Projekt Tischtennis Verwaltung

16.03.2016

# Anforderung

Es soll eine Tischtennis-Verwaltung programmiert werden, dabei sollen alle Themen aus OCP Anwendung finden (insbesondere Threading und Streams) und möglichst viele Entwurfsmuster verwendet werden.

Es soll testgetrieben entwickelt werden.

Spätere Erweiterungen:

1. Erweiterung/Austausch der Basis, alles auf das Spring-Framework setzen.
2. Implementierung eines UIs, so dass die Verwaltung im Liferay-Portal stattfinden kann.

Es darf viel erweitert oder umgebaut werden. Dabei wird Refactoring betrieben.

# Datenmodell

## Spieler

* Eigenschaften
  + Geschlecht: Männlich oder Weiblich
  + Person: rechts- oder linkshänder
  + Punktezahl/Spielstärke: ergibt sich durch das Spiel mit anderen
  + Verein
  + Mannschaft
  + Jeder Spieler hat 1 Scläger, ohne Schläger kann er nicht spielen
* Invarianten
  + Jeder Spieler muss in genau einem Verein spielen
  + Wenn er bereit ist in der Mannschaft zu spielen, kann er für eine Saison in der Mannschaft gemäß Spielstärke spielen.
  + Neue Spieler haben die Punktezahl 0.
* Bemerkungen
  + Es gibt die Unterscheidung in M/W, aber alle Mannschaften sind gemischt.
  + Alle Spieler sind prinzipiell Mannschaftsspieler, d.h. es gibt kein Flag, ob ein Spieler nicht in der Mannschaft spielen will.

## Verein

* Eigenschaften:
  + Stadt -> Bundesland
  + Name = TTV Stadt + Jahreszahl der Gründung
* Liste der Mitglieder nach Spielstärke geordnet, die können in den Mannschaften in einer Saison spielen.
* Jeder Verein gehört zu einer Stadt, die zu einem Bundesland gehört
* Die Vereine verlieren keine Mitglieder, Mitglieder können den Verein nicht wechseln
* Invarianten
  + Es gibt keinen Spieler in einer höheren Mannschaft, der eine geringere Spielstärke als irgendeins Spieler in einer unteren Mannschaft hat
  + Jahreszahl liegt zw. 1930 und 2010
  + Die Liga muss zu dem Bundesland passen, in dem die Stadt liegt

## Mannschaft

* Alle Spieler sind immer einsatzbereit.
* Spielt pro Saison in einer Liga
* Name: Vereinsname + Nummer
* Invarianten:
  + Alle Spieler mit höherer Nummer müssen auch eine bessere Spielstärke als die Spieler mit kleinerer Nummer haben.
  + Die gemeldete Liga muss zur Stadt des Vereins passen.

## Liga

* Hat höchstens eine übergeordnete Liga
* Kann mehrere untergeordnete Ligen haben
* Invarianten
  + Wenn Liga A in der Liste der untergeordneten Ligen von Liga B ist, muss A auch die übergeordnete Liga von B sein
  + In jeder Liga kann von jedem Verein nur genau eine Mannschaft vertreten sein.

## Saison

* Repräsentiert eine Spielzeit
* Alle Mannschaften aller Ligen spielen gegeneinander
* Eigenschaften
  + Hat Name, der sie eindeutig identifiziert
  + Hat Anfangs- und Enddatum
  + Liste der Spiele, die stattgefunden haben
* Ablauf
  + 0) Alle Vereine besetzen ihre Mannschaften der jew. Liga mit Spielern
  + 1) Für jede Liga: jede Mannschaft spielt gegen jede andere (s. Mannschaftsspiel)
  + 2) am Ende:
    - Die letzten Mannschaften steigen ab in die untere Liga gemäß Stadt  
      Aber nur, wenn die jew. Liga voll ist.
    - alle ersten Mannschaften gleichgestellter Ligen spielen gegeneinander.   
      Die besten Mannschaften steigen auf, je nach Anzahl der Plätze.  
      In der Bundesliga wird die beste Mannschaft zum „Meister“ gekürt.
  + Es wird ein ausführlicher Spielbericht über alle Spiele erstellt
* Funktionalität
  + Zu jeder Saison gibt es detaillierte Auswertungen über einen Berichtsgenerator

## Mannschaftsspiel

* Eigenschaften: Gastmannschaft, Heimmannschaft
* Modus: jeder Spieler gegen jeden
* Einzelspiel:
  + 3 Gewinnsätze, jew. Bis 11, am Ende zwei Unterschied
  + Jeder Ballwechsel dauert die Zeit T (z.B. eine Sekunde)
  + 6er M. spielen an 3 Platten, 4er M. an 2
  + Sowie eine Platte frei wird, kann ein neues Spiel daran stattfinden
  + In das Spiel fließt die Spielstärke ein.   
    Bei jedem Ballwechsel ergibt sich der Ausgang durch Zufall gewichtet aufgrund der Spielstärke.
  + Es gibt ein detailliertes Protokoll
  + Nach jedem Spiel wird die Spielstärke des Siegers erhöht:
    - 1 Pkt. Bei gleicher Spielstärke
    - 2 Pkt. Wenn besserer Spieler geschlagen wird.
* Invarianten
  + Gastm. Und Heimm. Haben gleich viele Spieler.

## TT-Welt

* Ist die Klammer außenrum.
* Es gibt eine Liste aller Vereine
* Es gibt eine Liste der Liegen
* Funktionalität
  + Speichern u. Laden
  + Anlegen und durchführen einer Saison
* Struktur
  + Bundesliga
    - Landesliga Nord
      * Regionalliga Schleswig-Holstein
      * RL Niedersachsen
      * RL Berlin-Brandenburg
    - LL Mitte
      * RL Pfalz
      * RL NRW
      * RL Hessen
      * RL Sachsen
    - LL Süd
      * RL BaWü
      * RL Bayern
* Mannschaftsgröße
  + 6er Mannschaften
* Ligengröße:
  + BL: bis zu 6 M
  + LL: bis zu 5 M
  + RL: bis zu 10 M
* Es wird immer saisonweise gespielt (S. Saison)
  + Am Ende der Saison wird
    - Die Auf- und Abstiege durchgeführt
    - Der TT-Welt-Meister gekürt

# ImplementierungsDetail

### OCP

1. Die Protokolle der einzelnen Spiele und der gesamten Saison werden in das **Dateisystem** geschrieben

* Inhalte über **io-Streams**
* Pfad-Verwaltung mit **nio**

1. Für jeden Spieler wird in einer **JDBC-Datenbank-Tabelle** gespeichert, welche Spiele er durchgeführt hat (Mannschaft, Datum, Gegner) und wie sich seine Spielstärke entwickelt.
2. Die Spiele einer Saison werden als **Queue** zusammengestellt und mit **Multi-Threading** nacheinander ausgeführt.
3. Die Mannschaften müssen nicht immer neu gebildet werden, sondern werden über **Objektserialisierung** geladen und gespeichert.
4. Es muss ein ausführliches **Exception**-Handling geben.   
   Dazu wird eine kleine Hierachie von neuen Exceptions eingeführt, die auch sporadisch mal geworfen werden können.
5. Der Berichtsgenerator wird über **Streams und Lambdas** implementiert.
6. Jeder Verein liefert seine **Liste** der Mannschaften sortiert nach Spielstärke
7. Die Spieler des Vereins werden als **Set** gespeichert.
8. Zur Ausgabe z.B. der Saison wird die **toString()-**Funktion verwendet.
9. Die Regionen, Bundesländer und Städte werden über Enums angelegt.
10. Die Spieler werden über interfaces abgehandelt.

### Patterns

1. Die Schläger sind immutable Objects, die über eine Factory hergestellt werden
2. Die Spieler werden über einen Builder gebaut
3. Für die Verwaltung der Vereine gibt einen Vereinsmanager als Singleton

# ERweiterungsideen

* Klasse Mensch verallgemeinert: Verein bekommt Vorsitzenden etc. : Refactoring
* Spieler können auch mal krank werden, d.h. Ersatzspieler
* Ligen sind verteil über unterschiedliche Zeitzonen Spezifisch auszuschließende Bereiche
* Die Spieler können zwischen den Saisons den Verein wechseln   
  und auch ganz aus dem Verein austreten
* Die Namen der Spieler werden über einen Webservice ausgelesen
* Spieler haben Alter: Altersklassen ??
* Mannschaften unterschiedlicher Größe
* Unterscheidung nach Geschlecht: Männer- und Frauen-Mannschaften, -Ligen etc.