Diese Hausaufgabe darf **nicht** in einer Gruppe bearbeitet werden. Abgabe bis **spätestens Freitag 11.06.2010, 23.59 Uhr** per Mail mit dem Betreff **PMSS2010 HA6 <Matrikelnummer>** (z. B. „PMSS2010 HA5 12345678“) an **pm@cs.uni-kassel.de**. Für diese Hausaufgabe gibt es 24 Punkte.

Hinweis zur Abgabe: Die Abgabe **MUSS** als exportiertes Eclipse Projekt erfolgen. 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:

```
PMSS2010 HA5 A<i> <Matrikelnummer>,
```

wobei <i> für die Aufgabennummer steht. Beispiel:

```
PMSS2010 HA5 A1 12345678.
```

Vorbereitung

Für die Bearbeitung der Hausaufgabe 6 benötigen Sie eine Implementierung des Klassendiagramms von Mancala. Sie können entweder ihre eigene Implementierung aus Hausaufgabe 5 verwenden oder eine fertige Version vom Blog zur Veranstaltung herunterladen (`http://seblog.cs.uni-kassel.de/category/currentterm/pm/`) und als Projekt in Eclipse importieren.

Darüber hinaus benötigen Sie ein aktuelles *Fujaba4Eclipse*-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/fileadmin/se/update`.

Storyboards (24P)

Erstellen Sie mit Fujaba4Eclipse Storyboards zu folgenden Szenarien. Für jedes Szenario werden 3 Punkte vergeben. Passen Sie die Methoden `Mancala::checkEnd()` und `Pit::moveStones()` soweit an, bis alle von den Storyboards generierten JUnit Tests erfolgreich durchlaufen. Die Endsituation und dementsprechend die Implementierung der beiden Methoden muss die offizielen PMSS2010 Mancala Regeln (zu finden im PMSS2010 Blog!) beachten.

Abzugeben ist das exportierte Eclipse Projekt, welches ein Fujaba Projekt mit den Storyboards sowie die korrekte und vollständige Implementierung der beiden obigen Methoden enthält.

- *Steine bewegen, keine Umrundung*

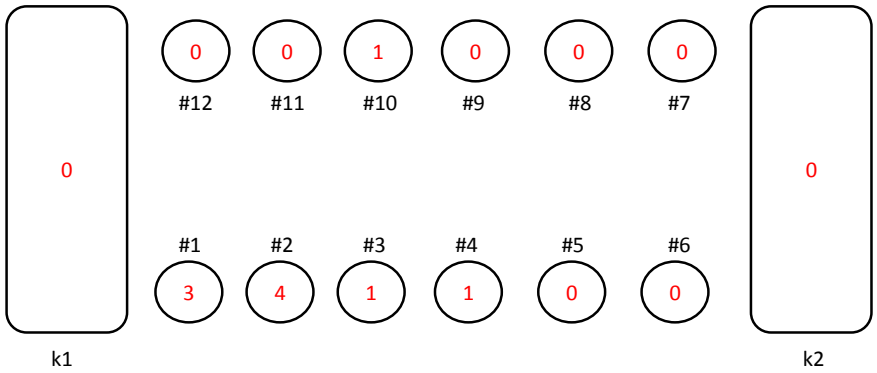

Abbildung 1: Steine bewegen, keine Umrundung

Spieler mit den *unteren* Pits ist an der Reihe. Aufruf von `Pit::moveStones()` auf Pit #1.

- *Steine bewegen, mit Umrundung, Gegenüber leer*

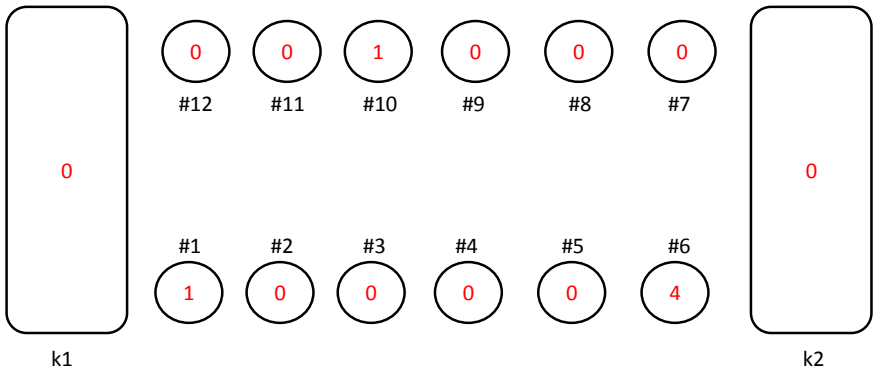

Abbildung 2: Steine bewegen, mit Umrundung, Gegenüber leer

Spieler mit den *unteren* Pits ist an der Reihe. Aufruf von `Pit::moveStones()` auf Pit #6.

- *Steine bewegen, mit Umrundung, Gegenüber besetzt*

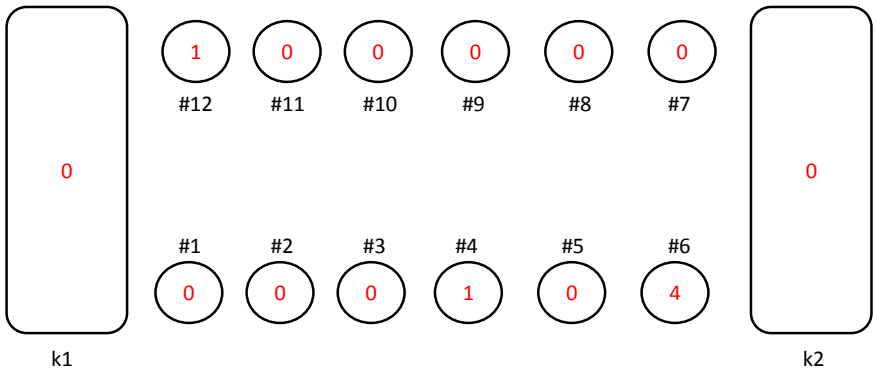

Abbildung 3: Steine bewegen, mit Umrundung, Gegenüber besetzt

Spieler mit den *unteren* Pits ist an der Reihe. Aufruf von `Pit::moveStones()` auf Pit #6.

- *Steine bewegen, Klauen möglich*

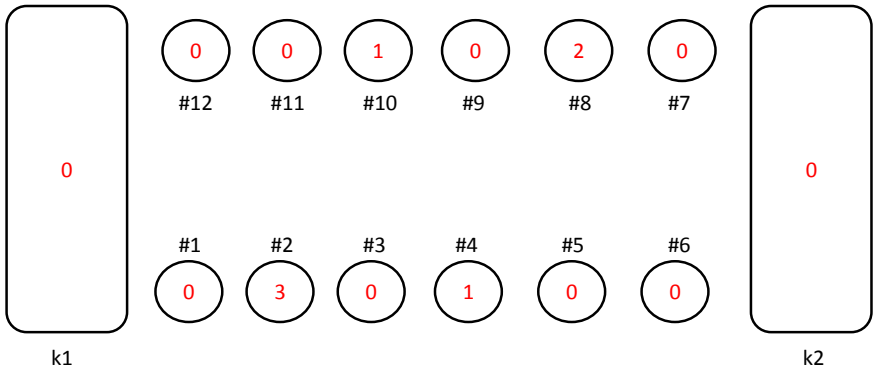

Abbildung 4: Steine bewegen, Klauen möglich

Spieler mit den *unteren* Pits ist an der Reihe. Aufruf von `Pit::moveStones()` auf Pit #2.

- *Steine bewegen, Klauen nicht möglich*

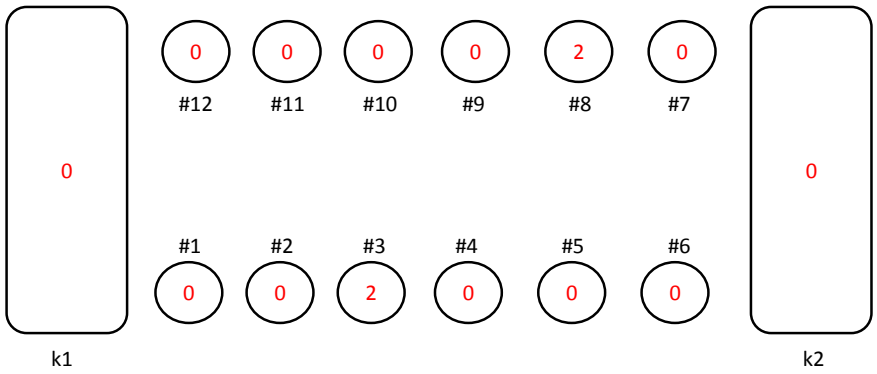

Abbildung 5: Steine bewegen, Klauen nicht möglich

Spieler mit den *unteren* Pits ist an der Reihe. Aufruf von `Pit::moveStones()` auf Pit #3.

- *Steine bewegen, Spieler nicht dran*

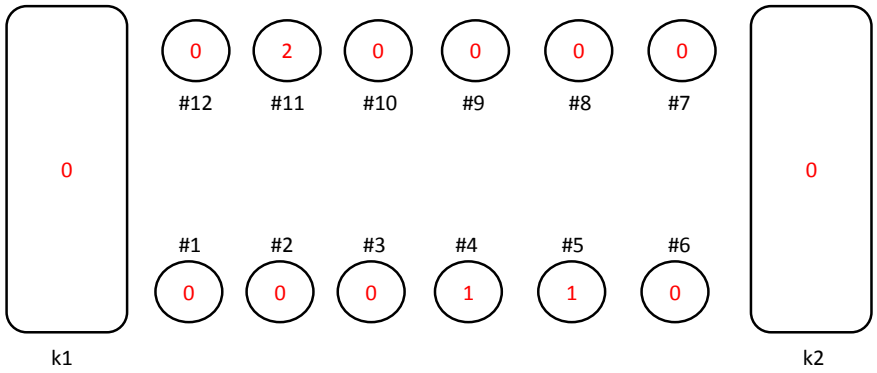

Abbildung 6: Steine bewegen, Spieler nicht dran

Spieler mit den *oberen* Pits ist an der Reihe. Aufruf von `Pit::moveStones()` auf Pit #4.

- *Gewinner ermitteln (positiv)*

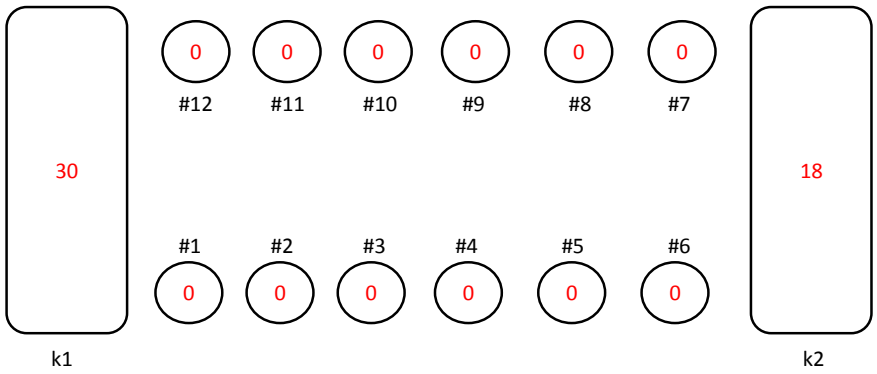

Abbildung 7: Gewinner ermitteln (positiv)

Aufruf von `Mancala::checkEnd()`. (Hinweis: Es muss einen „winner-Link“ geben.)

- *Gewinner ermitteln (negativ)*

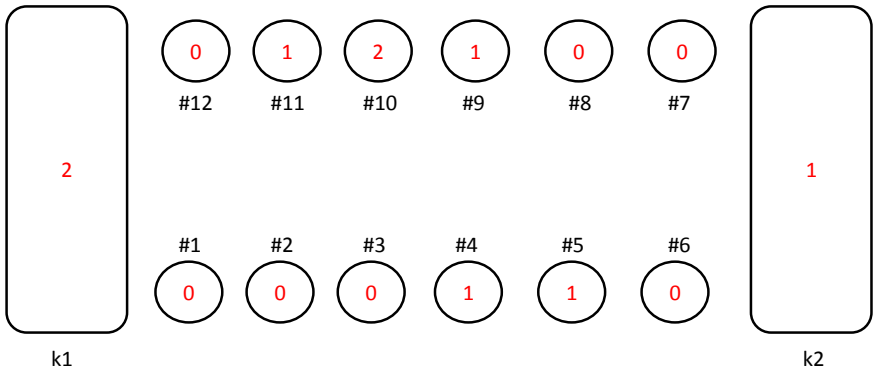

Abbildung 8: Gewinner ermitteln (negativ)

Aufruf von `Mancala::checkEnd()`. (Hinweis: Es darf keinen „winner-Link“ geben.)