Die Aufgaben können (müssen aber nicht) in Gruppen von bis zu drei Leuten bearbeitet werden. Falls die Aufgaben in einer Gruppe bearbeitet werden, genügt **eine** Abgabe mit den Namen aller Gruppenmitglieder. Abgabe bis **spätestens Mittwoch 12.05.2010** per Mail mit dem Betreff **PMSS2010 HA4 <Matrikelnummer>** (z. B. „PMSS2010 HA4 12345678“) an **pm@cs.uni-kassel.de**. Für diese Hausaufgabe gibt es 16 + 2 Punkte.

Hinweise zur Abgabe:

- Aufgabe 1 und 2 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:

  ```
  PMSS2010 HA4 A<i> <Matrikelnummer>,
  ```

  wobei <i> für die Aufgabennummer steht. Beispiel:

  ```
  PMSS2010 HA4 A1 12345678.
  ```

- Falls die Zusatzaufgabe bearbeitet wird, diese als PDF abgeben. Andere Formate führen zu Punktabzug!

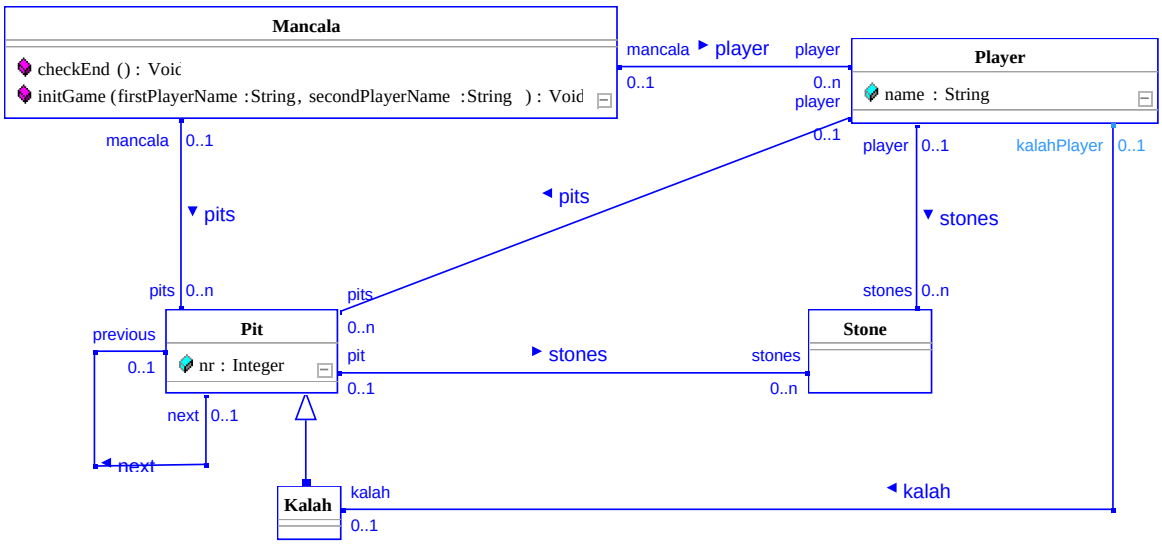

Abbildung 1: Mancala Klassendiagramm

Aufgabe 1 - Implementierung eines Klassendiagramms (10P)

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: Fügen Sie den Klassen Zugriffsmethoden (add/remove/get) für zu-n-Assoziationen hinzu und wählen Sie eine geeignete Containerklasse (z. B. `ArrayList`,`Vector`, `HashSet`,...). Sorgen Sie dafür, dass bei bidirektionalen Assoziationen die Rückrichtung automatisch mitgesetzt wird.

Aufgabe 2 - JUnit Tests (6P)

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

- pits (Mancala - Pit)
- kalah (Player - Kalah)
- next (Pit - Pit)

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

Hinweis: Da 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.

Zusatzaufgabe (2P)

Gegeben sei das in Abbildung 2 Objektdiagramm:

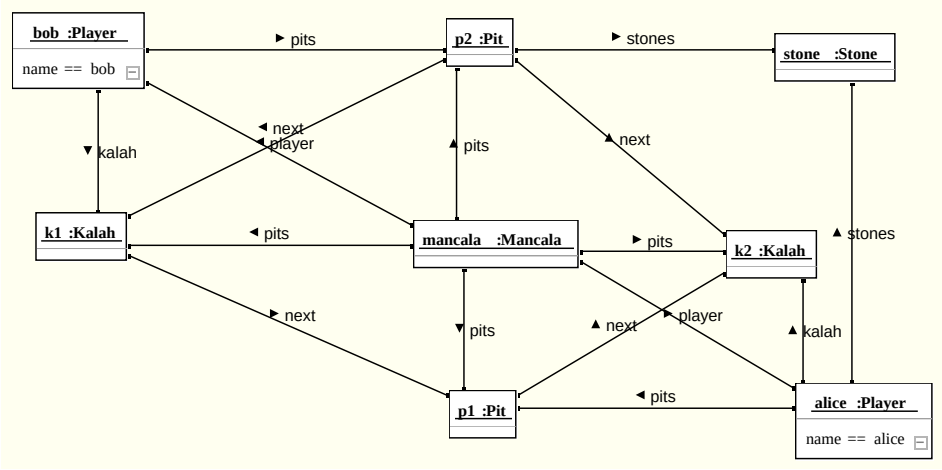

Abbildung 2: Mancala Objektdiagramm

Nun wird folgende Anweisung durchgeführt: `k1.getStones().add(stone)` Erklären Sie, wie sich die Objektstruktur nach dieser Anweisung ändert.

Hinweis: Das Objektdiagramm bezieht sich auf das in Abbildung 1 dargestellte Klassendiagramm. Die zugehörige Implementierung stellt referentielle Integrität sicher.