UNIKASSEL VERSITÄT

Research Group Software Engineering

Programmiermethodik

Übung 5

Wintersemester 2011 / 12

Fachgebiet Software Engineering

Tobias George

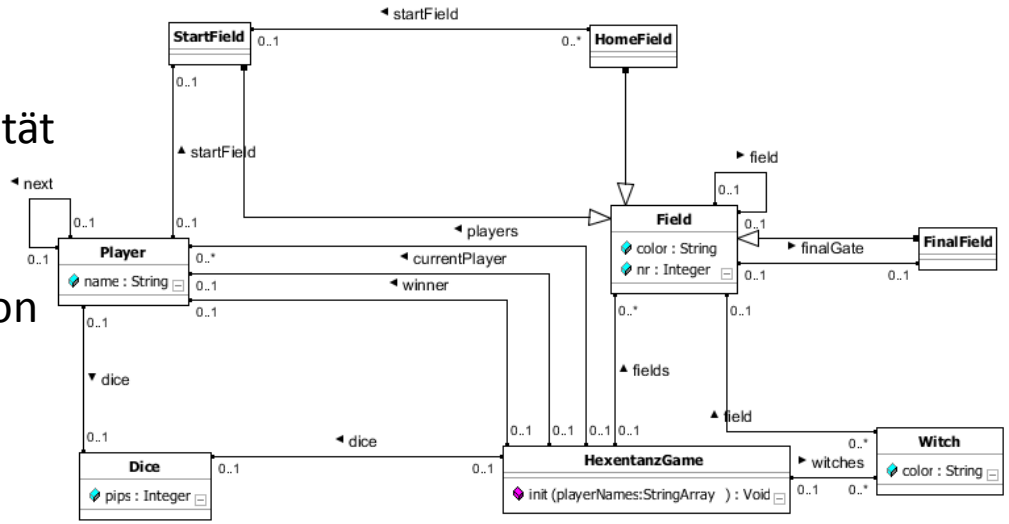
george@uni-kassel.de

Agenda

- **Besprechung HA 3**
- **Praktische Übung**
  - Fortsetzung Methodenentwurf
- **eDOBS**
- **Vorschau HA 4**

Besprechung HA3 I

- **Aufgabe 1: Implementierung Klassendiagramm**
  - Implementierung erfolgt strukturiert durch vorgegebenes Schema
    1. Klassen
    2. Attribute
    3. Methoden
    4. Assoziationen
  - Sichern von referentieller Integrität
- **Referentielle Integrität**
  - Immer bezüglich **einer** Assoziation



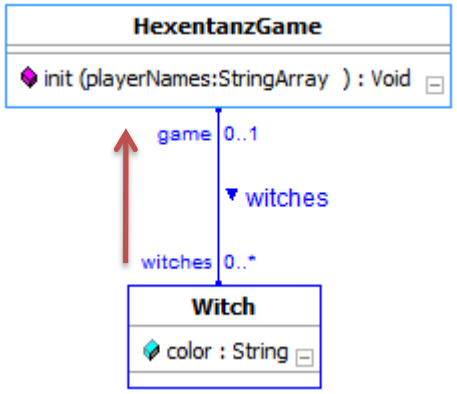
Besprechung HA3 II

- **Zusatzaufgabe: JUnit Tests**
- **Sicherstellen, dass referentielle Integrität korrekt implementiert ist. Beispiel:**


```
@Test
public void testHexentanzGameWitchReferentialIntegrity()
{
    // Create start situation
    HexentanzGame hexentanzGame = new HexentanzGame();
    Witch witch = new Witch();

    // Add the turn to the game
    hexentanzGame.addToWitches(witch);

    // Check referential integrity
    assertTrue("Witch was not added to hexentanz game", hexentanzGame.hasInWitches(witch));
    assertSame("Hexentanz game not found", hexentanzGame, witch.getHexentanzGame());

    // Remove the turn
    hexentanzGame.removeFromWitches(witch);

    // Check referential integrity
    assertFalse("Witch is still contained", hexentanzGame.hasInWitches(witch));
    assertNull("Witch still references the wizard game", witch.getHexentanzGame());
}
```

Besprechung HA3 III

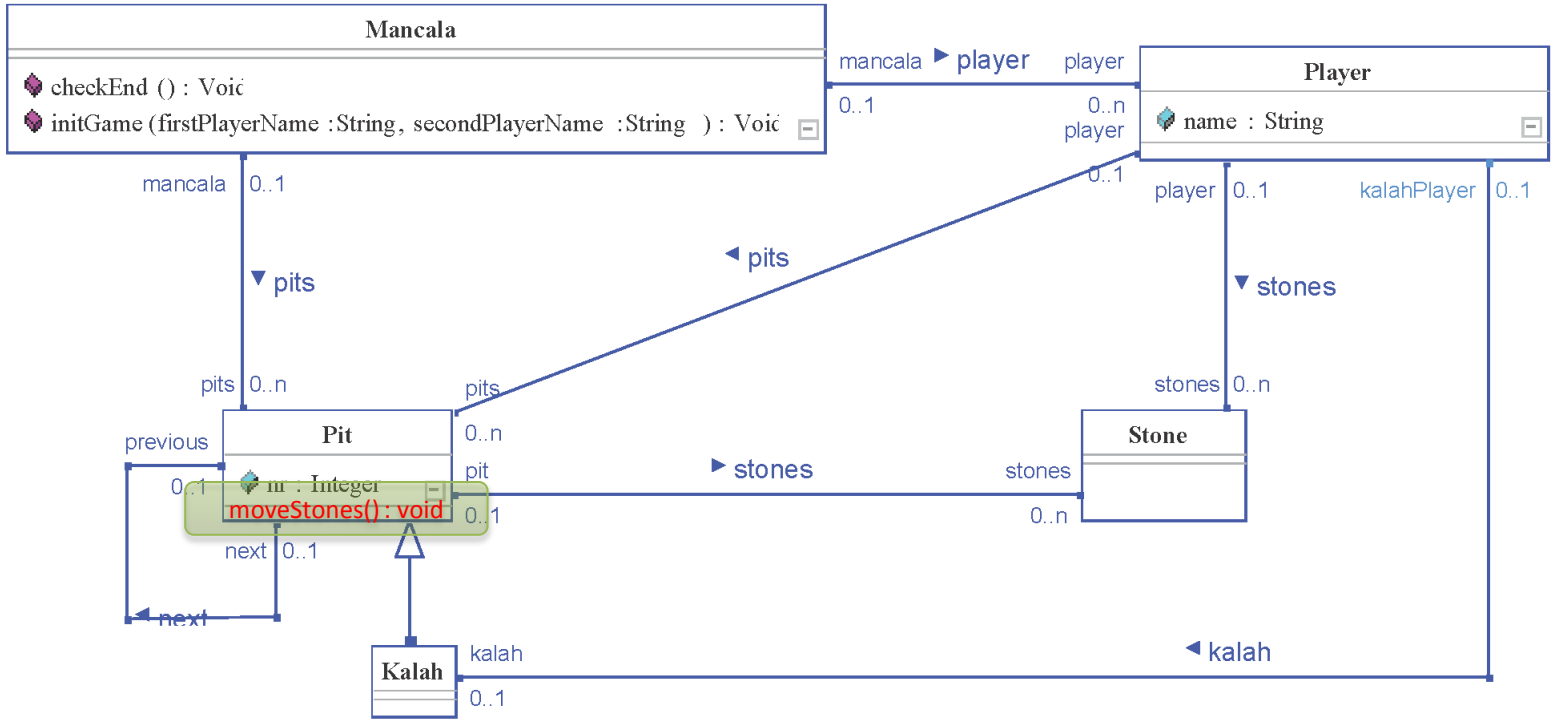
- **Rückrichtung:**




```
@Test
public void testWitchHexentanzGameReferentialIntegrity()
{
    // Create start situation
    HexentanzGame hexentanzGame = new HexentanzGame();
    Witch witch = new Witch();

    // Set wizard game
    witch.setHexentanzGame(hexentanzGame);

    // Check referential integrity
    assertTrue("Witch was not added to hexentanz game", hexentanzGame.hasInWitches(witch));
    assertSame("Hexentanz game not found", hexentanzGame, witch.getHexentanzGame());

    // Remove the game
    witch.setHexentanzGame(null);

    // Check referential integrity
    assertFalse("Witch is still contained", hexentanzGame.hasInWitches(witch));
    assertNull("Witch still references the wizard game", witch.getHexentanzGame());
}
```

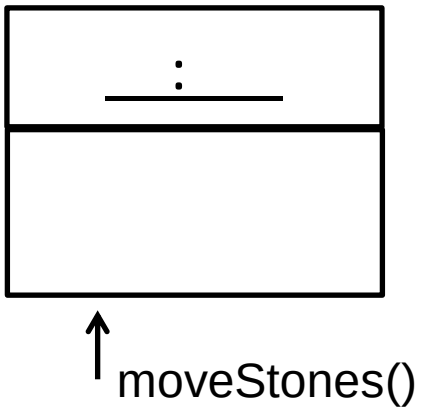
Methodenentwurf – Praktische Übung

- **Beispiel: „Mancala“**

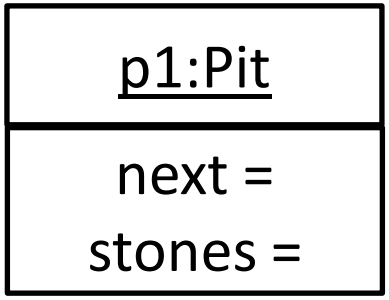


- **Textueller Methodenentwurf für: `moveStones():void` der Klasse `Pit`**

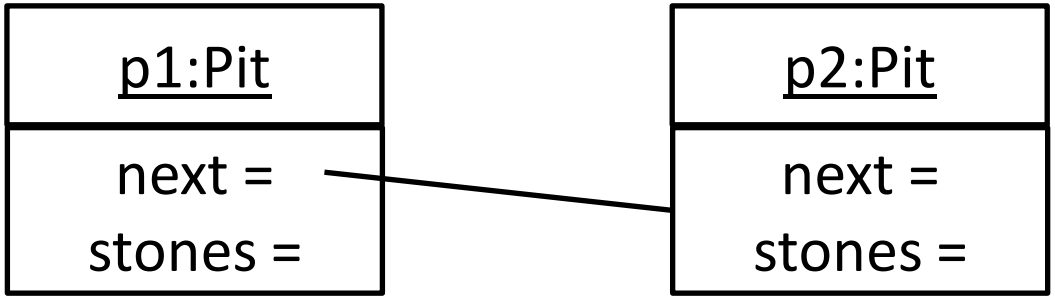
Methodenentwurf – Praktische Übung



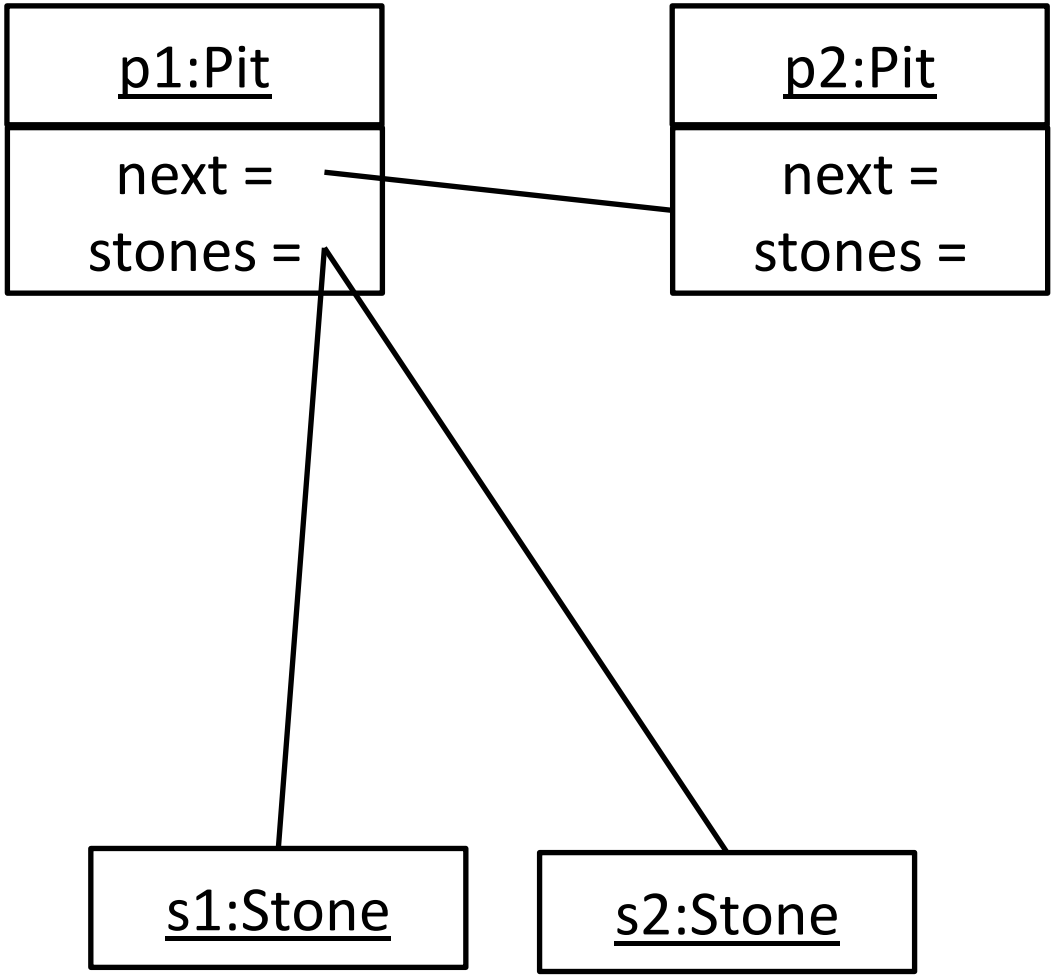
Methodenentwurf – Praktische Übung



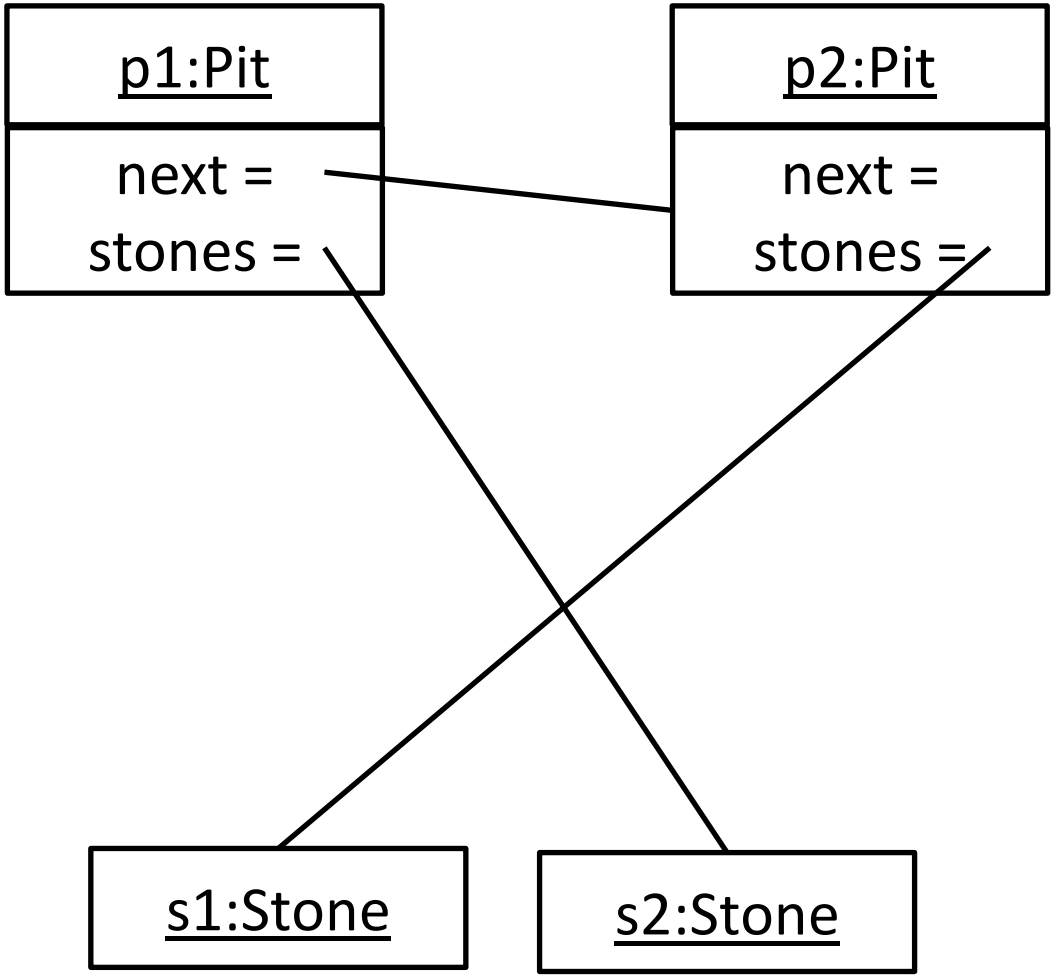
Methodenentwurf – Praktische Übung



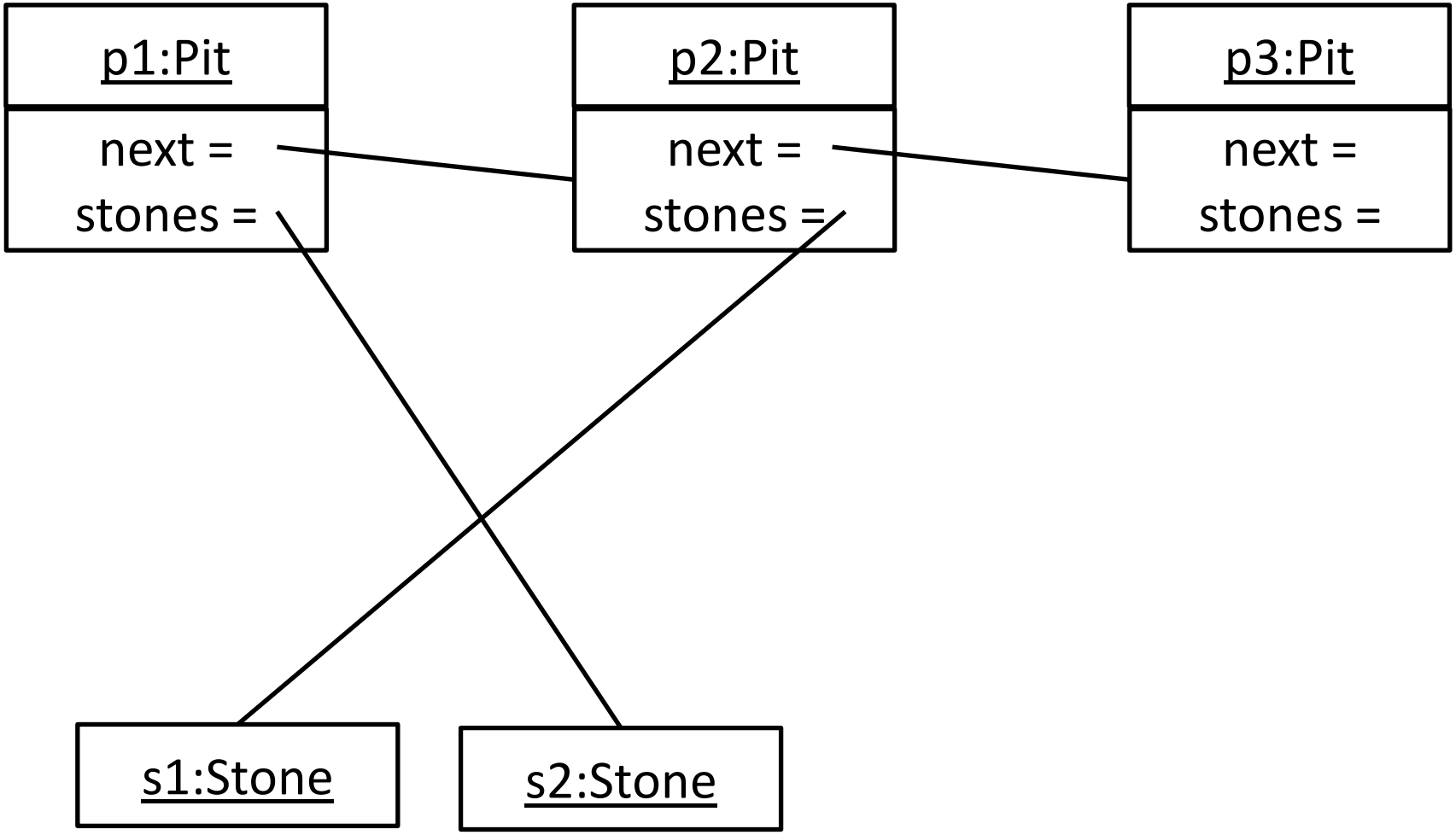
Methodenentwurf – Praktische Übung



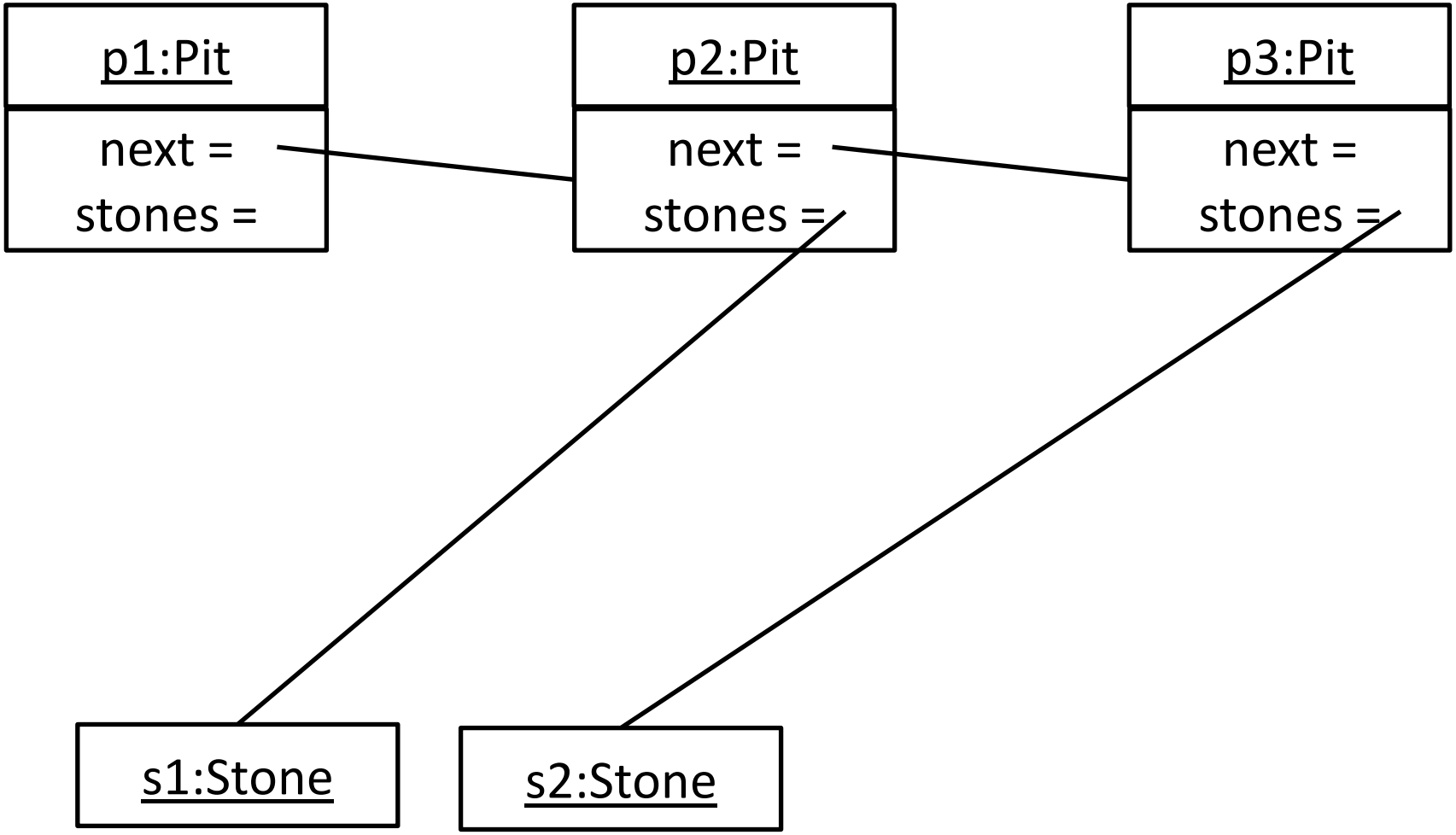
Methodenentwurf – Praktische Übung



Methodenentwurf – Praktische Übung



Methodenentwurf – Praktische Übung



Methodenentwurf I – Praktische Übung

- **Anforderung:**
  - Die Stones in einem Pit sollen auf die nachfolgenden Pits verteilt werden.
  - Tipp: Die Stones mit Iterator durchgehen, Pit.iteratorOfStones()
- **Mögliche Lösung:**
  1. Setze Variable „`nextPit`“ auf nächstes Pit
  2. Überprüfe ob Stones in aktuellem Pit vorhanden sind
  3. Für jeden Stone:
     1. Werfe Stone in „`nextPit`“
     2. Setze Variable „`nextPit`“ auf nächstes Pit

Methodenentwurf II – Praktische Übung

- Implementierung in Java:
- Eclipse Projekt vom Blog herunterladen
  - http://seblog.cs.uni-kassel.de/wp-content/uploads/2011/11/PMWS1112MancalaEclipseProject.zip
- In Eclipse importieren
- JUnit Test schreiben, der:
  - 3 Pits erzeugt und diese miteinander verbindet
  - 2 Stones in das erste Pit legt
  - Methode `moveStones()` auf dem ersten Pit aufruft
  - Testet, ob in dem ersten Pit kein Stone und in jedem anderen Pit ein Stone liegt
- Methode `moveStones():void` der Klasse `Pit` nach textuellem Entwurf implementieren

Methodenentwurf III

- **Mögliche Implementierung**

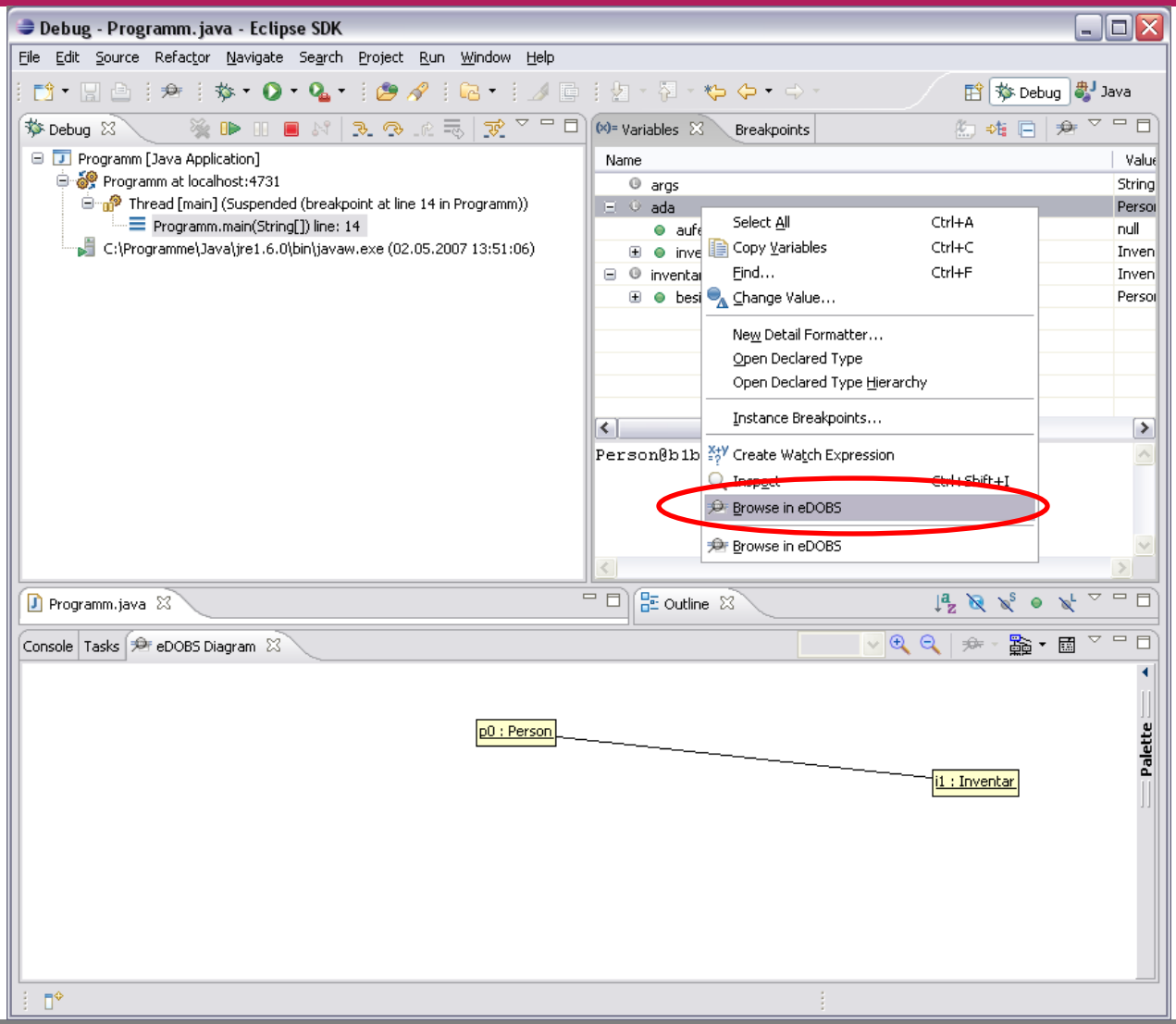
```
1  public void moveStones ()
2  {
3          Pit nextPit = getNext();
4          for(Iterator<Stone> iter = getIteratorOfStones(); iter.hasNext();)
5          {
6                  Stone stone = iter.next();
7                  nextPit.addToStones(stone);
8                  nextPit = nextPit.getNext();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

- **Verifikation durch Zetteltest in der nächsten Vorlesung/Übung**

eDOBS I

- **eDOBS (eclipse Dynamic Object Browsing System)**
  - Update Site: http://www.se.eecs.uni-kassel.de/se/fileadmin/se/update
  - Unter Required NUR: Fujaba4Eclipse Kassel Release auswählen
- **„Variables-View als Objektdiagramm“**
- **Zeigt die Objekte im Speicher (Java Heap) grafisch an**
- **Klassisches Objektdiagramm**
  - Instanzen, Attributbelegungen, Links
- **Erlaubt Interaktion mit den Objekten**
  - Ändern von Attributwerten
  - Aufruf von Methoden
  - Erzeugen/Löschen von Objekten und Links

eDOBS II

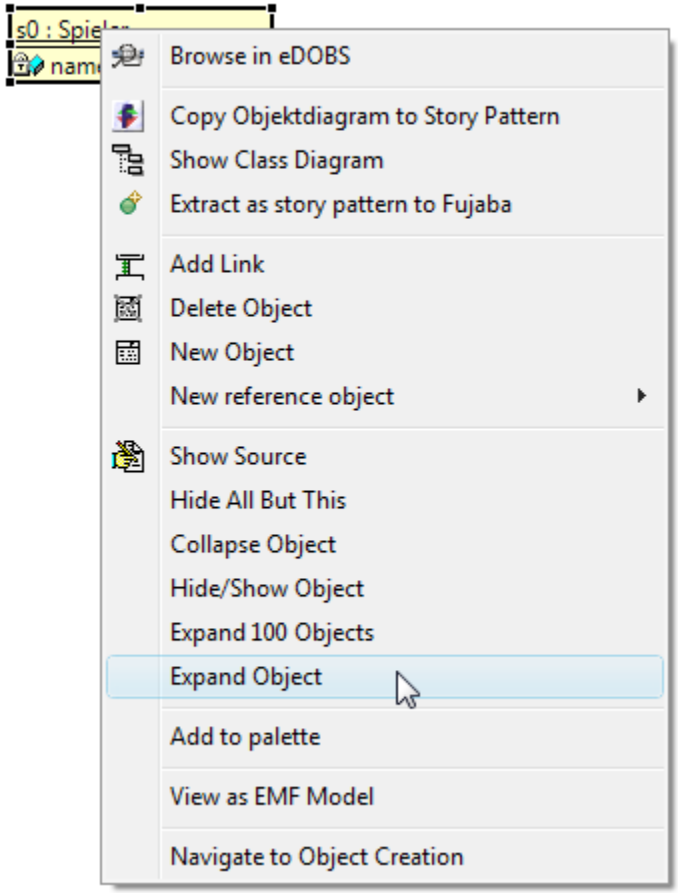

Debug - Programm.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
Debug
Programm [Java Application]
Programm at localhost:4731
Thread [main] (Suspended (breakpoint at line 14 in Programm))
Programm.main(String[]) line: 14
C:\Programme\Java\jre1.6.0\bin\javaw.exe (02.05.2007 13:51:06)
Variables
Breakpoints
Name
args
ada
Select All Ctrl+A
Copy Variables Ctrl+C
Find... Ctrl+F
Change Value...
New Detail Formatter...
Open Declared Type
Open Declared Type Hierarchy
Instance Breakpoints...
Create Watch Expression
Browse in eDOBS
Browse in eDOBS
Programm.java
Outline
Console
Tasks
eDOBS Diagram
p0 : Person
i1 : Inventar
Palette

eDOBS III

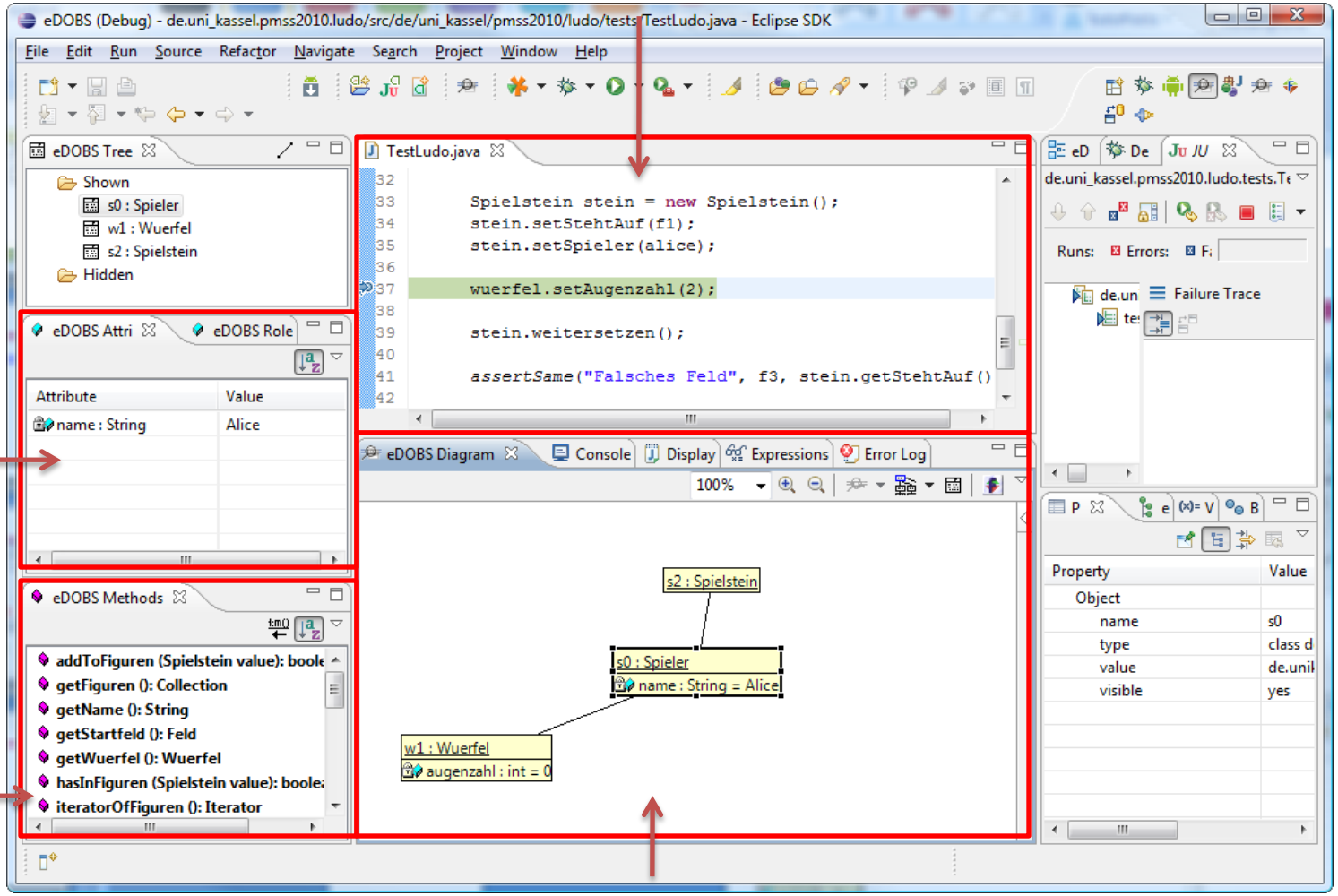
- **Breakpoint im Programm setzen**
- **Starte das Programm im Debug-Modus**
  - Debug as -> Java Application, NICHT Debug as eDOBS VM
- **Warte bis der Debugger auf dem Breakpoint hält**
- **Wähle eine Objekt-Variable im Variables-View (z.b. this), Rechtsklick, „Browse in eDOBS“**
- **Wechsle die Perspektive zu „eDOBS (Debug)“**

eDOBS IV

- **Objekte „expandieren“**
  - Zeige alle Nachbarn
  - Rechtsklick auf Objekt -> Expand



eDobs V


Quelltext
eDOBS (Debug) - de.uni_kassel.pmss2010.ludo/src/de/uni_kassel/pmss2010/ludo/tests/TestLudo.java - Eclipse SDK
File Edit Run Source Refactor Navigate Search Project Window Help
eDOBS Tree
Shown
s0 : Spieler
w1 : Wuerfel
s2 : Spielstein
Hidden
TestLudo.java
Spielstein stein = new Spielstein();
stein.setStehtAuf(f1);
stein.setSpieler(alice);
wuerfel.setAugenzahl(2);
stein.weitersetzen();
assertSame("Falsches Feld", f3, stein.getStehtAuf()
eDOBS Attri
eDOBS Role
Attribute
Value
name : String
Alice
eDOBS Methods
addToFiguren (Spielstein value): boole
getFiguren (): Collection
getName (): String
getStartfeld (): Feld
getWuerfel (): Wuerfel
hasInFiguren (Spielstein value): boole
iteratorOfFiguren (): Iterator
eDOBS Diagram
Console
Display
Expressions
Error Log
100%
s2 : Spielstein
s0 : Spieler
name : String = Alice
w1 : Wuerfel
augenzahl : int = 0
Runs:
Errors:
Failure Trace
Property
Value
Object
name
s0
type
class d
value
de.unil
visible
yes
Attribute anzeigen/ ändern
Methoden aufrufen
Objektdiagramm

Vorschau HA4 I

- **Deadline: 01.12.2011, 23:59 Uhr**
- **Vorbereitung:**
  - eDOBS installieren
  - Hexentanz Projekt herunterladen (optional)
    - Enthält bereits Implementierung des Klassendiagramms

      und den Rumpf Methode:

      `Hexentanz::init(String[] playerNames):void`

Vorschau HA4 II

- **Aufgabe 1**
  - `HexentanzGame::init(…):void` implementieren
    - Für jeden per Parameter übergebenen Namen einen Spieler erzeugen
    - Alle 64 Felder (16 Heimfelder, 4 Startfelder, 4 Abflugfelder, 16 Zielfelder, 24 normale Felder)
    - 4 Hexen pro Spieler auf den zugehörigen Heimfeldern
    - 1 Würfel
- **Aufgabe 2**
  - eDOBS Screenshot am Ende von
    - `HexentanzGame::initGame(…):void`

Ende

Jetzt: Betreutes Arbeiten

Ansonsten: Schönes WE!