Hausaufgabe 3 - Deadline (8te Woche)

Ziel: Erweiterung und Verbesserung der bisherigen Anwendung

In dieser dritten Hausaufgabe wird die bestehende Java-Konsolenanwendung um erweiterte Funktionen ergänzt. Die Aufgaben umfassen das Hinzufügen neuer Methoden, die das Sortieren und Filtern von Daten unterstützen, sowie die Implementierung eines FileRepository, um Daten aus Textdateien zu lesen und zu schreiben. Zusätzlich soll eine Methode entwickelt werden, die mindestens drei Entitäten verwendet und in der Anwendungslogik sinnvoll ist.

Aufgaben und Anforderungen:

1. Implementierung des FileRepository

Das FileRepository wird erstellt, um Daten in **`Textdateien`** zu speichern und zu laden, wobei Daten durch **`Kommas`** getrennt werden.

- Das FileRepository muss das **`IRepository<T>`**-Interface implementieren und folgende Methoden bereitstellen:
  - `create(T obj)`: Fügt eine neue Entität zur Datei hinzu.
  - `read(int id)`: Liest eine Entität anhand ihrer ID aus der Datei.
  - `update(T obj)`: Aktualisiert eine Entität in der Datei.
  - `delete(int id)`: Entfernt eine Entität aus der Datei.
- Die Struktur von `FileRepository` soll der von `InMemoryRepository` ähneln, jedoch mit Dateispeicher anstelle von `In-Memory-Speicherung`.

`Hinweis:` Stellen Sie sicher, dass die Datei nach jedem Schreibvorgang geschlossen wird und dass Fehler beim Zugriff auf die Datei ordnungsgemäß behandelt werden.

2. Neue Methoden zur Anwendung von Sortierungen

Die Anwendung muss **mindestens zwei Methoden** enthalten, die `Sortieroperationen` ausführen und für die Benutzer von Nutzen sind.

`Beispiele für sinnvolle Sortiermethoden:`

- `sortCoursesByAvailableSlots()`:
  - Sortiert alle Kurse nach der Anzahl der freien Plätze in aufsteigender Reihenfolge.

- Wird im `UniversityService` implementiert und gibt eine Liste der Kurse zurück, die für eine bestimmte Studentenregistrierung am sinnvollsten sind.

- `sortStudentsByCredits()`:
  - Sortiert die Studierenden nach der Anzahl ihrer eingeschriebenen Credits in absteigender Reihenfolge.
  - Diese Methode hilft dem Benutzer, Studierende mit hoher Studienbelastung zu identifizieren, z. B. für Beratungsgespräche oder Überprüfung der Kurswahl.

3. Neue Methoden zur Anwendung von Filtern

Die Anwendung muss **mindestens zwei Methoden** enthalten, die `Filteroperationen` ausführen und für die Benutzer sinnvoll sind.

`Beispiele für sinnvolle Filtermethoden:`

- `filterCoursesByCredits(int minCredits)`:
  - Filtert die Kurse, die mindestens die angegebene Anzahl von Credits haben.
  - Hilfreich für Studierende, die nach Kursen suchen, die ihre Anforderungen für eine Mindestanzahl von Credits erfüllen.
- `filterStudentsByEnrolledCoursesCount(int minCourses)`:
  - Filtert Studierende, die in einer Mindestanzahl von Kursen eingeschrieben sind.
  - Praktisch für Dozenten oder Administratoren, die Studierende identifizieren möchten, die eine bestimmte Kursbelastung haben.

4. Methode mit mindestens drei Entitäten

Eine zusätzliche Methode wird erstellt, die **`mindestens drei verschiedene Entitäten`** nutzt und sinnvoll in die Anwendungslogik integriert wird.

Beispiel für eine sinnvolle Methode mit mehreren Entitäten:

- `getTeachersByTotalEnrollment()`:
  - Diese Methode ermittelt die Anzahl der eingeschriebenen Studierenden für jeden Lehrer in einem Kurskatalog und gibt eine Liste der Lehrer in absteigender Reihenfolge ihrer Gesamtzahl an eingeschriebenen Studierenden zurück. Da Lehrer für mehrere

Kurse verantwortlich sein können, berechnet die Methode die gesamte Anzahl der Studierenden über alle Kurse eines Lehrers hinweg.

- Entitäten beteiligt:
    1. **Teacher** - Die Lehrer, für die die Gesamtzahl der eingeschriebenen Studierenden über alle Kurse hinweg ermittelt wird.
    2. **Course** - Die Kurse, in denen die Studierenden eingeschrieben sind und die von den Lehrern unterrichtet werden.
    3. **Student** - Die Studierenden, die in den Kursen eingeschrieben sind und die zur Gesamtanzahl der Studierenden für jeden Lehrer beitragen.
- Diese Methode ist nützlich für Administratoren und Studienberater, die eine Übersicht darüber haben möchten, welcher Lehrer für die meisten Studierenden verantwortlich ist.

5. Dokumentation und Namenskonventionen

- Alle neuen Methoden müssen mit Javadocs dokumentiert werden.
- Die Java-Benennungskonventionen müssen auf alle hinzugefügten Methoden und Variablen angewendet werden, um den Konsistenzanforderungen zu entsprechen.