Praktikum Programmierung von Komponentenarchitekturen (I267)
Prof. Dr. Mario Neugebauer
Dipl.-Medieninf. Alexander Wülfing

Aufgabe 06 – Vereinsverwaltung – Komplexaufgabe SpringJPA

Mit der folgenden Komplexaufgabe möchten wir eine aktive und kontinuierliche Auseinandersetzung mit prüfungsrelevanten Themen und ein eigenverantwortliches Studium stärken. Die Aufgabe fließt nicht in Ihre Prüfungsnote ein. Wir sehen uns diese jedoch an und möchten Ihnen gern individuelle Hinweise geben.

In den bisherigen Praktika haben Sie sich am Beispiel "Vereinsverwaltung" schon mit

- Vererbung
- Interfaces
- JDBC
- SpringJPA

beschäftigt und ein kleines Datenmodell (Person, Verein, …) aufgebaut.

Setzen Sie das gewonnene Wissen nun in einem **eigenen kleinen SpringJPA Beispielprojekt** um, welches das **Szenario „Vereinsverwaltung“** weiter sinnvoll durch entsprechende (Entitäts)Klassen ergänzt.

Sie können sich dazu gern mit anderen Studierenden in einer kleinen Gruppe (bis 5 Personen) online zusammenfinden und das Projekt gemeinsam bearbeiten.

Bei der Umsetzung eigener Ideen sind Sie völlig frei.

Es soll lediglich:

- die Spring Data JPA verwendet werden und
- ihr Projekt und der entsprechende Lösungsansatz kurz dokumentiert werden
  - wer hat am Projekt mitgewirkt (Projektteilnehmer)
  - Beschreibung der umgesetzten Idee(n)
  - welche Probleme traten bei der Implementierung auf und wie konnten diese gelöst werden?
  - …

Laden Sie Ihr Projekt (als .zip) und die zugehörige Beschreibung nach Fertigstellung im OPAL (Upload-Beispielprojekt) hoch

- bei Gruppenarbeit reicht es aus, wenn das Projekt und die zugehörige Dokumentation nur einmal hochgeladen wird

Ein paar (hoffentlich sinnvolle) Anregungen für eine mögliche Umsetzung/Erweiterung finden Sie auf den den folgenden Seiten.

Mögliche Erweiterungen

- Integration/Verwaltung von Vereinsmitgliedern auf Basis der Klasse Mitglied (vgl. Praktikum zur Vererbung) durch eine entsprechende @Entity
  - Mitglieder hinzufügen, löschen, …
  - Aufbau eines Repositorys
    - Definition von Abfragen
  - …

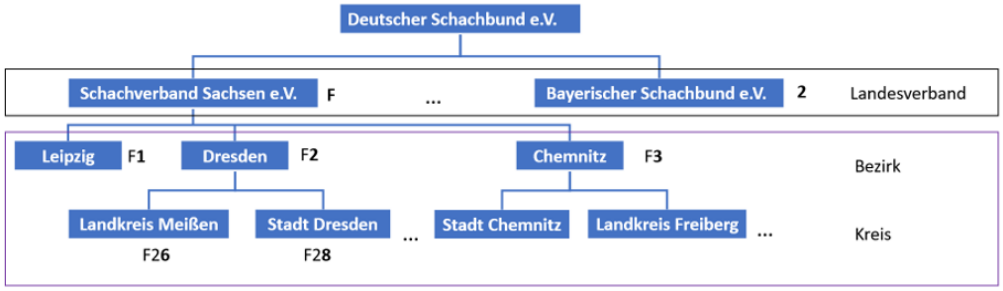
- Intialisieren der Datenbank, z.B. durch das Laden vordefinierter Daten (Personen, Mitglieder, …) unter Nutzung entsprechender (Konfigurations)Dateien
  - https://www.baeldung.com/spring-boot-data-sql-and-schema-sql
  - https://www.baeldung.com/spring-boot-h2-database
  - https://walkingtechie.blogspot.com/2018/12/execute-schema-and-data-sql-on-startup-spring-boot.html

- Persistieren der (H2 in memory) Datenbank
  - https://howtodoinjava.com/spring-boot2/h2-database-example/

- Durchführung regelmäßiger Vorgänge mit dem Spring Scheduler. Finden und implementieren Sie ein sinnvolles Szenario im Kontext der Vereins- bzw. Mitgliederverwaltung
  - https://www.baeldung.com/spring-scheduled-tasks
  - https://spring.io/guides/gs/scheduling-tasks/

- Verwaltung von Schachvereinen
  - Vereine hinzufügen, löschen, …
  - Aufbau eines Repositorys
    - Definition von Abfragen
  - Prüfen der Gültigkeit der ZPS Nummer des Vereins[1]
  - Der hierarchische Aufbau der ZPS (siehe Abbildung) in Bezirk und Kreis könnte beispielsweise genutzt werden, um alle Vereine im Landkreis Meißen zu ermitteln

[1] die ZPS spiegelt die hierarchische Organisation in Landes- und Unterverbände wieder (vgl. auch JDBC Aufgabe)



- …

- Integration/Umsetzung des Beobachter-Musters (Observer-Pattern) im Demo-Projekt an einem sinnvollen Beispiel (z.B. Aktualisierung der DWZ[2] oder Elo-Zahl[3] eines Spielers)
  - http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_10_002.htm#mjc7ffdeb540e30917defcbcab81cc06b4
  - https://www.philipphauer.de/study/se/design-pattern/observer.php#variationen

- Verwaltung von (Schach)Turnieren

Hinweis: Die Tabellenstruktur der Access-Datenbank „Turnierverwaltung“ aus den ersten Aufgaben könnte hier ebenfalls ein paar Ideen vermitteln.

Mitglieder des Vereins können an Schachturnieren teilnehmen. Ein **Schachturnier** wird durch einen Turniercode eindeutig identifiziert, wobei folgendes gilt[4]:

Der Turniercode bildet folgende Informationen ab:
- **Turniertermin** (5 Zeichen)
  - zwei Stellen für das Kalenderjahr (codiert, B3 = 2013, …, B8 = 2018, B9 = 2019, C0 = 2020, C1 = 2021, …)
  - Kalenderwoche (3. und vierte Stelle)
  - – (Bindestrich)
- Organisationsebene (Bezirk/Landkreis)
- Turnierart- und form (z.B. U8m)

Beispielsweise ist der Turniercode für die Bezirkseinzelmeisterschaft (BEM) im Jahr 2019 für die Altersklasse U8m der folgende:

B909-**F20**-**08M**

[2] https://de.wikipedia.org/wiki/Deutsche_Wertungszahl
[3] https://de.wikipedia.org/wiki/Elo-Zahl
[4] https://www.schachbund.de/id-75-turniercode.html

Die **Teilnehmer** eines Turniers sind entweder 1…n Mannschaften eines Vereins oder einzelne Spieler, die in einer bestimmten **Altersklasse** (U8, U10, U12, U14, U16, U18[5]) paarweise in mehreren Runden gegeneinander spielen.

Eine **Mannschaft** besteht aus mehreren Spielern und hat einen Mannschaftsnamen.

Eine **Runde** findet an einem bestimmten Veranstaltungsort (Adresse) statt. Ein Spieler (einer Mannschaft) spielt in einem **Spiel** die weißen Schachfiguren[6] der andere Spieler die schwarzen Schachfiguren. Ein Spiel endet mit dem Sieg für Weiß (1:0), dem Sieg für Schwarz (0:1) oder unentschieden/Remis[7] ($\frac{1}{2}:\frac{1}{2}$).

Schachturnier

- Turniercode (z.B. B909-**F20-08M** siehe oben)
- Bezeichnung (z.B. BEM Spielbezirk Dresden U8M)
- Turnierstart
  - Datum/Uhrzeit
- Turnierende
  - Datum/Uhrzeit
- Meldetermin (<= Veranstaltungstermin)
- Teilnahmegebühr
- Runde(n)
  - Rundennummer (1. Runde, 2. Runde, …)
  - Veranstaltungsort (Adresse)
  - Rundenbeginn
    - Datum
    - Uhrzeit
  - Rundenart
    - Vorrunde
    - Zwischenrunde
    - Endrunde
    - Keine Einteilung
- Veranstalter
- Mannschaftsleiter der entsprechenden Runde(n) (Person)

[5] Ggf. noch unterteilt in männlich und weiblich, d.h. U8m, U8w, 10m, U10w usw.
[6] https://de.wikipedia.org/wiki/Schachfigur
[7] https://de.wikipedia.org/wiki/Remis