Richtlinien und Hinweise für die MW-Übungsaufgaben

- Jeder Student bekommt ein Projektverzeichnis unter /proj/i4mw/<*loginname*>.
- In diesem Projektverzeichnis soll für jede Übungsaufgabe ein SVN-Repository angelegt werden. Das Unterverzeichnis dafür soll aufgabe*X* heißen (*X* ist durch die Aufgabennummer zu ersetzen). Für die fünfte Aufgabe z.B. mit:
  `svnadmin create --fs-type fsfs /proj/i4mw/<loginname>/aufgabe5`
- Zum Abgabezeitpunkt sollen alle notwendigen Quelldateien in das entsprechende Subversion-Repository eingecheckt werden.
- Alle Gruppenmitglieder sollen gemeinsam ein SVN-Repository verwenden.
- Die Lösungen müssen eigenständig erstellt worden sein. Verstöße werden geahndet.
- Die *Java Coding Guidelines* sollten befolgt werden:
  - Klassennamen beginnen mit einem *Großbuchstaben*.
  - Methodennamen beginnen mit einem *Kleinbuchstaben*.
  - Variablennamen beginnen mit einem *Kleinbuchstaben*.
  - Konstante (final static) Variablen bestehen nur aus *Großbuchstaben*.
  - Variablenamen sind ausdrucksfähig und beschreiben deren Zweck.
- Wenn eine bestimmte Programmstruktur in der Aufgabenstellung verlangt wird, muss die Lösung diese einhalten. Die Lösung soll
  - Namen von Klassen und Packages
  - Sichtbarkeit von Variablen und Methoden
  - Namen, Parametertypen und Rückgabewerte von Methoden

  wie in der Aufgabenstellung beschrieben verwenden.
  Abweichungen hiervon sind zu begründen.
- Es wird negativ bewertet, falls
  - verlangte Funktionalität nicht implementiert wird
  - nicht verlangte Funktionalität implementiert wird
  - das Programm nicht kompiliert.
- Der Code muss lesbar und nachvollziehbar sein. Falls komplizierte Teile vorkommen, sollten diese mit Kommentaren erläutert werden.
- Bei Problemen mit der Aufgabenstellung könnt ihr euch jederzeit an uns wenden: *{felser, rrkapitz, gernoth}@informatik.uni-erlangen.de*

Übungsaufgabe #1: MWLibrary - Erste Schritte

24.10.2006.

In dieser Aufgabe wird das Grundgerüst für eine Bibliotheksverwaltung erstellt, das in den folgenden Aufgaben ausgebaut werden soll. Es sollen Klassen für Medienobjekte (Buch, CD, Zeitschrift) erstellt, sowie eine einfache Datenbank implementiert werden, welche diese Objekte verwaltet. Außerdem soll eine Benutzerschnittstelle erstellt werden um mit der Bibliotheksverwaltung zu interagieren.

a) Erstelle ein Interface `Item`, das die gemeinsamen Methoden aller Medienobjekte definiert. Im Hinblick auf spätere Aufgaben sind mindestens folgende Methoden vorzusehen:

   `String getTitle()` und `void setTitle(String title)`
   Zum Setzen und Abfragen des Titels.
   (Der Titel steht stellvertretend für die Informationen über das Medium)

   `void borrow()` und `int giveback()`
   Hiermit wird das Objekt als ausgeliehen bzw. zurückgeben markiert.
   Falls es bereits verliehen ist, soll eine `AlreadyBorrowedException` geworfen werden.
   Außerdem soll ein Ausleihzähler verwaltet werden, der bei jedem Ausleihen erhöht wird.
   `giveback` liefert den aktuellen Ausleihzählerwert zurück.

   `String toString()`
   Die über ein Medium gespeicherten Informationen (Titel, Ausleihstatus, Ausleihzähler, ...) sollen ausgegeben werden.

b) Erstelle drei Klassen `Book`, `Journal` und `CD`, die alle das Interface `Item` implementieren.

c) Schreibe die Datenbank-Klasse `SimpleDBImpl` zur Verwaltung der `Item`-Objekte. Dazu ist ein Interface `SimpleDB` mit mindestens folgenden Methoden zu implementieren:

   `void register(Object key, DBObject data)`
   Fügt ein neues Objekt `data` unter dem eindeutigen Schlüssel `key` der Datenbank hinzu.
   Als Schlüssel kann ein beliebiges Objekt dienen, Daten sollen das Interface `DBObject` implementieren. `DBObject` soll dabei ein leeres "Marker-Interface" sein.

   `Object[] getAllKeys()`
   Gibt die Schlüssel aller gespeicherten Objekte als Array zurück

   `DBObject get(Object key)`
   Liefert das Objekts mit dem entsprechenden Schlüssel zum lesenden Zugriff zurück.

   `DBObject lockAndGet(Object key)`
   Wie `get()`, jedoch für schreibenden Zugriff. Vor einem erneuten Aufruf von `lockAndGet()` muss das Objekt mit `unlockAndPut()` wieder freigegeben werden,

   `void unlockAndPut(Object key, DBObject data)`
   Gibt ein mit `lockAndGet`() gesperrtes Objekt wieder frei.

   Eine mögliche Implementierung könnte darin bestehen die Objekte in zwei verschiedenen Hashtabellen zu speichern, eine für gesperrte und eine für zugreifbare Objekte.

   Im Fehlerfall sollen geeignete Exceptions generiert werden (z.B. `AlreadyExistsException`, `NotFoundException` und `AlreadyLockedException, NotLockedException`).

d) Schreibe eine Klasse `LibraryFrontend`, die als Schnittstelle zwischen dem Benutzer und der Datenbank dient und mindestens folgende Methoden aufweist:

`void registerItem(String classname, String title)`
: Fügt ein neues Objekt der Datenbank hinzu. Durch `classname` wird der Typ des Objekts angegeben (`Book`, `CD` oder `Journal`). (Klasse dynamisch laden!)

`void list()`
: Zeigt eine Liste aller Medien an.

`void borrowItem(String title)`
: Leiht das Objekt aus und erhöht den Ausleihzähler erhöhen.

`int returnItem(String title)`
: Das Objekt wird wieder zurückgegeben. Der Rückgabewert entspricht dem Ausleihzähler.

Fehler sind durch geeignete Exceptions auszudrücken (z.B. `AlreadyExistsException`, `NotFoundException`, `AlreadyBorrowedException` und `NotBorrowedException`).

Die Kommunikation mit dem Benutzer soll über eine Eingabemaske ähnlich einer Shell realisiert werden. Zum Registrieren eines neuen Buches könnte z.B. folgender Aufruf dienen:

```
> register Book "Operating Systems"
```

Bearbeitung: bis zum 02.11.2006/20:00 Uhr
Alle notwendigen Quelldateien müssen im SVN-Repository eingecheckt sein.
Die Bearbeitung ist in 2er Gruppen möglich.