Hausaufgabe 4 - Deadline (11te Woche)

Ziel: Erweiterung und Finalisierung der bestehenden Java-Konsolenanwendung durch Implementierung einer Datenbankverbindung, Behandlung von Fehlern mit benutzerdefinierten Ausnahmen (Custom Exceptions), sowie durch Erstellung von Tests zur Sicherstellung der Funktionsfähigkeit.

Aufgaben und Anforderungen:

1. Implementierung des DBRepository

Erstellen Sie eine neue Repository-Implementierung, die die Speicherung und Verwaltung von Daten in einer relationalen SQL-Datenbank ermöglicht.

- Die Klasse **`DBRepository<T>`** muss das Interface **`IRepository<T>`** implementieren.
- Es müssen alle CRUD-Operationen (`create`, `read`, `update`, `delete`) unterstützt werden.
- Die Datenbankstruktur (Tabellen und Felder) sollte die bereits implementierten Entitäten widerspiegeln (z. B. `Students`, `Courses`, `Teachers`).

Hinweis: Studenten können jede relationale SQL-Datenbank (z. B. `MySQL`, `PostgreSQL`, `SQLite`) und jede Methode für die Datenbankanbindung (z. B. `JDBC`) verwenden. Wichtig ist, dass die Implementierung die Schichtenarchitektur der Anwendung beibehält und funktional ist.

2. Einführung von Custom Exceptions

Implementieren Sie benutzerdefinierte Ausnahmen, um verschiedene Fehlerarten in der Anwendung zu behandeln.

- Erforderliche `Exception-Klassen`:
  - **`ValidationException`**: Wird ausgelöst bei ungültigen Eingaben oder Basisvalidierungsfehlern (z. B. leere Felder, negatives ID-Format).
  - `EntityNotFoundException`: Wird ausgelöst, wenn eine Entität anhand der ID nicht gefunden wird.
  - `DatabaseException`: Behandelt Fehler, die während der Datenbankoperationen auftreten (z. B. Verbindungsprobleme).
  - `BusinessLogicException`: Für Geschäftslogik-Fehler (z. B. Überschreiten der maximalen Kursplätze).

- Verwendung der Exceptions in den Schichten:
  - Presentation Layer: Überprüfung grundlegender Eingaben. Fehlerhafte Eingaben lösen ValidationException aus, die dem Benutzer als Fehlermeldung angezeigt werden.
  - Controller Layer: Führt Validierungen durch und reicht Fehler als Exceptions an den Service Layer weiter.
  - Service Layer: Verwendet BusinessLogicException für Verstöße gegen Geschäftsregeln und verarbeitet Ausnahmen aus dem Repository Layer.
  - Repository Layer: Wandelt datenbankspezifische Fehler in DatabaseException um.

3. Implementierung von Tests

Um die Funktionsfähigkeit der Anwendung sicherzustellen, implementieren Sie Tests mit einem Test-Framework wie JUnit.

- Alle Tests in einer einzigen Klasse: Erstellen Sie eine Testklasse, z. B. `ApplicationTest`, die alle Tests für die gesamte Anwendung enthält.
- Deklaration der benötigten Entitäten: Alle notwendigen Testdaten (z. B. Studenten, Kurse, Lehrer) werden in der Testklasse initialisiert.
- Tests für CRUD-Operationen: Überprüfen Sie alle CRUD-Methoden (create, read, update, delete) in einem einzigen Test.
  - Beispiel: Ein Student wird erstellt, gelesen, aktualisiert und gelöscht.
- Tests für komplexe Methoden: Jede Methode, die die Geschäftslogik betrifft, wird durch mindestens zwei Tests überprüft:
  - Ein Test für den normalen Ablauf (erfolgreicher Vorgang).
  - Ein Test, der sicherstellt, dass die korrekte Exception ausgelöst wird.