Die Aufgaben müssen einzeln bearbeitet und abgegeben werden. Die Abgabe muss bis **spätestens Donnerstag 02.06.2011 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 **20 + 8** Punkte.

Hinweise zur Abgabe:

- Aufgabe 1 und die Zusatzaufgabe als exportierte Eclipse Projekte. Falls Sie mehrere Projekte anlegen, können diese alle in **eine** .zip Datei gepackt werden (erledigt die Eclipse Export Funktion automatisch!). Sind die Projekte nicht korrekt exportiert, können diese bei der Korrektur nicht berücksichtigt werden (es bietet sich also an, den Import des exportierten Projekts auszuprobieren).

  WICHTIG Bennen Sie ihre Projekte nach folgendem Schema:

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

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

  ```
  PMSS2011_HA3_A1_12345678.
  ```

Vorbereitung

Für die Bearbeitung der Hausaufgabe 3 benötigen Sie eine Implementierung des Klassendiagramms von Wizard. Sie können entweder ihre eigene Implementierung aus Hausaufgabe 2 verwenden oder eine fertige Version vom Blog zur Veranstaltung herunterladen (`http://seblog.cs.uni-kassel.de/category/currentterm/pm-ss11/`) und als Projekt in Eclipse importieren.

Darüber hinaus benötigen Sie das *eDOBS*-PlugIn für Eclipse. Installieren Sie das PlugIn wie in der Übung gezeigt über den Eclipse Update Mechanismus. Die URL für die Update-Site lautet: `http://www.se.eecs.uni-kassel.de/se/fileadmin/se/projects/eDOBS/update`.

Aufgabe 1 - Implementierung (6P)

Implementieren Sie die Methode

`init(playerNames:StringArray):void`

Sie soll das Spiel initialisieren und folgende Objekte erstellen:

- Für jeden per Parameter übergebenen Namen einen Spieler
- Einen „Open-“ und einen „Closed-Stack“
- Alle 60 Karten:
  - 13 gelbe Karten (1-13)
  - 13 blaue Karten (1-13)
  - 13 rote Karten (1-13)
  - 13 grüne Karten (1-13)
  - 4 Zauberer
  - 4 Narren

Vergessen Sie nicht alle Objekte korrekt zu verlinken.

Aufgabe 2 - eDOBS (4P)

Rufen Sie die Methode `init(...)` mit dem Parameter `new String[]{„Alice“, „Bob“, „Charlie“}` auf. Setzen Sie am Ende der Methode `init(...):void` einen Breakpoint und fertigen Sie einen Screenshot von eDOBS an, indem die aktuelle Objektstruktur nach Ausführung der
`init(...):void` Methode zu sehen ist. Im Screenshot sollte folgendes zu sehen sein:

- Drei Spieler mit den als Parameter übergebenen Namen
- Einen „Open-“ und einen „Closed-Stack“
- Alle 60 Karten

sowie die Links zwischen den Objekten. Hinweise zur Bedienung des eDOBS finden sich auf den Folien zur Übung. **WICHTIG: Gruppieren Sie die Objekte sinnvoll (z.B. alle Karten einer Farbe) um die Korrektur zu erleichtern!**

Aufgabe 3 - Zetteltest, eDOBS

Die Klasse `WizardGame` soll eine Methode `startGame():void` besitzen, welche das Spiel auf die erste Runde vorbereitet. Hierzu wird ein „Turn“-Objekt angelegt, jedem Spieler eine Karte vom „closedStack“ zugewiesen sowie eine Trumpf Karte auf den „openStack“ gelegt. Eine Beispielimplementierung ist in Listing 1 abgedruckt.

```
1  public void startGame ()
2      {
3          // Create the first turn
4          Turn firstTurn = new Turn();
5          firstTurn.setNumber(1);
6          addToTurns(firstTurn);
7          setCurrentTurn(firstTurn);
8
9          Player currentPlayer = null;
10         Card card = null;
11         // Give each player a card
12         for (Player player : getPlayers())
13         {
14             currentPlayer = player;
15             card = getClosedStack().popCard();
16             currentPlayer.addToCards(card);
17         }
18
19         // Put one card on the open stack
20         card = getClosedStack().popCard();
21         getOpenStack().addToCards(card);
22     }
```

Listing 1: Beispielimplementierung der Methode WizardGame::startGame():void

Falls Sie die im Blog zur Verfügung gestellte Implementierung verwenden, ist diese Methode bereits implementiert.

3.1 - Zetteltest (8P)

Fertigen Sie einen Zetteltest zur Ausführung der Methode `startGame():void` an. Nutzen Sie hierfür die Tabelle auf der nächsten Seite. Die Objektstruktur nach erstmaliger Ausführung bis Zeile 14 ist bereits vorgegeben. Zeichnen Sie für jeden Schritt die vollständige Objektstruktur **nach** einem Durchlauf der Schleife. Falls Sie die Implementierung in Listing 1 verwenden, ist die Schleife in den Zeilen 11-16 gemeint. Die finale Objektstruktur (also nachdem Zeile 20 ausgeführt wurde) soll in die letzte Zeile der Tabelle eingetragen werden.

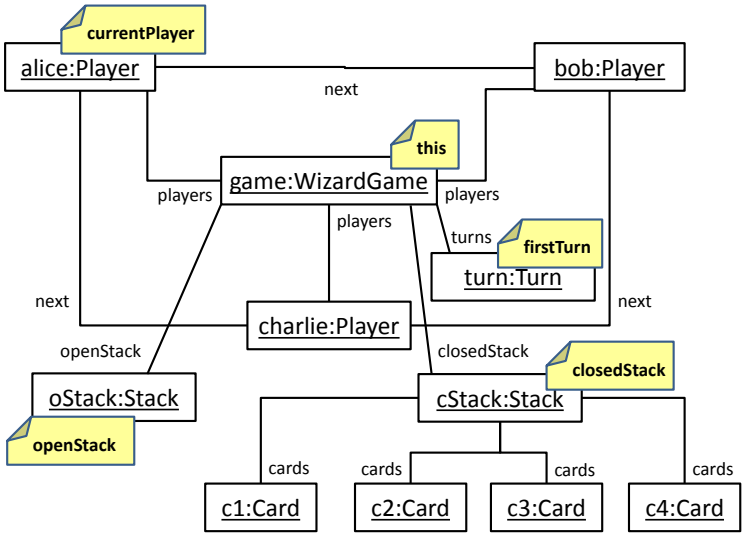
Schritt	Objektstruktur
1	
2	
3	
4	
5	

Tabelle 1: Tabelle für Zetteltest

3.2 - eDOBS (2P)

Setzen Sie einen Breakpoint an das Ende der Methode `startGame():void` und fertigen Sie einen Screenshot der eDOBS Objektstruktur an.

Zusatzaufgabe - Implementierung erweitern (8P)

Schreiben Sie einen JUnit Test, der die Korrektheit der ersten Vorhersagerunde überprüft. Die erste Vorhersagerunde sei korrekt falls:

- Jeder Spieler ein „Forecast“ Objekt besitzt, welches der aktuellen Runde zugeordnet ist.
- Das `trickcount` Attribut der Klasse `Forecast` nicht größer als das `number` Attribut des zugeordneten „Turn“ Objektes ist.

Erweitern Sie anschließend ihre Implementierung der Methode `Wizard::startGame():void` um jeden Spieler am Ende der Methode um seine Vorhersage zu bitten. Hierzu soll der Klasse `Player` eine Methode `doForecast:void` hinzugefügt werden. Die Implementierung der Methode `Player::doForecast():void` soll:

- Ein „Forecast“ Objekt erstellen und es dem Spieler zuweisen.
- Das „Forecast“ Objekt mit der aktuellen Runde verbinden (`WizardGame::currentTurn`).
- Dem „Forecast“ Objekt einen beliebigen Wert für `trickcount` zuweisen (natürlich kleiner als die momentane Runde)

Fertigen Sie einen Screenshot der eDOBS Objektstruktur an, *bevor* die Spieler zur Abgabe ihrer Vorhersagen aufgefordert werden (Startsituation) und *nach* der Abarbeitung der Methode (Endsituation) an. Erzeugen Sie in ihrem Test die in Tabelle 1 Zeile 1 dargestellte Objektstruktur und rufen Sie `Wizard::startGame():void` auf.

Die Abgabe für diese Aufgabe beinhaltet also:

1. Den JUnit Test
2. Die Erweiterung der Methode `WizardGame:startGame():void` sowie die Implementierung der Methode `Player::doForecast():void`.
3. Zwei Screenshots vom eDOBS.