UNIKASSEL VERSITÄT

SOFTWARE ENGINEERING
UNIVERSITÄT KASSEL

Programmiermethodik

Übung 5

Wintersemester 2011 / 12

Fachgebiet Software Engineering

Tobias George

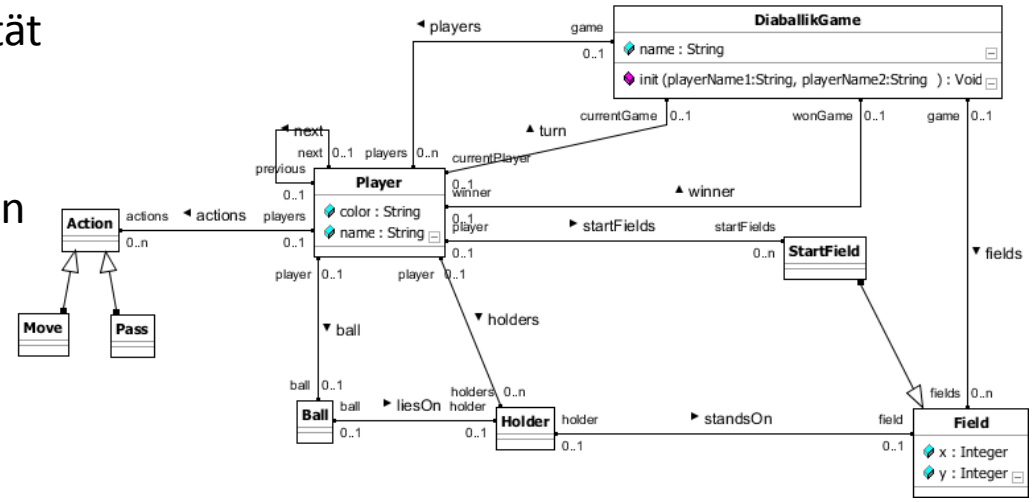
george@uni-kassel.de

Agenda

- **Besprechung HA 3**
- **Rückblick: Implementierung eines Klassendiagramms von Hand**
- **SDMLib**
- **Fujaba4Eclipse**
- **Praktische Übung**
  - Risiko Klassendiagramm in Fujaba/SDMLib erstellen und Code generieren
- **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

- **Aufgabe 2 (Zusatz): JUnit Tests**
- **Sicherstellen, dass referentielle Integrität korrekt implementiert ist. Beispiel:**

```
@Test
public void testDiaballikGamePlayerReferentialIntegrity ()
{
    // Create start situation
    DiaballikGame diaballikGame = new DiaballikGame();
    Player player = new Player();

    // Add the player to the game
    diaballikGame.addToPlayers(player);

    // Check referential integrity
    assertTrue("Player was not added to diaballik game",
               diaballikGame.hasInPlayers(player));
    assertSame("Diaballik game not found", diaballikGame, player.getDiaballikGame());

    // Remove the player
    diaballikGame.removeFromPlayers(player);

    // Check referential integrity
    assertFalse("Player is still contained in diaballikGame",
               diaballikGame.hasInPlayers(player));
    assertNull("Player still references the diaballik game",
               player.getDiaballikGame());
}
```

Rückblick: Implementierung eines KD von Hand

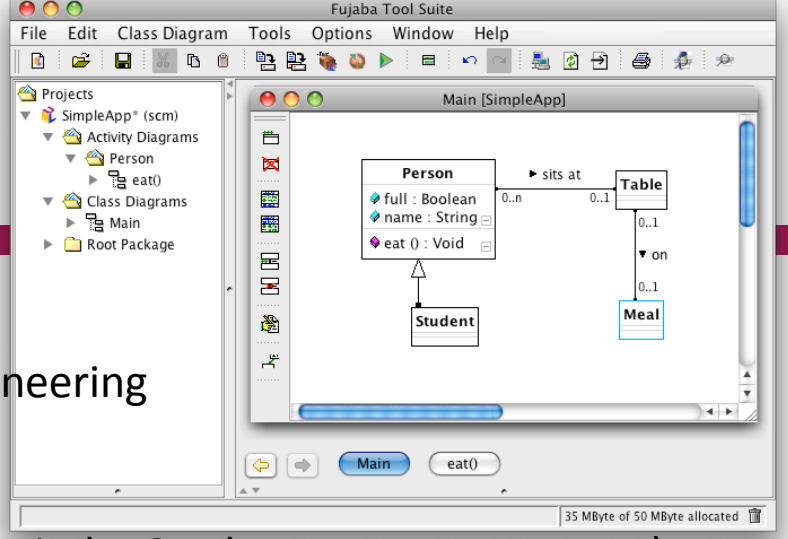
- Dauert lang
- Fehleranfällig (z.B. bezüglich der Referenziellen Integrität)
- „Man schreibt ständig das gleiche!“

Das lässt sich automatisieren: Codegenerierung!

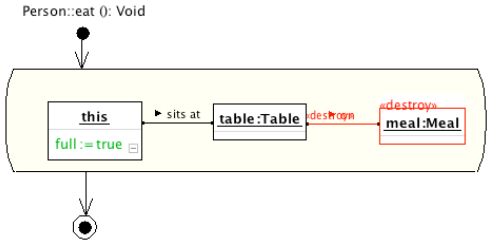
SDMLib

- **Means: Story Driven Modeling Library**
  - Intention: Code Generation and Reverse Engineering
- **Erlaubt u.a.:**
  - Das Erstellen eines Datenmodells (statische Struktur von Programmen)
  - Das Erstellen von Storyboards
  - Quellcode
  - Diagramme generieren zu lassen
- **Heute: Klassendiagram-Anteil!**

Fujaba & Fujaba4Eclipse

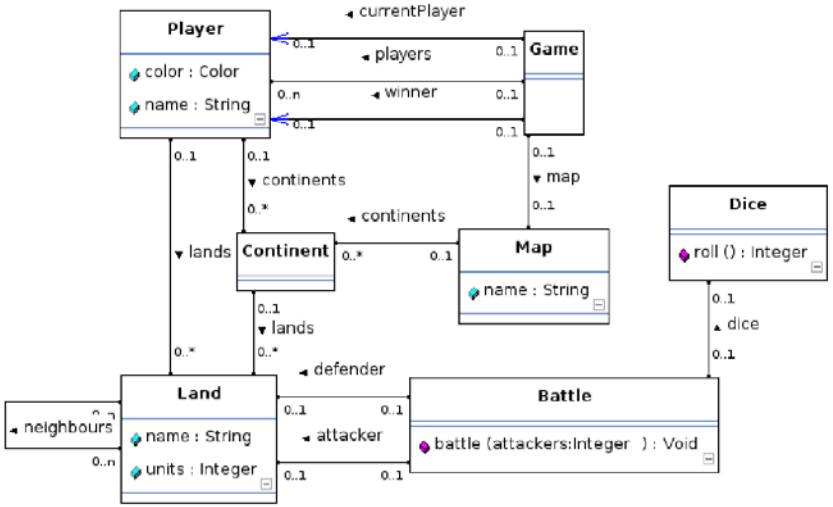


- **Means: From UML to Java And Back Again**
  - Intention: Code generation and Reverse Engineering
- **Erlaubt u.a.:**
  - Das Erstellen von UML Klassendiagrammen (statische Struktur von Programmen)
  - Das Erstellen von Story- bzw. Aktivitätsdiagrammen (zur Modellierung von Verhalten)
  - Aus Diagrammen (Java, EMF, C++...) Code zu generieren



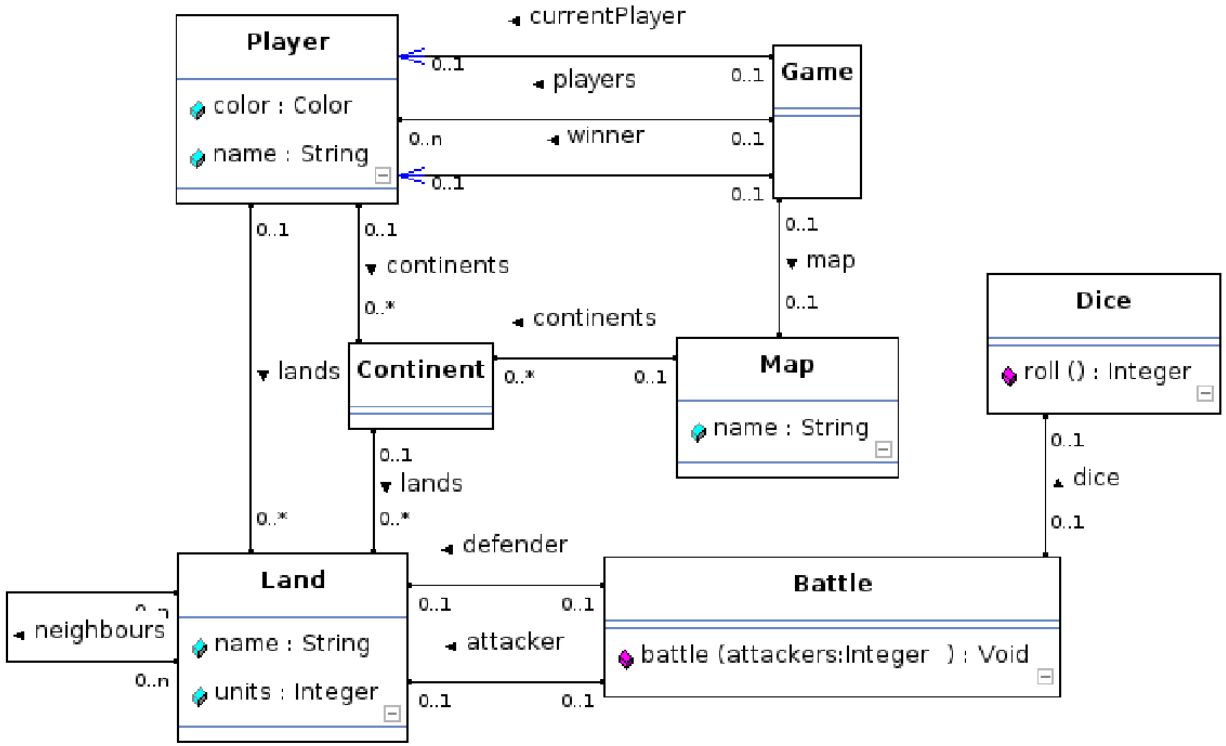
- **Heute: Klassendiagram-Anteil!**
- **Fujaba4Eclipse: Eclipse Plugin → Fujaba mit Eclipse Integration**

Praktische Übung: Risiko Klassendiagramm



Praktische Übung

- Erstellt das Klassendiagramm zu dem Brettspiel Risiko in Fujaba/SDMLib und generiert Code
- Zeigt das Ergebnis einem Betreuer



Vorschau HA4 I

- **Deadline: 29.11.2011, 23:59 Uhr**
- **Vorbereitung:**
  - Gerade Matrikelnummer: Fujaba4Eclipse Plugin installieren bzw.
  - Ungerade Matrikelnummer: SDMLib Projekt herunterladen, importieren und in das eigene Projekt einbinden

Vorschau HA4 II

- **Aufgabe 1**
  - Gerade Matrikelnummer: Diaballik Klassendiagramm mit Fujaba4Eclipse modellieren

  bzw.

  - Ungerade Matrikelnummer: Diaballik Klassendiagramm mit SDMLib modellieren
- **Aufgabe 2**
  - Methode DiaballikGame::initGame(…):void

  implementieren die den Initialzustand herstellt:

  2 Spieler,

  15 Felder,

  2x 5 Startfelder,

  2X 5 Holder,

  2X 1 Ball

Ende

Jetzt: Betreutes Arbeiten

Ansonsten: Schönes WE!