Die Aufgaben müssen einzeln bearbeitet und abgegeben werden. Die Abgabe muss bis **spätestens Donnerstag 19.05.2011 um 23:59 Uhr** über unser Hausaufgabenabgabesystem **http://seblog.cs.uni-kassel.de/pmss11/** erfolgen. Die Abgabe ist nur als einzelne *.zip oder *.jar-Datei möglich. Daher müssen alle für eine Abgabe relevanten Daten zu einem solchen Archiv kombiniert werden. Diese Hausaufgabe gibt **21,5 + 6** Punkte.

Hinweise zur Abgabe:

- Die Hausaufgabe als exportiertes Eclipse Projekt (*.zip, **nicht** den gesamten Workspace) abgeben. Das kann mit Hilfe der Eclipse Export Funktion durchgeführt werden. Ist das Projekt nicht korrekt exportiert, kann es bei der Korrektur nicht berücksichtigt werden (es bietet sich also an, den Import des exportierten Projektes auszuprobieren).

  WICHTIG Benennen Sie ihre Projekte für diese und alle zukünftigen Abgaben nach folgendem Schema:

  ```
  PMSS2011_HA<a>_A<b>_<Matrikelnummer>,
  ```

  wobei <a> für die aktuelle Hausaufgabe und <b> für die Aufgabennummer steht. Beispiel:

  ```
  PMSS2011_HA2_A1_12345678.
  ```

- Falls die Zusatzaufgabe bearbeitet wird, müssen die Tests dem Eclipse Projekt von Aufgabe 1 hinzugefügt werden. Denken Sie daran den einzelnen Testmethoden aussagekräftige Namen zu geben.

Aufgabe 1 - Implementierung des Klassendiagramms zu Wizard (21,5P)

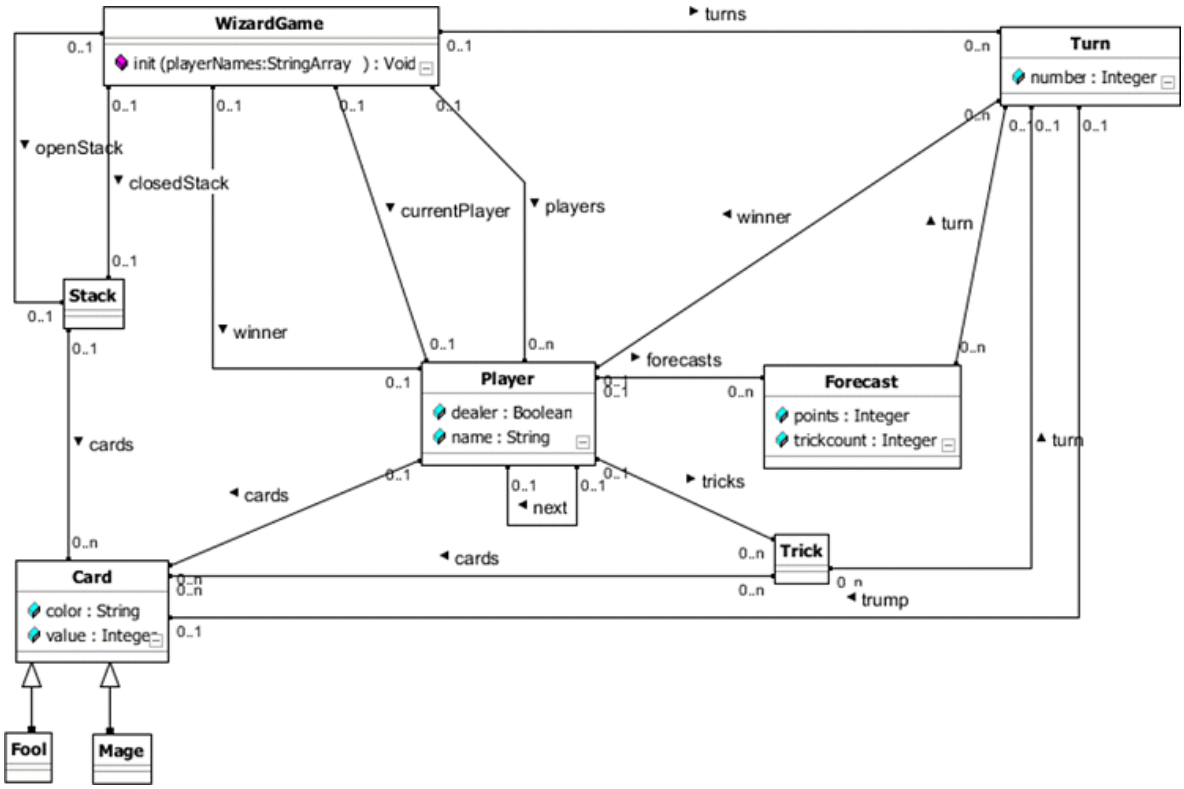

Abbildung 1: Wizard Klassendiagram

Gegeben ist das in Abbildung 1 dargestellte Klassendiagramm. Implementieren Sie das Klassendiagram in Java. Folgende Vorgehensweise wird vorgeschlagen:

1. Erstellen Sie eine 'public class' in separater .java-Datei für jede Klasse im Diagramm.
2. Fügen Sie den Klassen die entsprechenden Attribute hinzu. Achten Sie auf Verkapselung der Attribute durch getter/setter!
3. Fügen Sie den Klassen die ensprechenden Methoden hinzu - eine Implementierung der Methodenrümpfe ist nicht gefordert.
4. Implementieren Sie die Assoziationen und stellen Sie referenzielle Integrität sicher: Dazu fügen Sie den Klassen Zugriffsmethoden (add/remove/get) für zu-n-Assoziationen hinzu und wählen eine geeignete Containerklasse (z. B. `ArrayList`,`Vector`,`HashSet`,...). Sorgen Sie dafür, dass bei bidirektionalen Assoziationen die Rückrichtung automatisch mitgesetzt wird.

Zusatzaufgabe - JUnit Tests (6P)

Schreiben Sie JUnit Tests, die die referenzielle Integrität der folgenden bidirektional Assoziationen testen:

- cards (Trick - Card)
- turns (WizardGame - Turn)
- trump (Turn - Card)

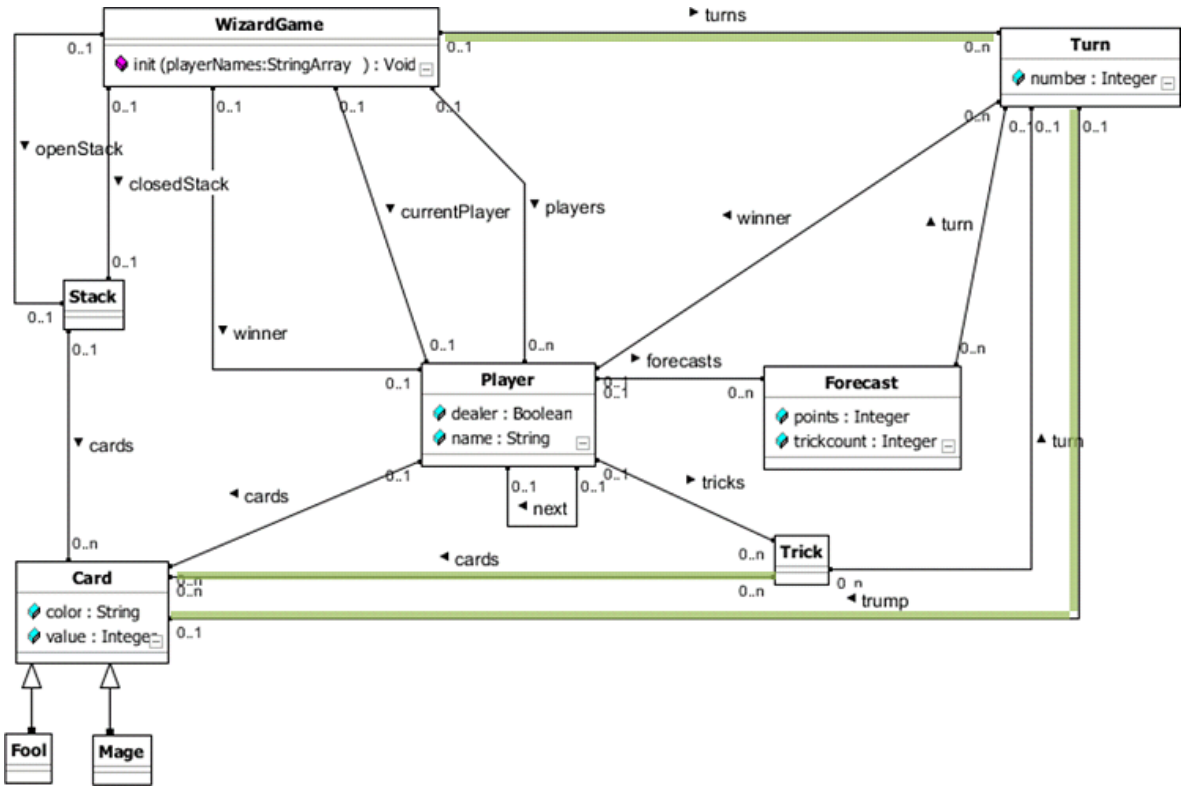

Abbildung 2: Wizard Klassendiagram mit hervorgehobenen Assoziationen

In Abbildung 2 sind diese drei Assoziationen noch einmal im Klassendiagram farblich hervorgehoben.

Es soll sowohl das Setzen/Hinzufügen als auch das Löschen von Elementen überprüft werden. Schreiben Sie Tests die **beide** Richtungen der Assoziationen testen.

Hinweis: Weil die Tests zum Löschen eines Elements vorraussetzen, dass das Element vorher hinzugefügt wurde, können diese in einem einzigen Test zusammengefasst werden. Sie benötigen also 3 (Assoziationen) $*$ 2 (Rollen) $=$ 6 Tests. Nutzen Sie zur Verifizierung ihrer Tests passende assert-Methoden.

Bsp:

- Assert.assertTrue(object.trueExpected());
- Assert.assertFalse(object.falseExpected());
- Assert.assertEquals(3, object.getSize());