Die Abgabe muss bis **spätestens Donnerstag 12.01.2012 um 23:59 Uhr** über unser Hausaufgabenabgabesystem **http://seblog.cs.uni-kassel.de/pmws1112/** 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 **23** 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:

  ```
  PMWS1112_HA<a>_A<b>_<Matrikelnummer>,
  ```

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

  ```
  PMWS1112_HA7_A1_12345678.
  ```

Vorbereitung

Zur Bearbeitung dieser Aufgabe benötigen Sie ein fertig implementiertes Hexentanz Modell sowie eine grafische Oberfläche. Sie können das zu dieser Hausaufgabe gehörende Hexentanz Projekt im PM Blog **http://seblog.cs.uni-kassel.de/category/currentterm/pmws1112/** herunterladen.

Aufgabe 1 - Model-View-Controller (23P)

In dieser Hausaufgabe soll das Hexentanz Modell mit der in Übung 7 vorgestellten GUI verbunden werden. Hierzu ist es notwendig eine Reihe von Controllern zu implementieren, die die verschiedenen Modellelemente (z. B. Field, Player, Witch,...) mit ihrer grafischen Darstellung verbinden. In Abbildung 1 ist die bereits aus Übung 8 bekannte MVC Übersicht nochmals dargestellt.

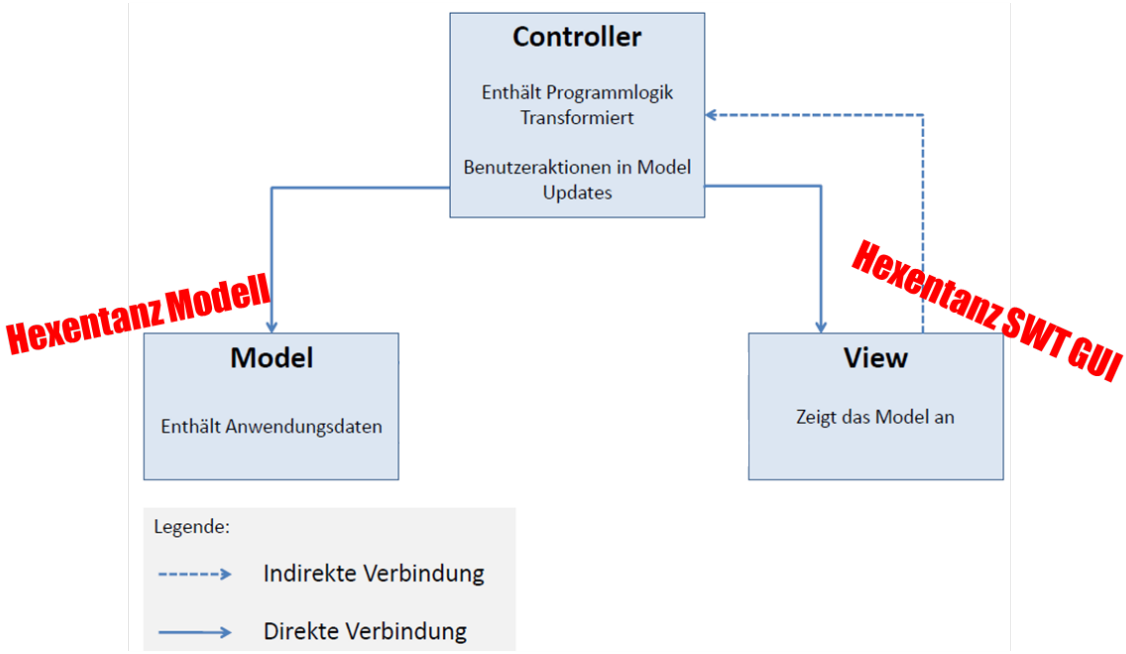

Abbildung 1: Model-View-Controller Entwurfsmuster

1.1 - LoginController (5P)

Erstellen Sie eine Klasse `LoginController`. Der `LoginController` bekommt im Konstruktor das HexentanzGame Modell Objekt sowie den in Übung 7 vorgestellte Loginoberfläche (HexentanzLoginScreenGUI) übergeben. Der Controller soll folgende Aktionen ausführen:

- Drückt der Nutzer auf „Start Game", soll zunächst geprüft werden, ob in mindestens zwei Textfelder etwas eingegeben wurde. Falls nicht, soll ein Popup erscheinen mit einer entsprechenden Meldung (Tip: Hier können statische Methoden der Klasse `MessageDialog` verwenden werden).
- Drückt der Nutzer auf „Start Game" und es wurden mindestens zwei Namen eingegeben, soll der

`GameController` instanziiert und gestartet werden. Dafür ist die init-Methode aufzurufen und für jeden Spieler die Methode chooseColor(..) - sie erzeugt Hexen und verbindet den Spieler mit seinem Startfeld. Dies soll folgendermaßen mit dem Aufruf der Methode findPlayer(..) der Klasse Util kombiniert werden:

```
// Assign the colors to players
for(String playerName : playerNames.keySet())
{
Util.findPlayer(playerName, game).chooseColor(playerNames.get(playerName));
}
```

Der `LoginController` hat damit seinen Dienst erfüllt und kann gestoppt werden, wobei zunächst die eigene stop-Methode aufgerufen wird und erst danach die start-Methode des Game-Controllers.

- Drückt der Nutzer auf „Exit Game" soll das Spiel beendet werden.

1.2 - GameController (6P)

Erstellen Sie eine Klasse `GameController`. Der `GameController` bekommt im Konstruktor das HexentanzGame Modell Objekt sowie die in Übung 7 vorgestellte Spieloberfläche (HexentanzGameScreenGUI) übergeben. Der Controller soll folgende Aktionen ausführen:

- Für jeden Spieler instantiiert der `GameController` einen `PlayerController` und startet ihn.
- Für jede Hexe instantiiert er einen `WitchController` und startet ihn.
- Für den Würfel instantiiert er einen `DiceController` und startet ihn.
- Der `GameController` soll sich bei folgenden Modellelementen als `PropertyChangeListener` registrieren und bei Änderungen die entsprechenden Anpassugen der GUI vornehmen.
  - Hexentanz.winner (um das Spielende und den Gewinner zu überwachen)
- Wird ein Event empfangen, dass der Winner Link beim HexentanzGame gesetzt wurde, soll ein Popup mit dem Namen des Gewinners eingeblendet werden. Nach Klicken des OK-Buttons soll das Spiel beendet werden.

1.3 - PlayerController (3P)

Erstellen Sie eine Klasse `PlayerController`. Der `PlayerController` bekommt im Konstruktor das Player Objekt sowie die in Übung 7 vorgestellte Spieloberfläche (HexentanzGa-

meScreenGUI) übergeben. Der Controller soll folgende Aktionen ausführen:

- Der `PlayerController` soll sich bei folgendem Modellelement als `PropertyChangeListener` registrieren und bei Änderungen die entsprechenden GUI Elemente updaten:
  - Player.instruction des übergebenen Player Objekts.
- Der `PlayerController` soll das Update für folgendes grafische Element übernehmen:
  - Die Label des übergebenen HexentanzGameScreen Objekts aktualisieren. Es zeigt an, welcher Spieler an der Reihe ist und welche Aktion er ausführen soll.

1.4 - WitchController (3P)

Erstellen Sie eine Klasse `WitchController`. Der `WitchController` bekommt im Konstruktur das Witch Objekt sowie die in Übung 7 vorgestellten Spieloberfläche (HexentanzGameScreenGUI) übergeben. Der Controller soll folgende Aktionen ausführen:

- Der `WitchController` soll sich bei folgenden Modellelementen als `PropertyChangeListener` registrieren und bei Änderungen die entsprechenden GUI Elemente updaten:
  - Witch.field des übergebenen Witch Objekts.
- Der `WitchController` soll das Update für folgende grafische Elemente übernehmen:
  - Das Hexen Bild auf dem Label des übergebenen HexentanzGameScreen Objekts aktualisieren. Es zeigt eine aufgedeckte Hexe an.
  - Wenn sich die Hexe rückwärts bewegt: Das entsprechende Hexenbild in dem Label setzen.
  - Wenn sich die Hexe nicht bzw. nicht rückwärts bewegt: Das graue Hexenbild in dem Label setzen.

1.5 - DiceController (4P)

Erstellen Sie eine Klasse `DiceController`. Der `DiceController` bekommt im Konstruktor das Dice Objekt sowie die in Übung 7 vorge-

stellte Spieloberfläche (HexentanzGameScreenGUI). Der Controller soll folgende Aktionen ausführen:

- Der `DiceController` soll sich bei folgenden Modellelementen als `PropertyChangeListener` registrieren und bei Änderungen die entsprechenden GUI Elemente updaten:
  - Dice.pips des übergebenen Dice Objekts.
- Der `DiceController` soll das Update für folgende grafische Elemente übernehmen:
  - Das der Anzahl der gewürfelten Augen entsprechende Bild in dem diceLabel des übergebenen HexentanzGameScreen Objekts setzen.
- Der `DiceController` meldet sich als `SelectionListener` beim rollButten der im Konstruktor übergebenen Spieloberfläche (HexentanzGameScreen GUI) an.
- Drückt der Nutzer auf den Button soll auf dem Dice die Methode `Dice::roll()` aufgerufen werden.

1.6 - Hexentanz starten (2P)

Erstellen Sie eine Klasse `StartHexentanz` mit einer `main`-Methode, in der Sie den Login-Controller instanziieren und starten. Das Ergebnis beim Ausführen dieser Klasse sollte ein Erscheinen der Login Oberfläche sein.