UNIKASSEL
VERSITÄT

Research Group
Software Engineering

Programmiermethodik

Übung 5

Sommersemester 2011

Fachgebiet Software Engineering

Andreas Scharf

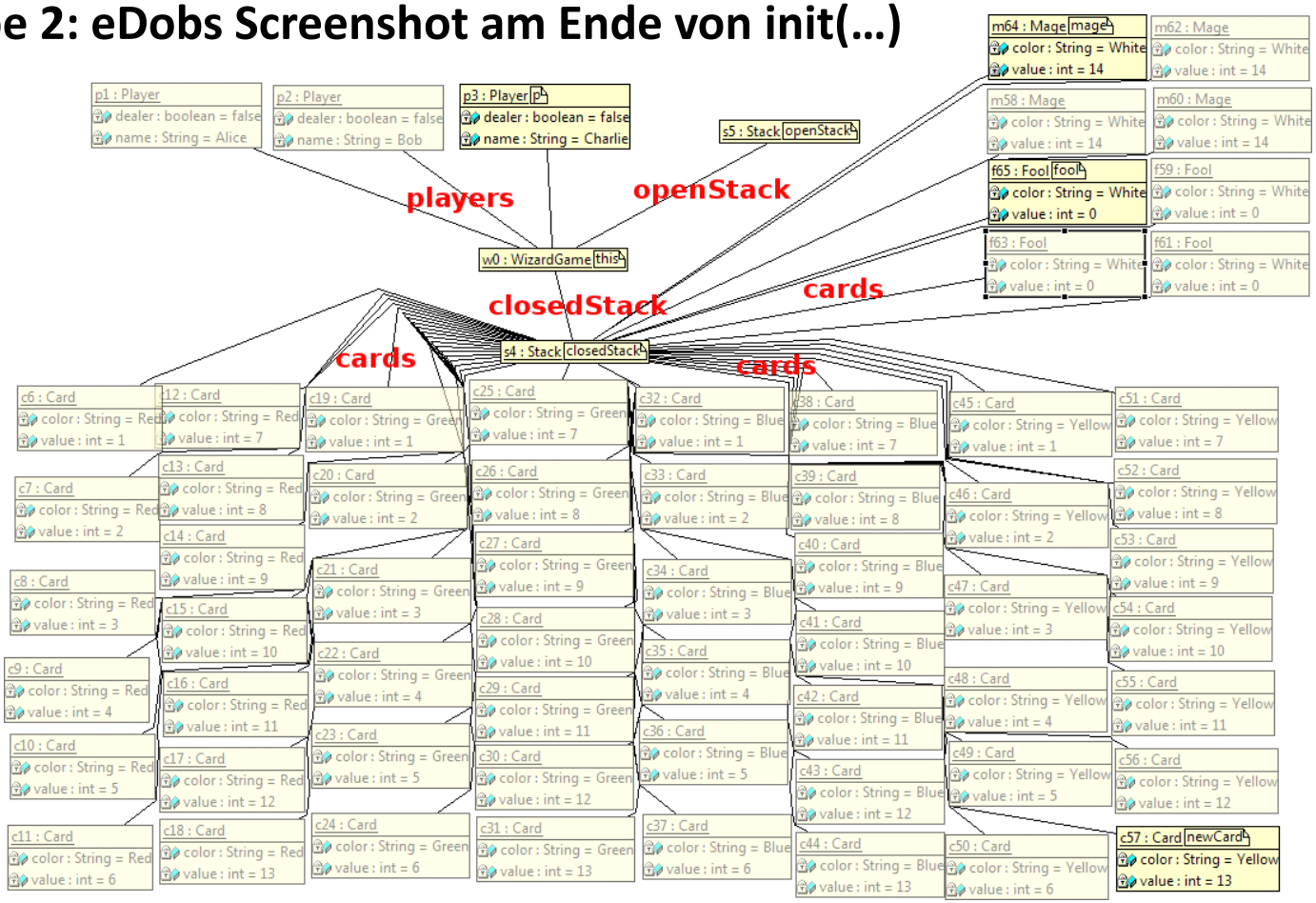
andreas.scharf@cs.uni-kassel.de

Agenda

- **Besprechung HA 3**
- **Änderungen am Wizard Klassendiagramm**
  - Beispiel-Objektstruktur
- **Fujaba 4 Eclipse: Storyboards**
- **Vorstellung HA 4**
- **Praktische Übung: Storyboards „Stich evaluieren“**

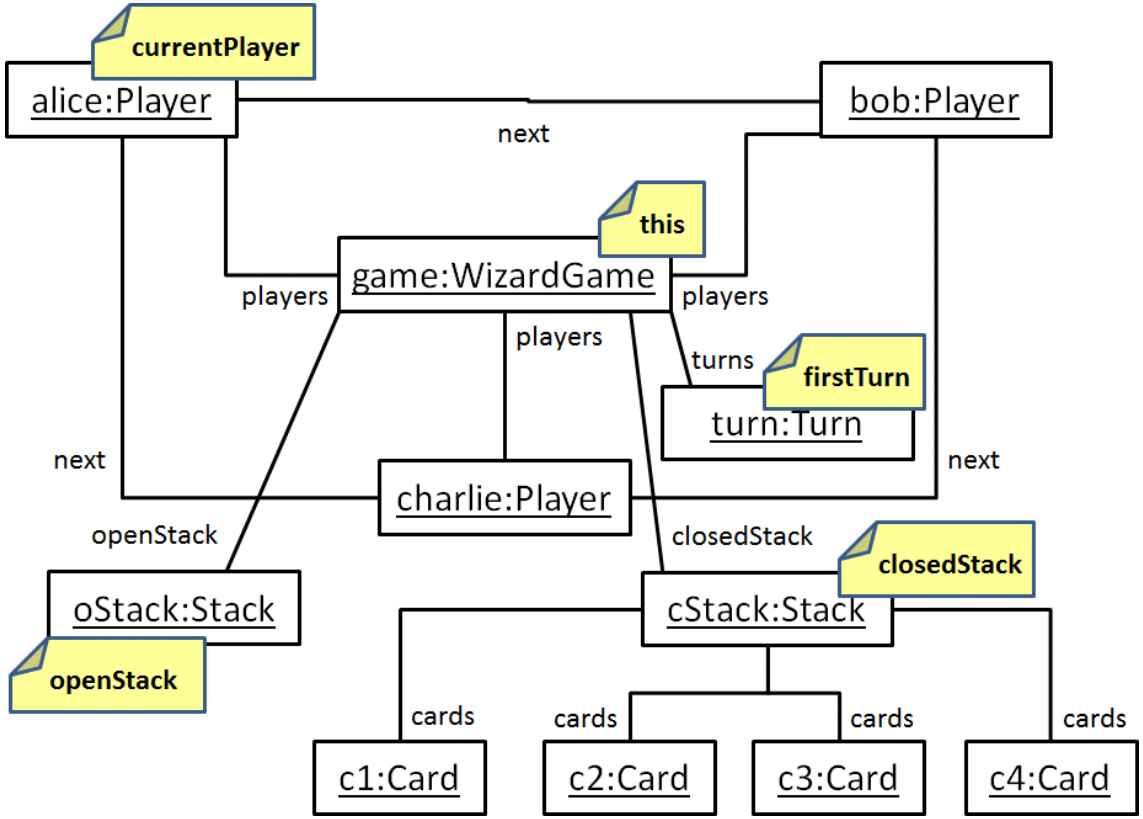
Besprechung HA 3 I

- **Aufgabe 1: WizardGame::init(playerNames:StringArray) implementieren**
- **Aufgabe 2: eDobs Screenshot am Ende von init(…)**



Besprechung HA 3 II

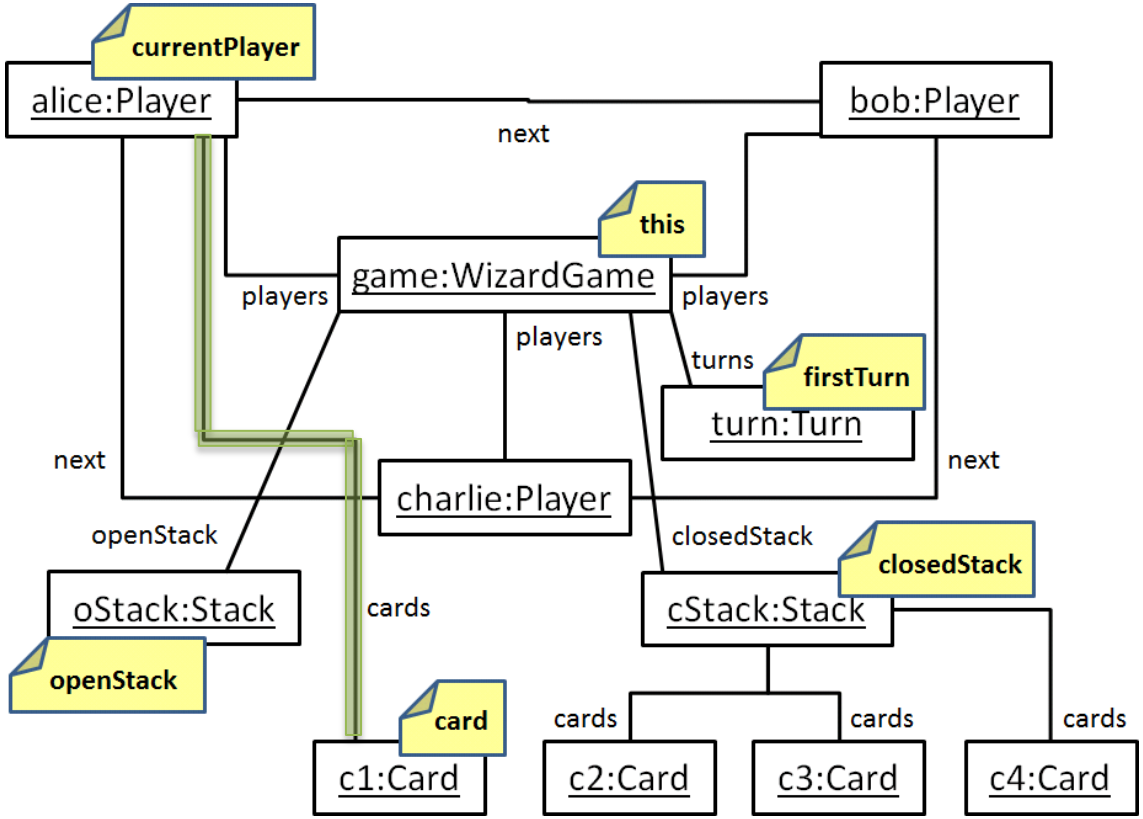
- **Aufgabe 3.1: Zetteltest zu WizardGame::startGame():void**



Schritt 1 (vorgegeben)

Besprechung HA 3 III

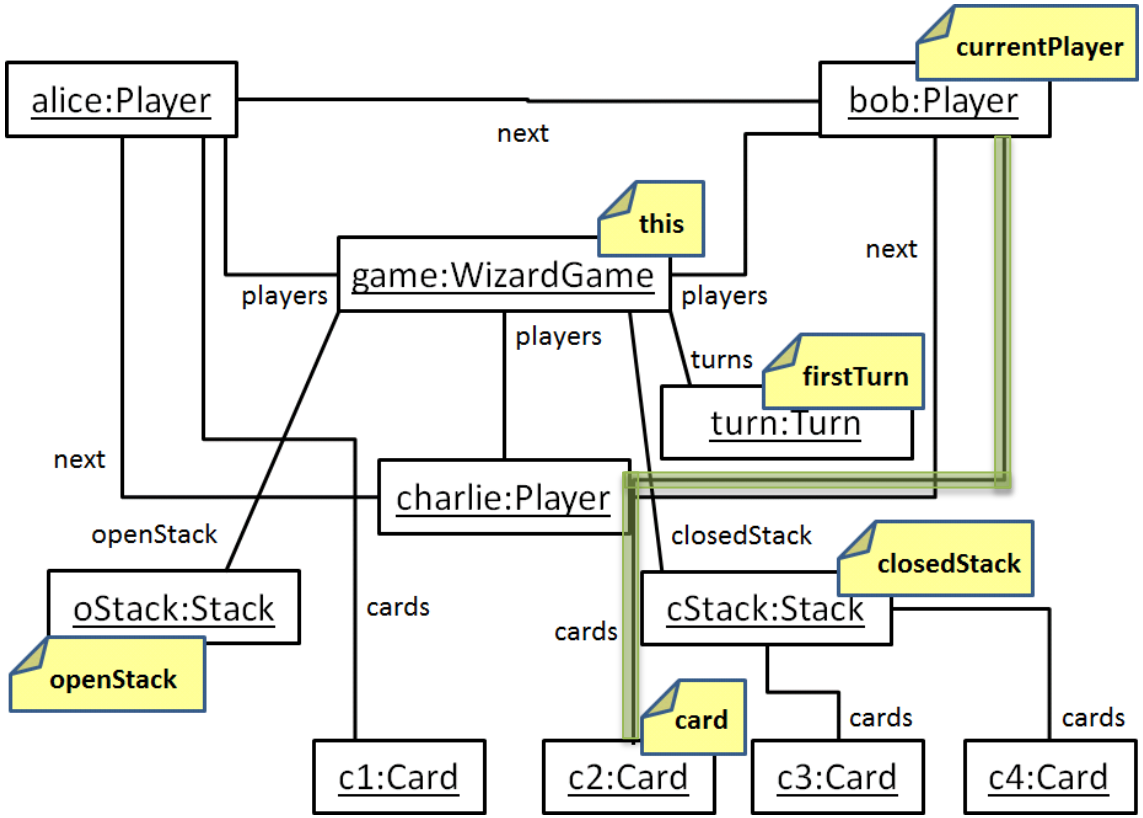
- **Aufgabe 3.1: Zetteltest zu WizardGame::startGame():void**



Schritt 2

Besprechung HA 3 IV

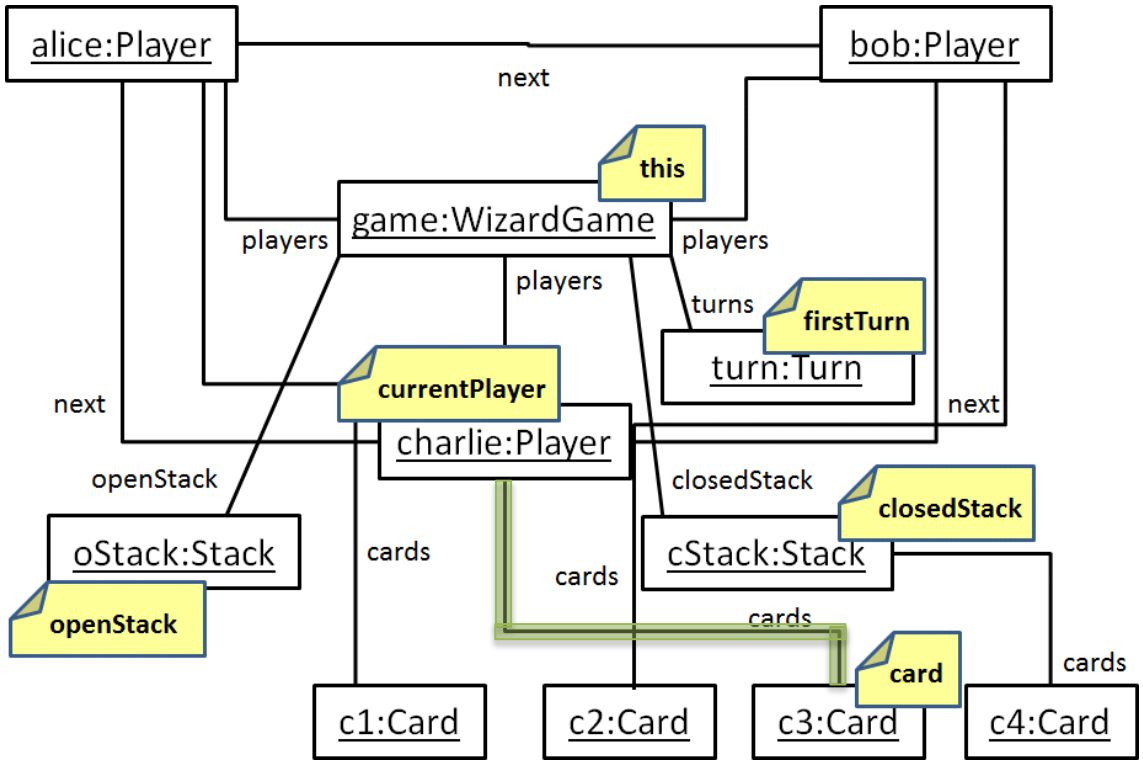
- **Aufgabe 3.1: Zetteltest zu WizardGame::startGame():void**



Schritt 3

Besprechung HA 3 V

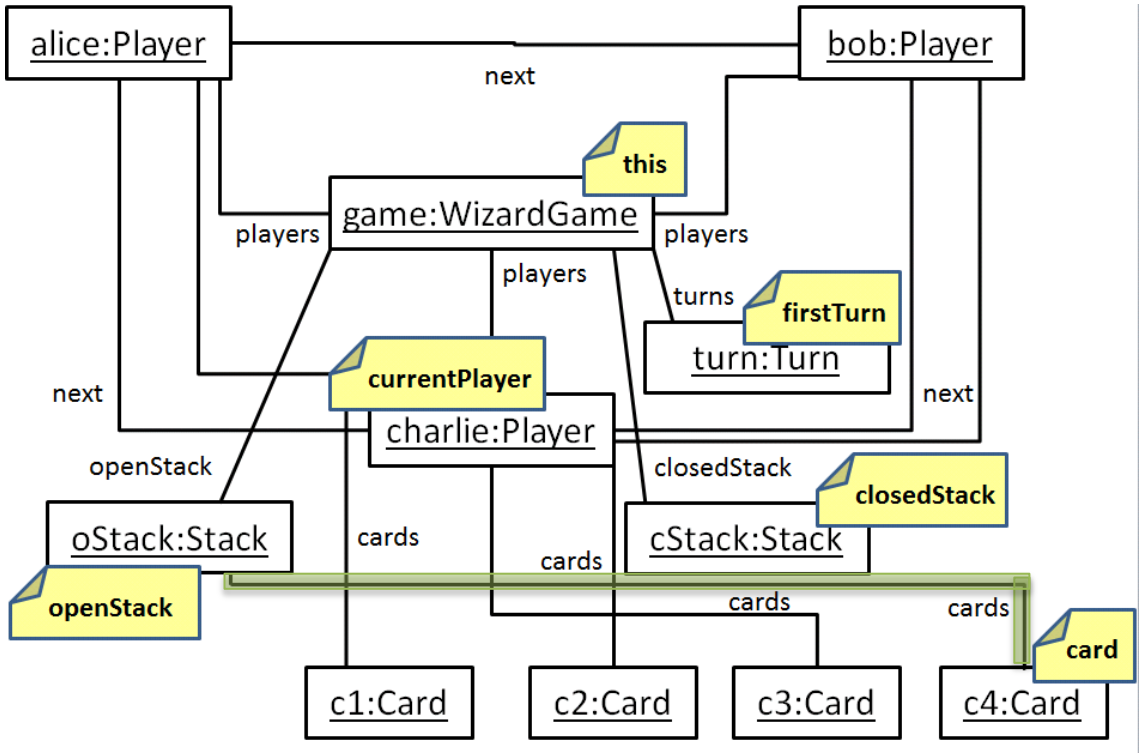
- **Aufgabe 3.1: Zetteltest zu WizardGame::startGame():void**



Schritt 4

Besprechung HA 3 VI

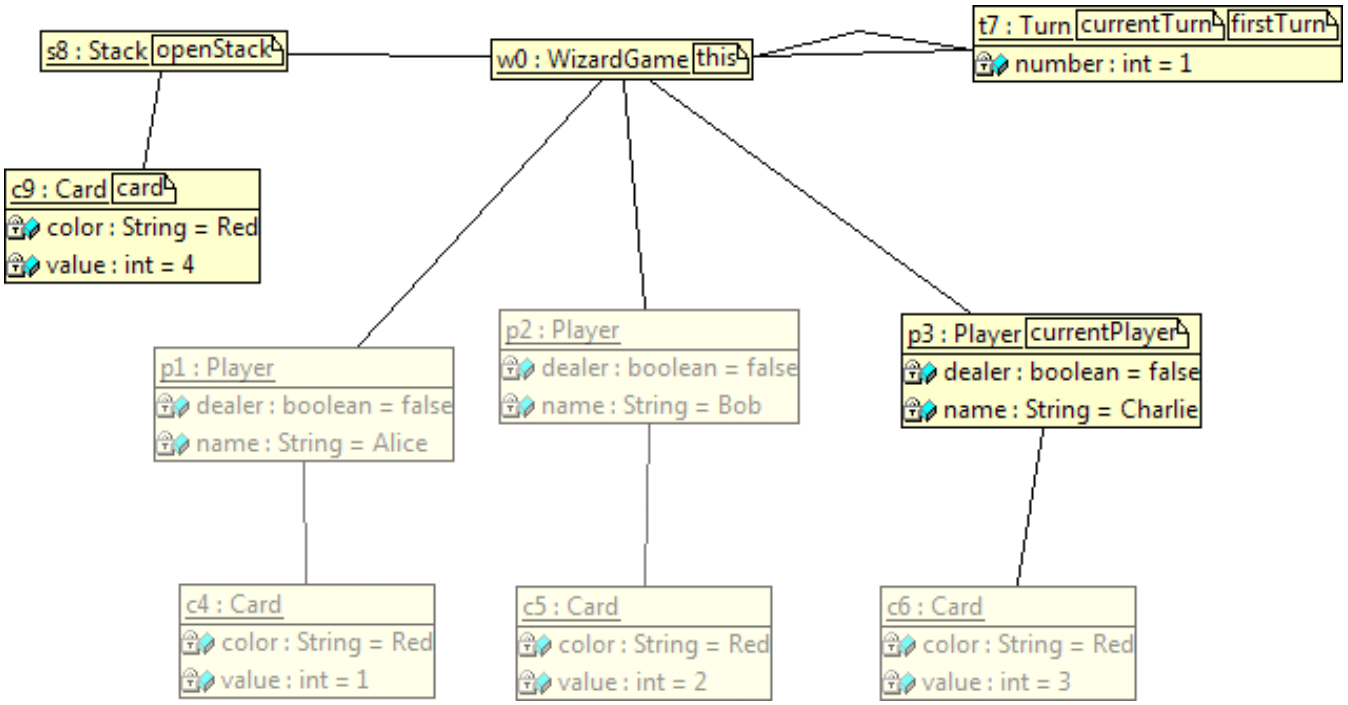
- **Aufgabe 3.1: Zetteltest zu WizardGame::startGame():void**



Schritt 5 (Endsituation)

Besprechung HA 3 VII

- **Aufgabe 3.2: eDobs Screenshot am Ende von WizardGame::startGame():void**



Besprechung HA 3 VIII

- **Zusatzaufgabe: Implementierung erweitern**
  - JUnit Test der prüft, ob die Vorhersagen der ersten Runden korrekt eingesammelt wurden
  - WizardGame::startGame():void soll zusätzlich die Spieler auffordern ihre Vorhersage abzugeben
  - Player::doForecast():void implementieren
    - Forecast Objekt erstellen und dem Spieler zuweisen
    - Forecast Objekt mit der der aktuellen Runde verbinden
    - Forecast::trickcount einen beliebigen Wert zuweisen (<= momentane Runde)

Besprechung HA 3 IX

- **JUnit Test**

```
// Create startsituation
WizardGame game = new WizardGame();
game.init(new String[]{"Alice", "Bob", "Charlie"});

// Action!
game.startTurn(1);

// Check if the forecasts are correct
Turn currentTurn = game.getCurrentTurn();
for (Player player : game.getPlayers())
{
   assertEquals("Player " +player.getName() +" didn't make a forecast!", 1,
                                                    player.getForecasts().size());
   Forecast forecast = player.getForecasts().iterator().next();
   assertSame("Turn is wrong!", currentTurn, forecast.getTurn());
   assertTrue("Forecast is really strange!", forecast.getTrickcount() <= currentTurn.getNumber());
}
```

Besprechung HA 3 X

- **Erweiterung der WizardGame::startGame():void Methode**

```
public void startGame ()
{
   …
   // Put one card on the open stack
   card = getClosedStack().popCard();
   getOpenStack().addToCards(card);

   // Let the players make their forecasts
   for (Player player : getPlayers())
   {
      player.doForecast();
   }
}
```

Besprechung HA 3 XI

- **Implementierung von Player::doForecast():void (z.B.)**

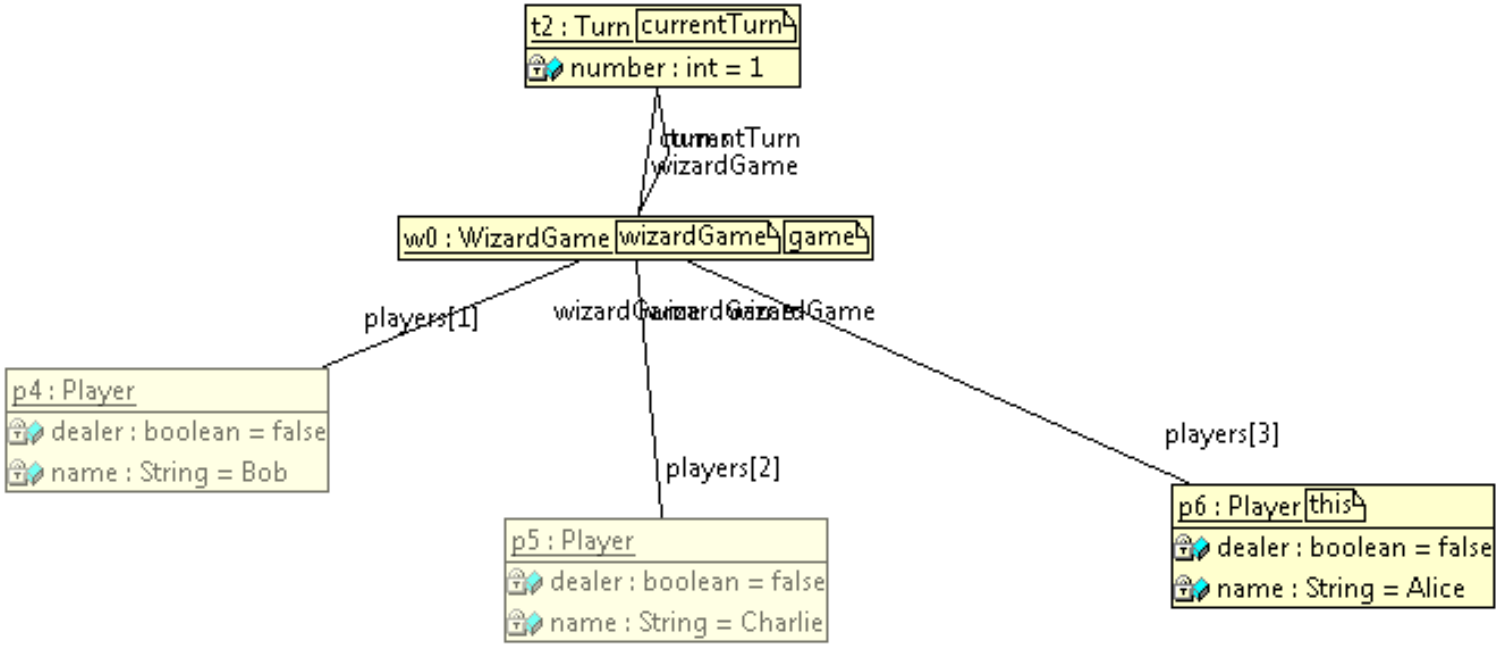
```
public void doForecast ()
{
   WizardGame game = getWizardGame();
   Turn currentTurn = game.getCurrentTurn();

   Forecast forecast = new Forecast();
   int randomTrickcount = (int)((Math.random()*100)%currentTurn.getNumber()) + 1;
   forecast.setTrickcount(randomTrickcount);
   addToForecasts(forecast);

   currentTurn.addToForecasts(forecast);
}
```

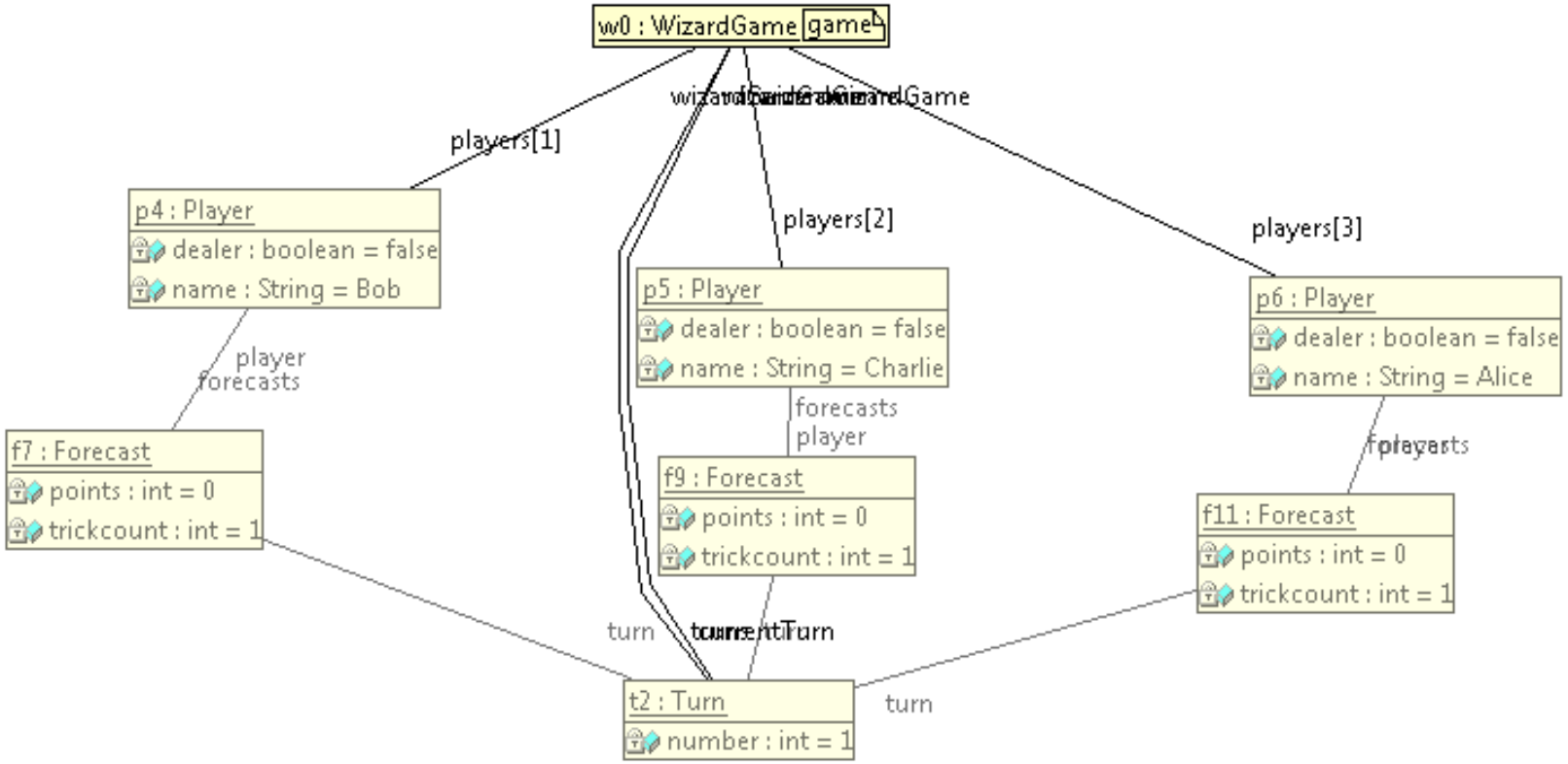
Besprechung HA 3 XII

- **eDobs Screenshot VOR der Abgabe der Vorhersagen**

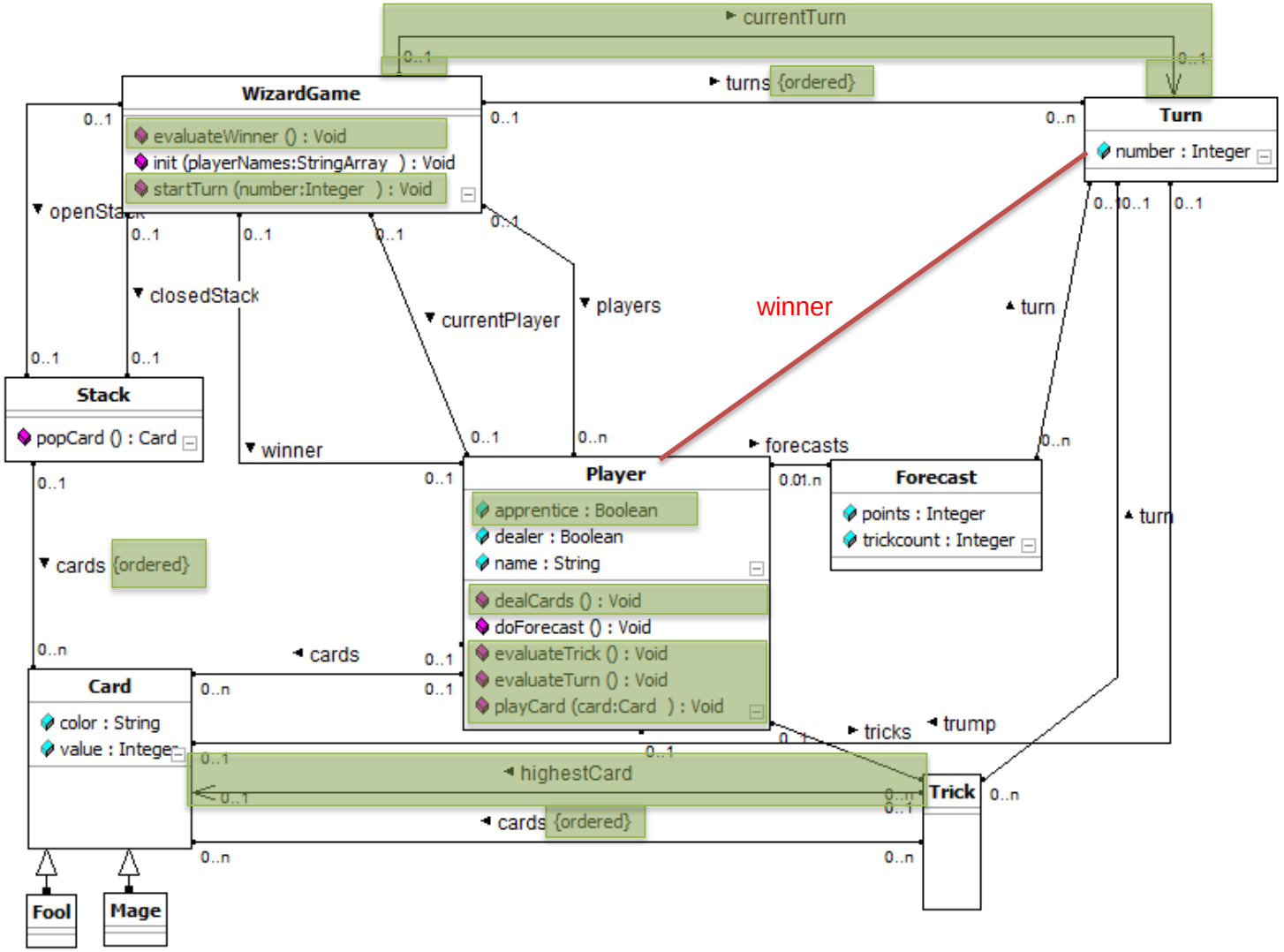


Besprechung HA 3 XIII

- **eDobs Screenshot NACH der Abgabe der Vorhersagen**

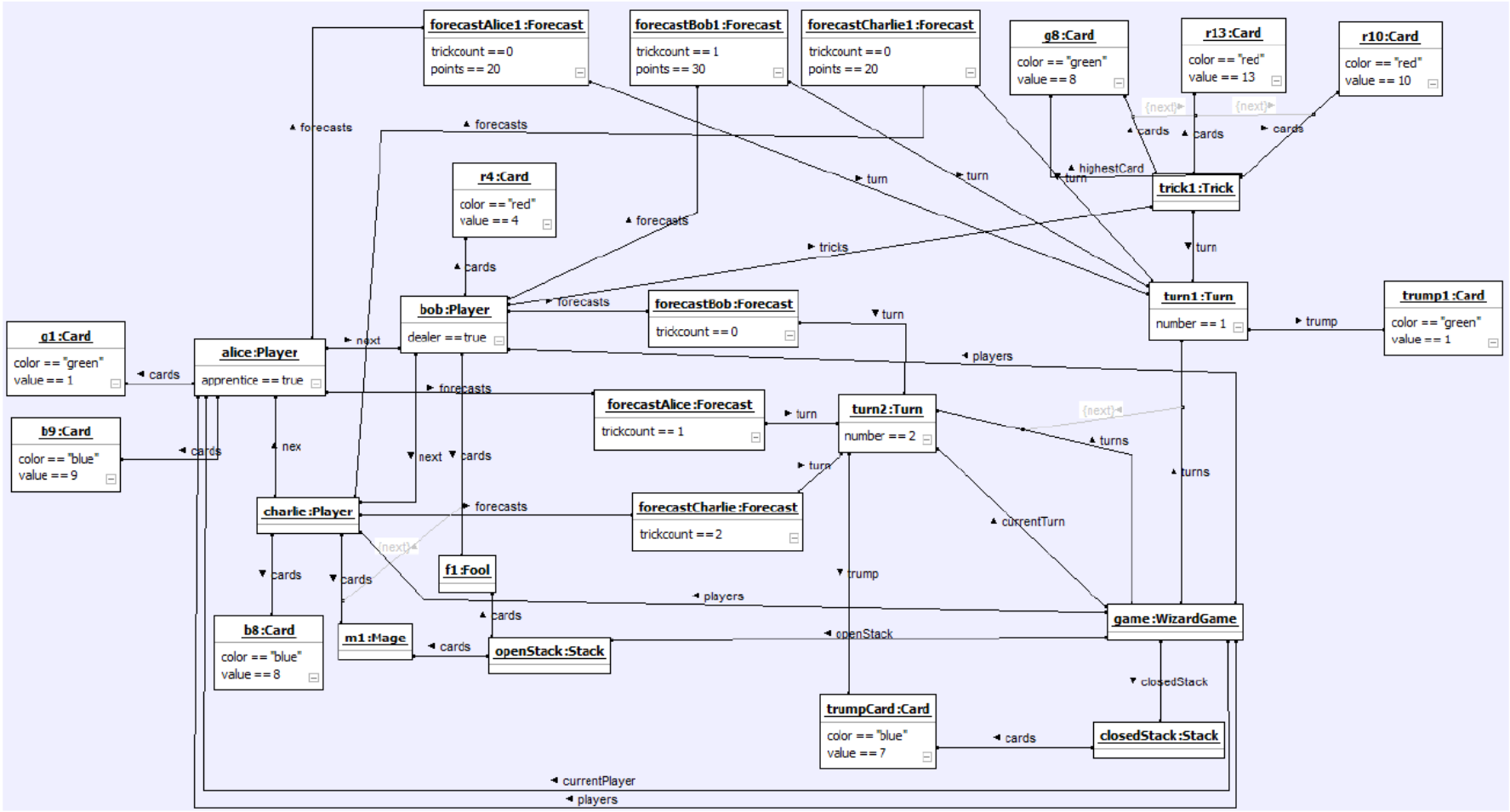


Änderungen am Wizard Klassendiagramm



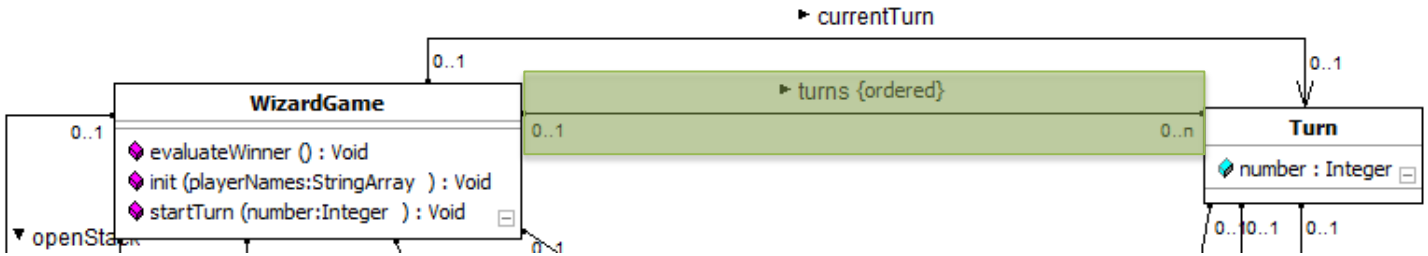
Beispiel-Objektstruktur

- **Beispiel im PM Blog**



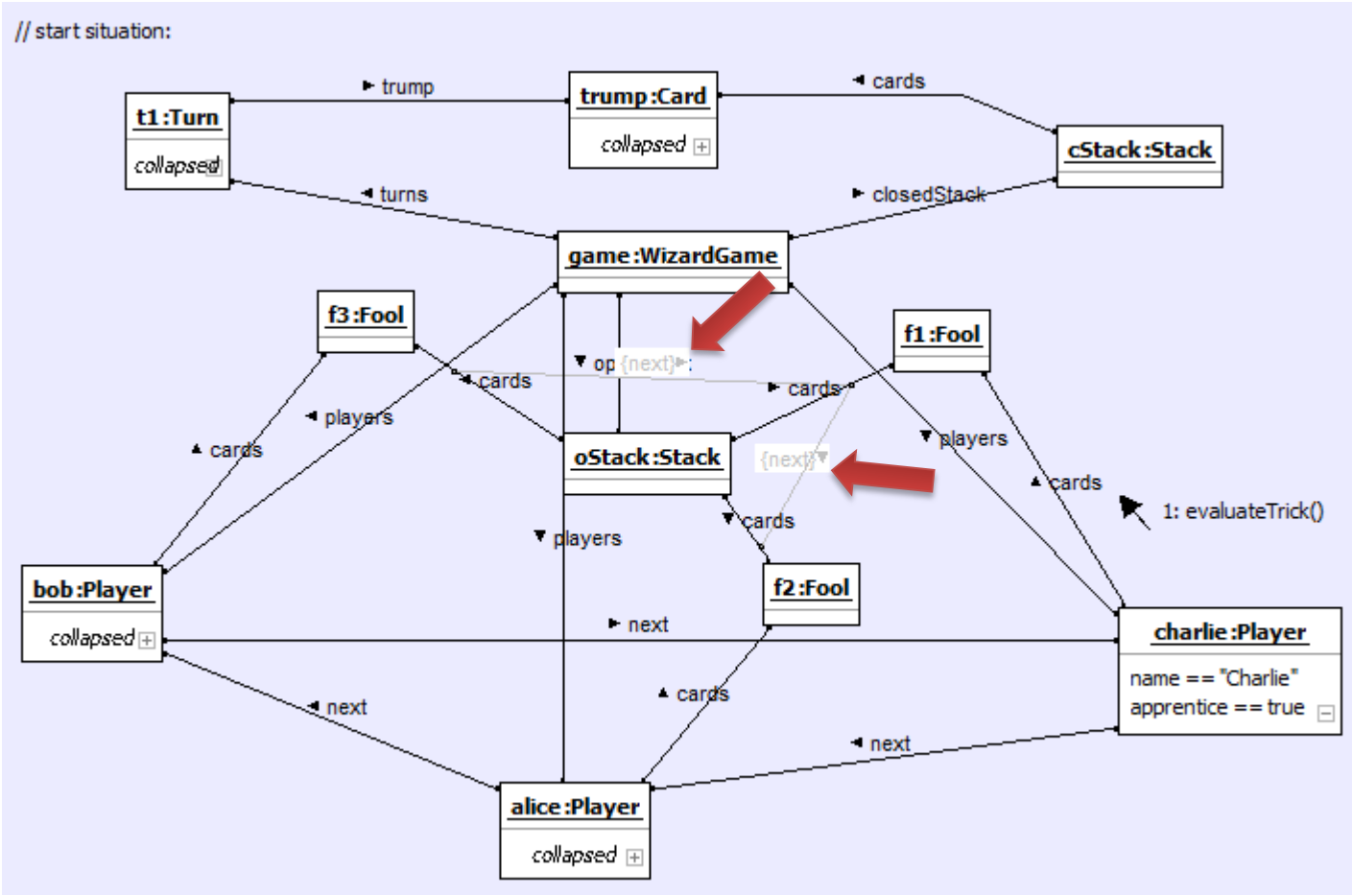
Fujaba 4 Eclipse: Storyboards I

- **Folien Übung 4:**
  - Installieren von Fujaba 4 Eclipse
  - Anlegen eines neuen Storyboards
  - Code generieren
  - JUnit Test ausführen
- **Heute „MultiLinks“**
  - Nötig bei {ordered} Assoziationen
  - Legt Reihenfolge der Elemente fest



Fujaba 4 Eclipse: Storyboards II

- **MultiLinks in Storyboards**



Vorstellung HA 4 I

- **Abgabe bis zum 16.06.2011, 23:59 Uhr**
- **Erstellen von Storyboards**
  - Runde 2 initialisieren, Normaler Trumpf
  - Runde 2 initialisieren, Zauberer Trumpf
  - Karte ausspielen: Kann Farbe bedienen, bedient Farbe
  - Karte ausspielen: Kann Farbe bedienen, bedient nicht
  - Karte ausspielen: Kann Farbe nicht bedienen, wirft andere Farbe ab
  - Stich evaluieren: Höchste Karte
  - Stich evaluieren: Höchster Trumpf
  - Stich evaluieren: 2 Narren

Vorstellung HA 4 II

- **Implementieren der Methoden**
  - `WizardGame::startTurn(turn:Integer):void`
  - `Player::playCard(card:Card):void`
  - `Player::evaluateTrick():void`

- **Bis alle Tests grün werden!**

Praktische Übung: Storyboards „Stich evaluieren“

- **Erstellt Storyboards zu** `Player::evaluateTrick():void`
  - Stich evaluieren: 3 Narren
    - Hinweis: Den Stich bekommt der erste, der einen Narren gespielt hat!
  - Stich evaluieren: 2 Zauberer, 1 andere Karte
    - Hinweis: Den Stich bekommt der erste der einen Zauberer gespielt hat!
  - Stich evaluieren: 1 Zauberer, 2 andere Karten

Ende

Schönes WE!