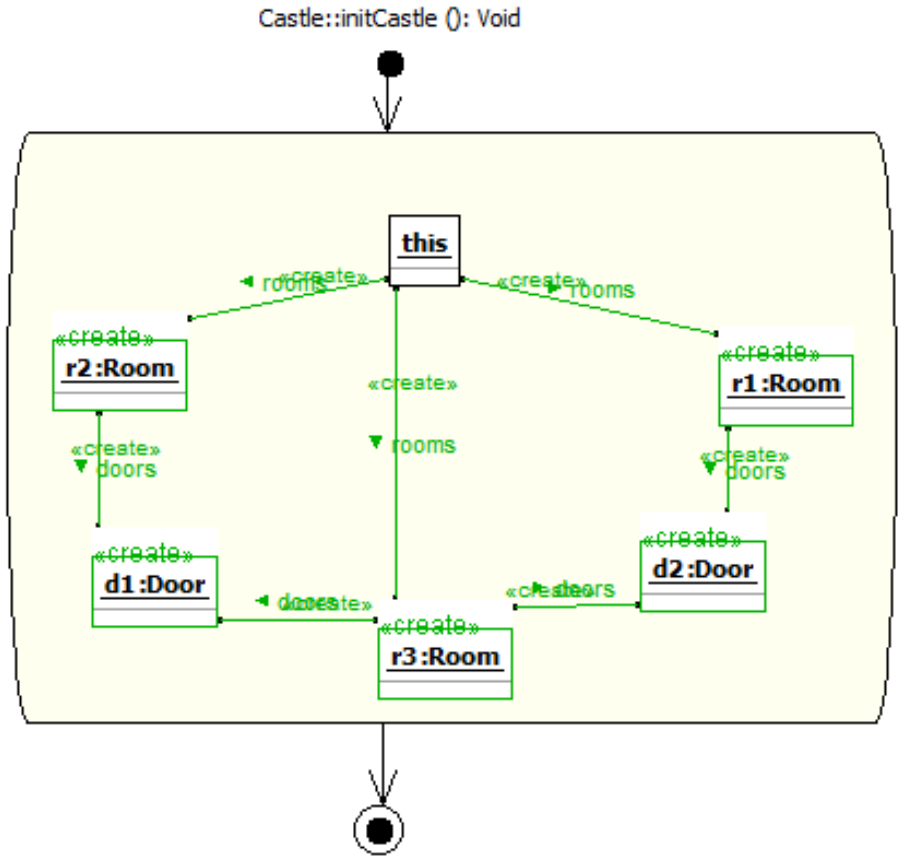
Storydiagramme

- **Zählschleife**



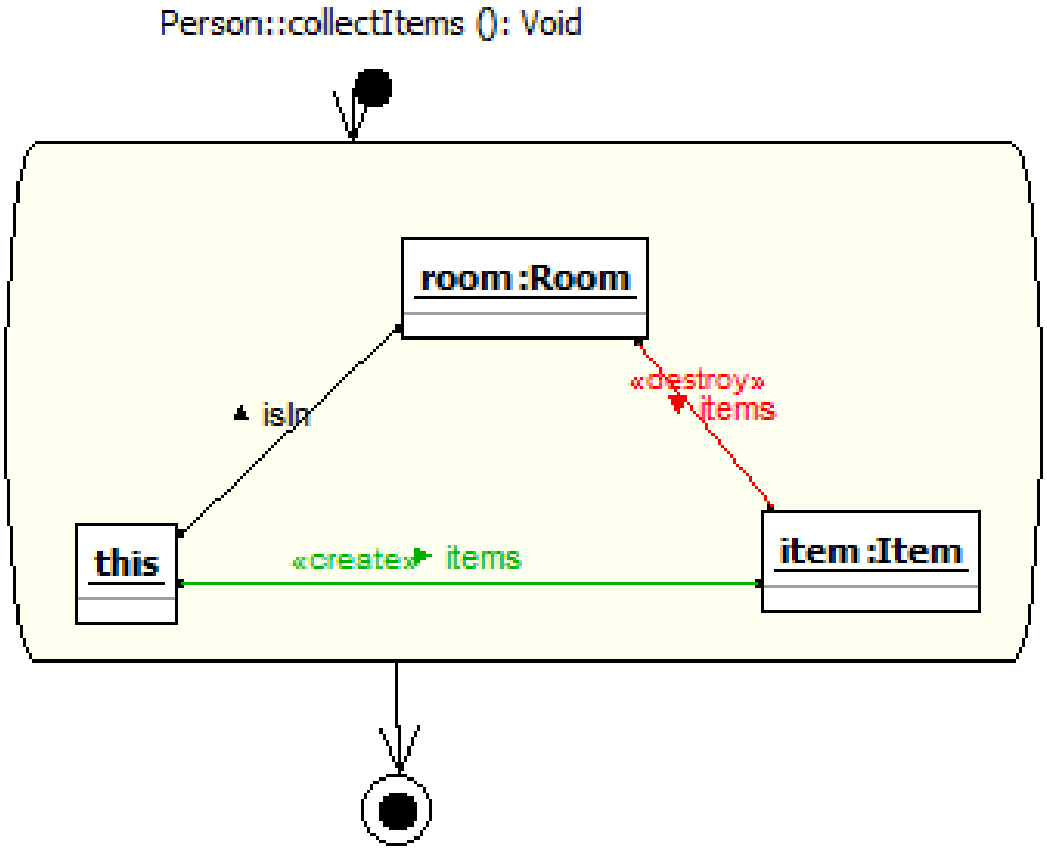
Storydiagramme

- **Objekte/Links erzeugen**
  - Zuerst werden Objekte erzeugt
  - Dann Links



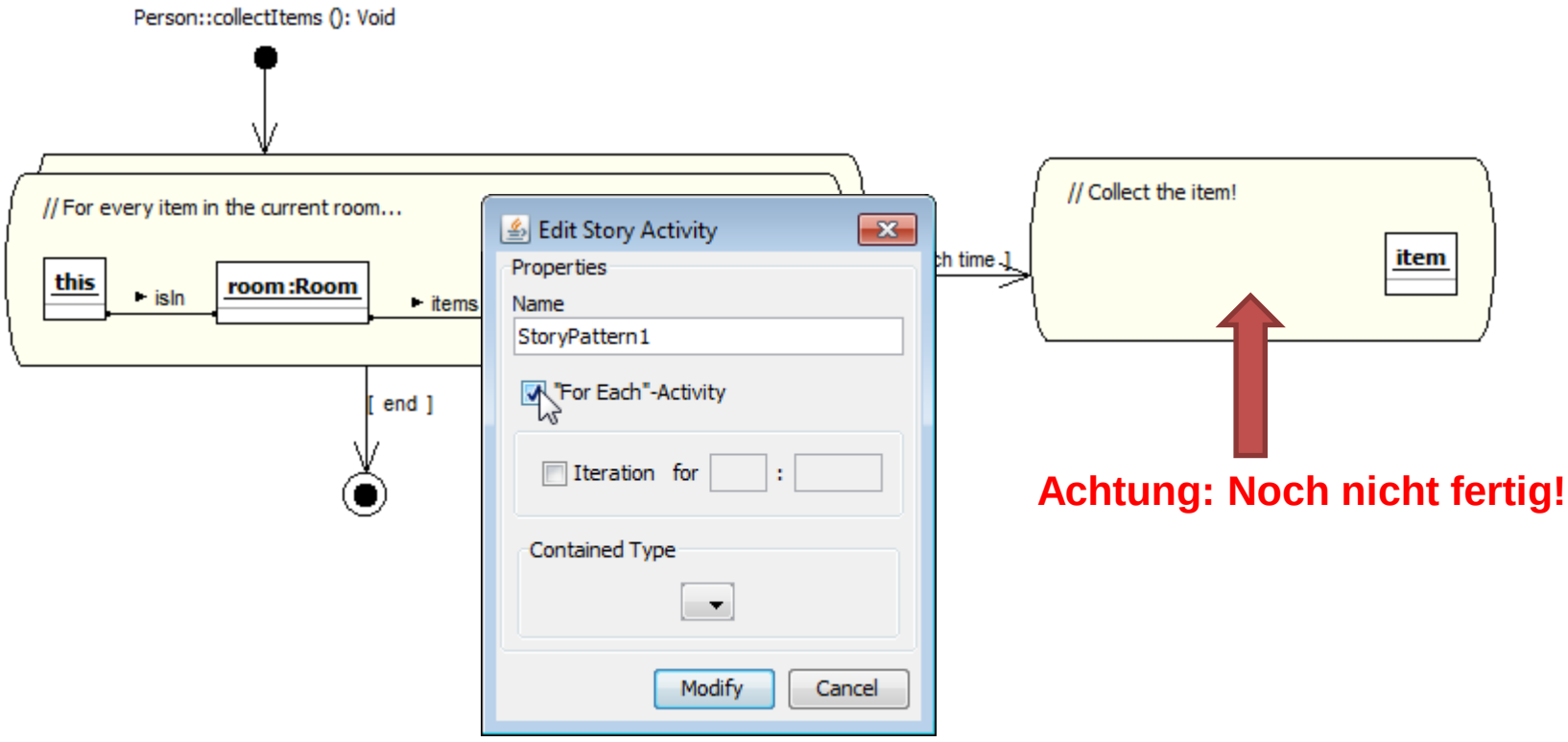
Storydiagramme

- **Objekte/Links zerstören**

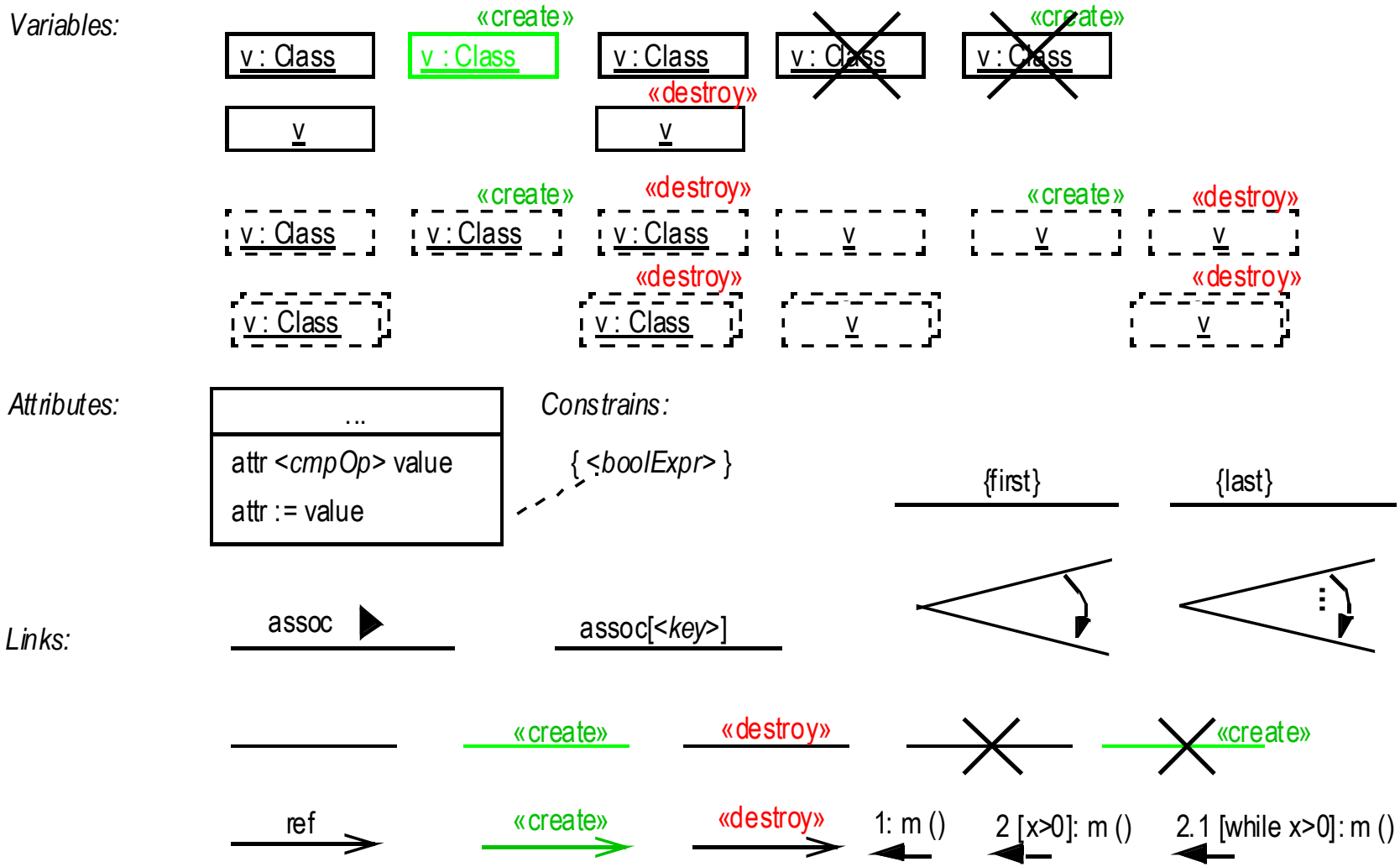


Storydiagramme

- **Mengen -> ForEach**



Rule Syntax: Overview



Wiederholung Zetteltest I

- **Zu prüfender Code**

```
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**

Wiederholung Zetteltest II

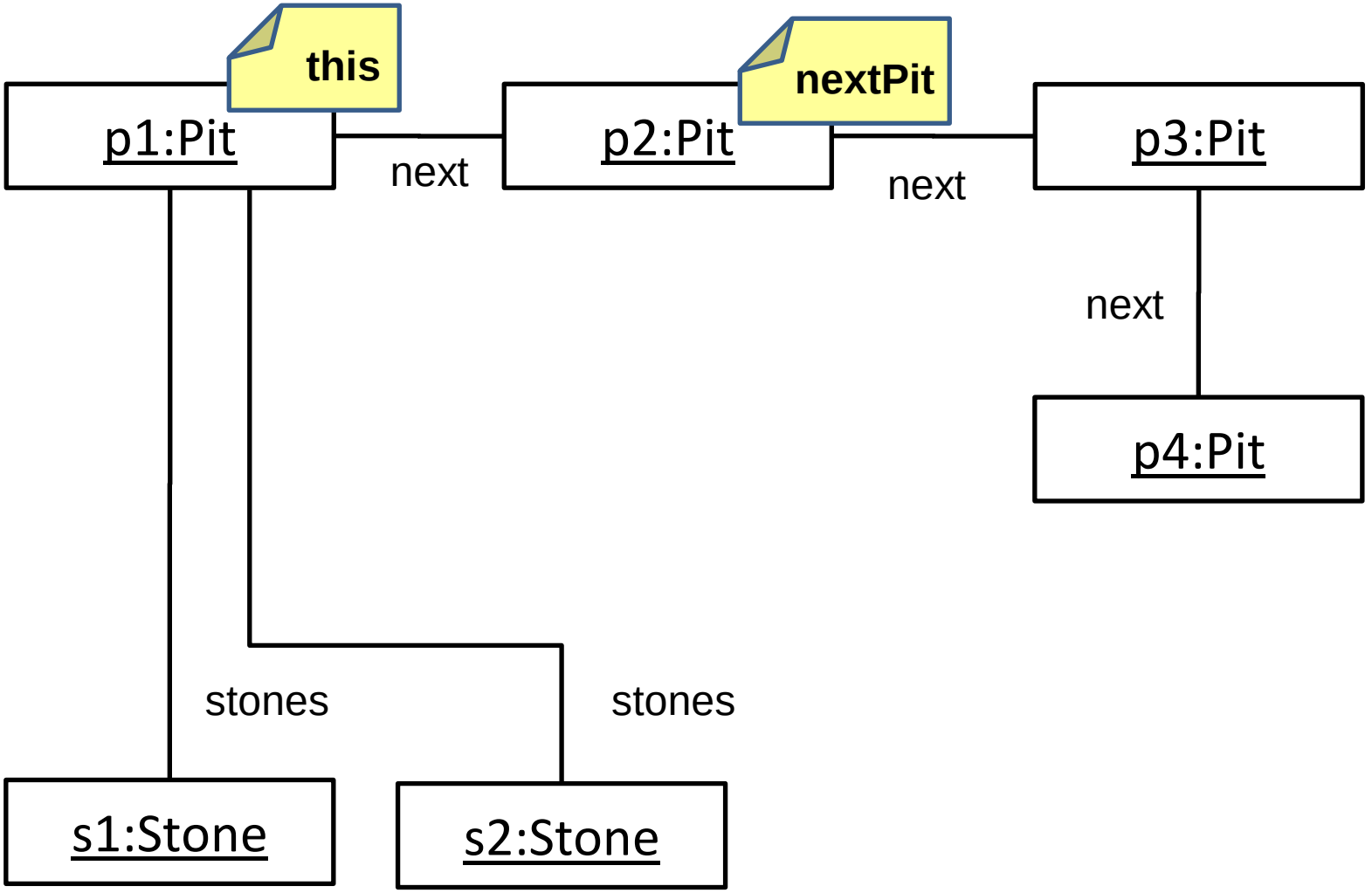
- **Zetteltest**
  - Hilft Methoden zu verstehen
  - Fehler aufzudecken
  - Ist „manuelles Debuggen“
- **Zetteltest Methode `moveStones():void` der Klasse `Pit`**

Wiederholung Zetteltest III

- **Generierter Code**

```
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 }
```

Wiederholung Zetteltest IV

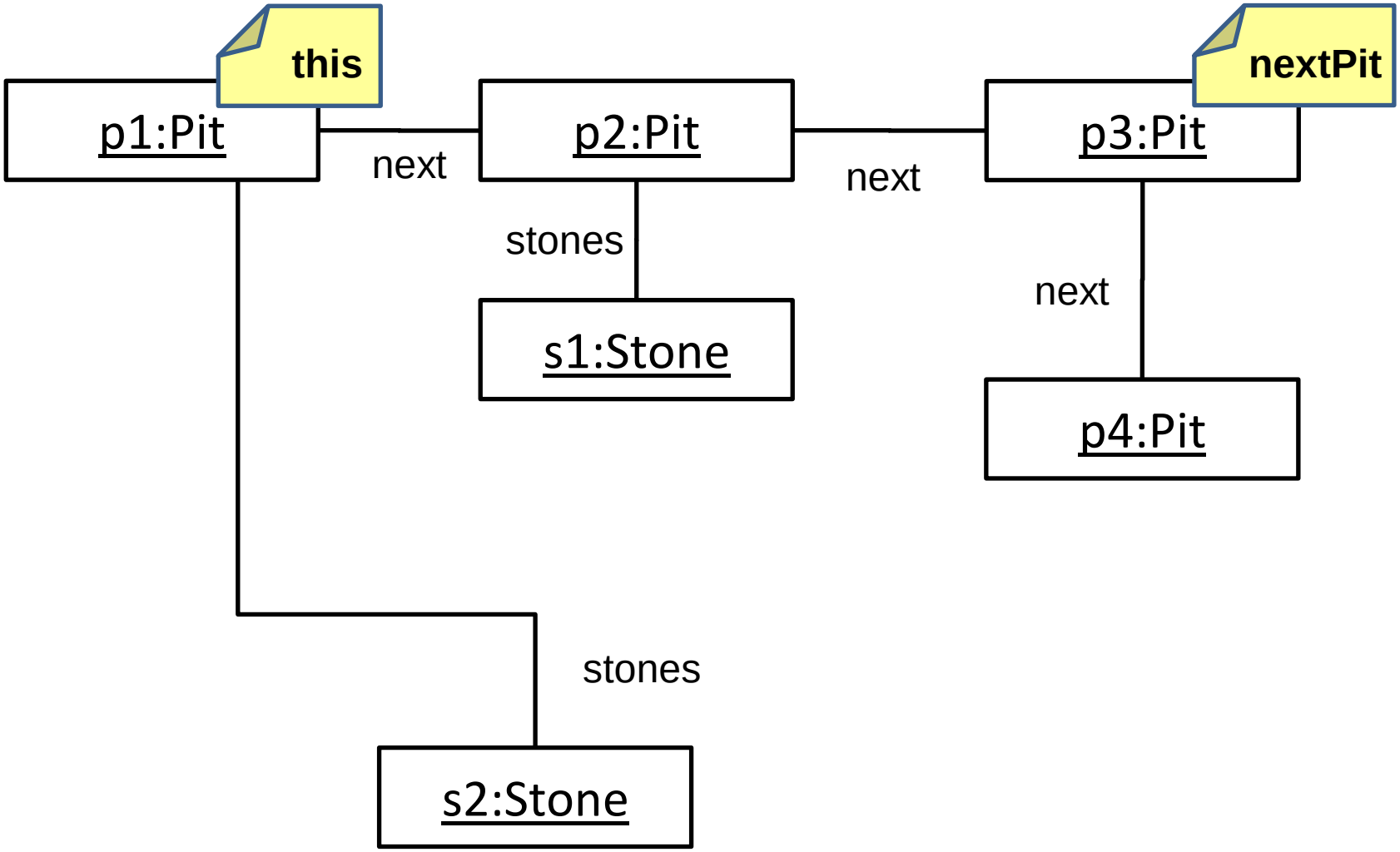


Wiederholung Zetteltest V

- **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 }
```

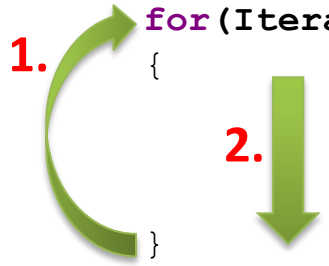
Wiederholung Zetteltest VI



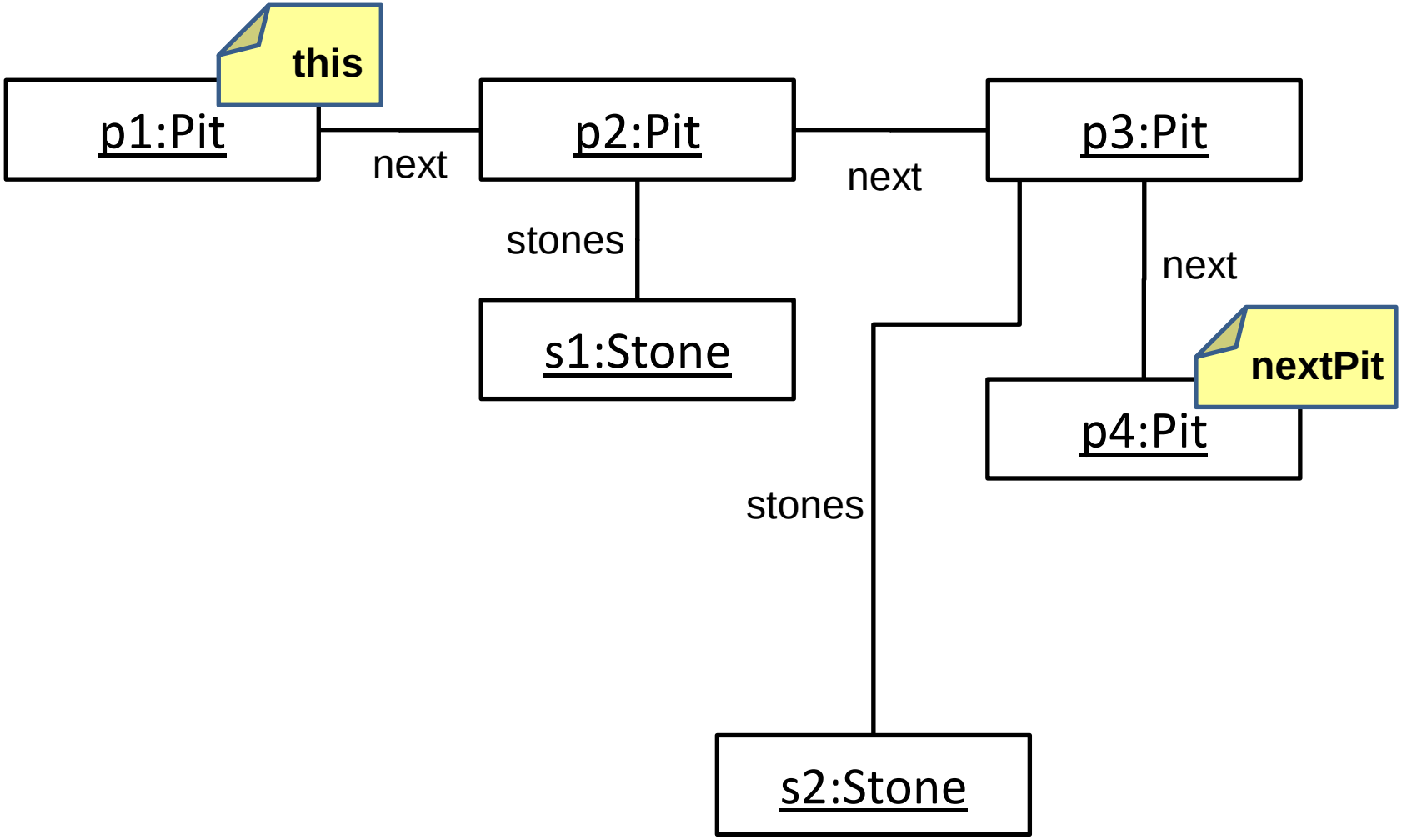
Wiederholung Zetteltest VII

- **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 }
```



Wiederholung Zetteltest VIII

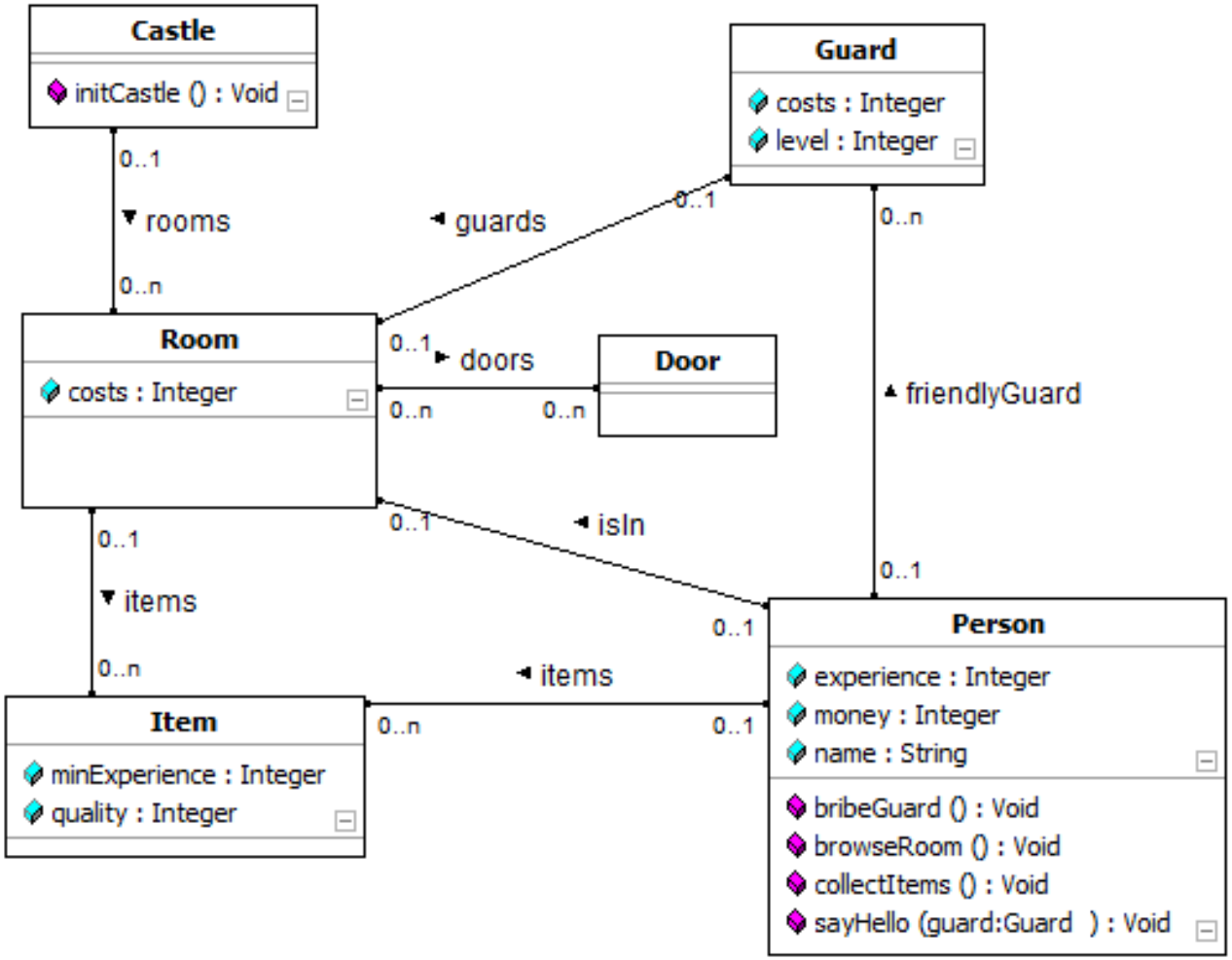


Vorschau HA8

- **Deadline: 19.01.2012, 23:59 Uhr**
- **Aufgabe 1:**
  - Zwei Storydiagramme:
    - Witch.move()
    - Game.checkWinner()
- **Aufgabe 2:**
  - Zetteltest zu Witch.move()

Praktische Übung

- **Castle**

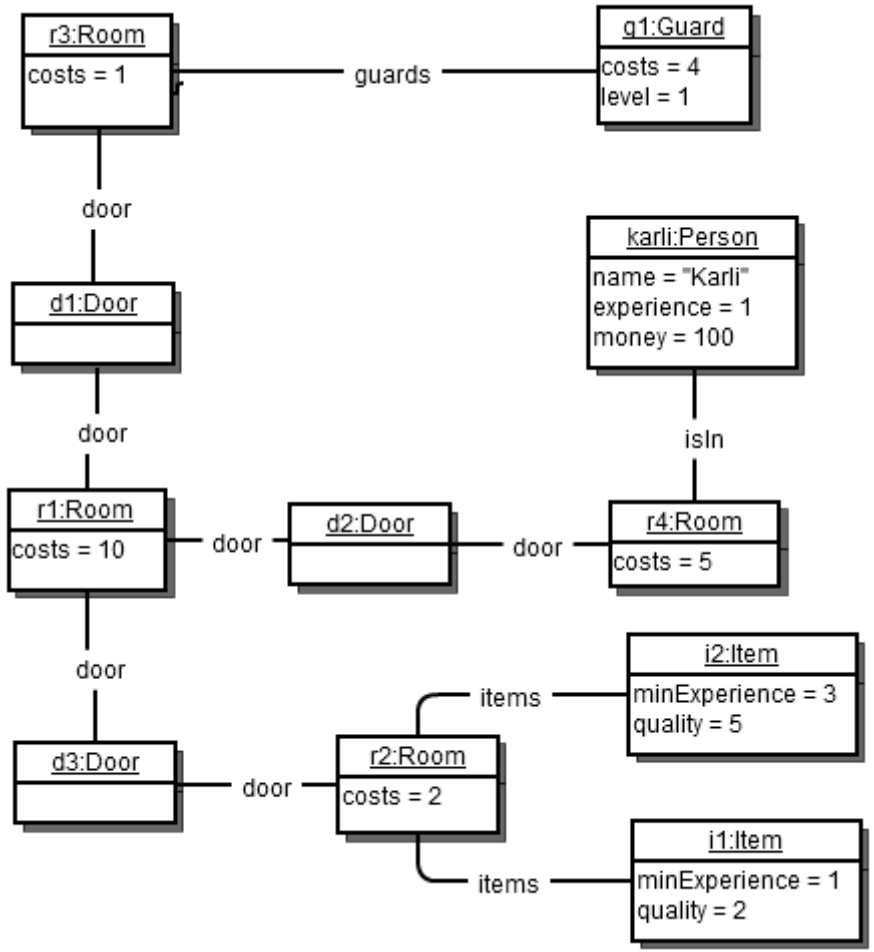


Praktische Übung

- **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 ein Storydiagramm für die Methode Castle::initCastle(), welche ein Schloss mit Räumen, Türen, Guards und Items erstellt. Orientiert euch dabei am Objekt-diagramm auf Folie 29
  - Erstellt ein Storyboard (Test) für die Methode Person::collectItems()
    - Die Methode soll Items aus dem aktuellen Raum einsammeln
      - Nur Items mit gleichem oder niedrigerem Level dürfen eingesammelt werden
    - Eingesammelte Items werden der Person zugeordnet und aus dem Raum entfernt
  - Erstellt ein Storydiagramm für die Methode Person::collectItems() und testet die Implementierung mit dem zuvor erstellten Storyboard

Ende

Jetzt: Betreutes Arbeiten

Ansonsten: Schönes WE!