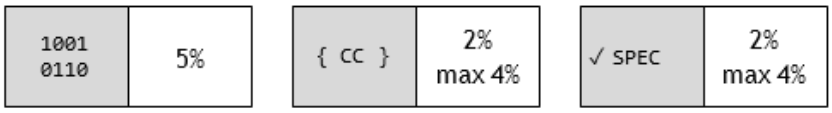


Opdracht

We ontwikkelen een platform om de prestaties van atleten op de Olympische spelen bij te houden, meer bepaald voor de atletiek atleten. Door middel van het platform kunnen prestaties van de atleten op de verschillende atletiekdisciplines geregistreerd worden en kunnen we op een scorebord de uitslag bekijken. Lees eerst de volledige opgave aub!

Zorg dat je geen **System.out.println** toevoegt in je code. Indien je informatie wenst weg te schrijven maak je gebruik van het logging framework **log4j** en gebruik je ook een aangepast log level. Log4j2 kan zonder extra dependencies of configuratie gebruikt worden in de toepassing.

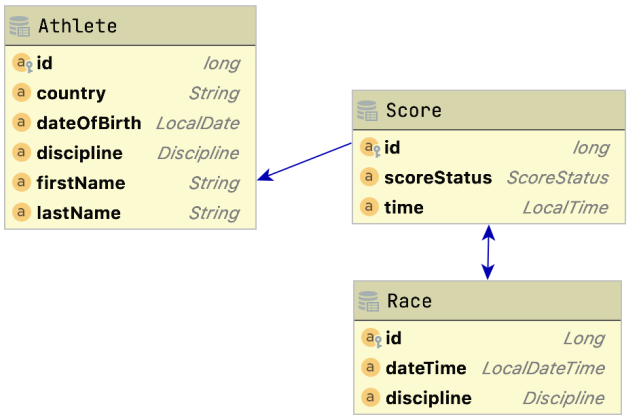
Het project is reeds geconfigureerd om een in-memory h2 databank te gebruiken. Wanneer de toepassing wordt opgestart wordt de databank aangemaakt en de run-methode van de klasse *be.pxl.paj.olympicgames.config.ImportData* wordt uitgevoerd.

Implementeer de opgegeven functionaliteit en voeg ook de gevraagde unit testen toe. Zorg ervoor dat je in je implementatie een goede opdeling maakt tussen **persistence-laag, business-laag** en **presentatie-laag**.

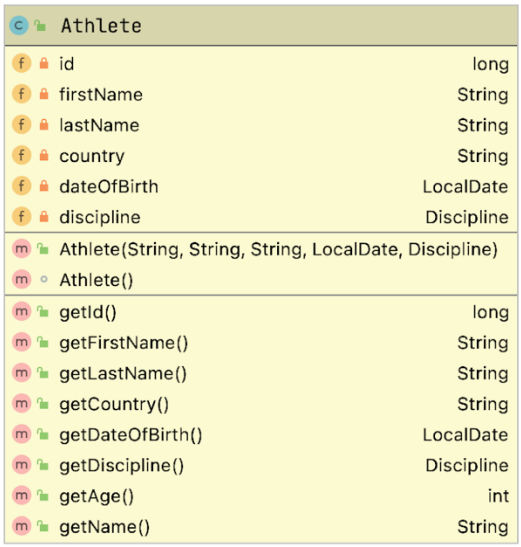
Opmerkingen:

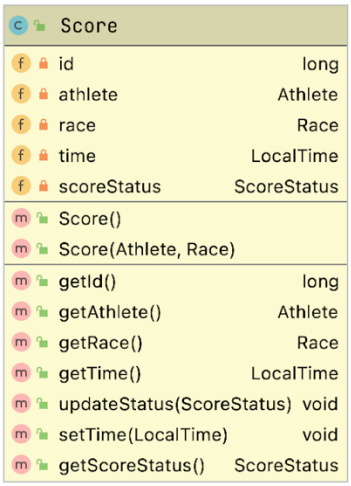
- Je mag het poortnummer van je application-server aanpassen.
- In de resources-folder vind je het script examen_valideren.sh dat via curl de Rest-endpoints en de servlet aanroept. De verwachtte output op het script vind je terug op de laatste pagina van deze bundel. Dit script zal gebruikt worden voor het quoteren van je oplossing.

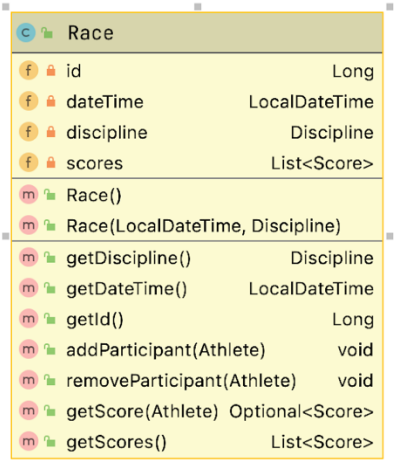
Database schema:

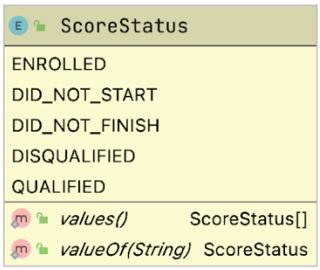


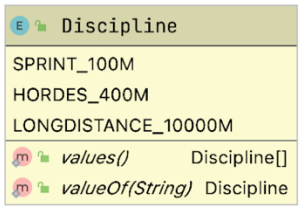
Domein-model:











In het bovenstaande schema vind je de entity-klassen die je moet vervolledigen. Maak gebruik van strategy **identity** voor het genereren van de primary keys van de entities (anders zullen de id's gebruikt in het script examen_valideren.sh niet correct zijn). De entity-klasse Athlete gebruiken we om de atleten te beheren. Iedere atleet is gespecialiseerd in één discipline: SPRINT_400M, HORDES_400M of LONGDISTANCE_1000M.
De entity-klasse Race bevat informatie over een race en over de atleten die deelnemen aan de race. Een atleet is gespecialiseerd in 1 discipline. Bij het toevoegen van de atleet aan een race moet er geverifieerd worden op discipline of hij mag deelnemen aan de race. Wanneer een atleet zich inschrijft voor een race wordt er een object van de entityklasse Score aangemaakt. Dit Score-object heeft status ENROLLED. Nadat de race gelopen is passen we een Score-object aan met status QUALIFIED en de gelopen tijd van de atleet. Wanneer een atleet niet aan de start verschijnt krijgt het Score-object status DID_NOT_START en wanneer een atleet gediskwalificeerd is (bijv. omwille van valse start) krijgt het Score-object status DISQUALIFIED.

Taken uit te voeren tijdens het examen:

A. Entity klassen vervolledigen
B. Een nieuwe race aanmaken
C. Atleten toevoegen aan of verwijderen uit een race
D. Resultaten van een race registreren
E. Unit Testen
F. Servlet om een scorebord van een race te tonen

A. Entity klassen vervolledigen

Zorg ervoor dat de entity-klassen correct geannoteerd zijn en leg de juiste relaties tussen de entity-klassen.

Een Rest endpoint om alle atleten op te vragen is **reeds voorzien**. Als je de entity-klassen correct hebt geannoteerd, start de applicatie op en kan je het volgende endpoint uittesten.

GET http://localhost:8082/olympicgames/athletes
Response 200 OK


```
[
  {
    "id": 1,
    "firstName": "Darron",
    "lastName": "Fahey",
    "country": "SZ",
    "age": 44
  },
  {
    "id": 2,
    "firstName": "Ole",
    "lastName": "Klein",
    "country": "VG",
    "age": 36
  },
  {
    "id": 3,
    "firstName": "Toni",
    "lastName": "Douglas",
    "country": "HR",
    "age": 17
  },

  … (En nog veel meer athleten, maar de json is hier even ingekort…)
]
```


B. Een race aanmaken

Voorzie een Rest endpoint en implementeer de business-logica zodat je een nieuwe race kan toevoegen in het systeem. Zowel de dateTime als discipline zijn verplichte velden bij het aanmaken van een nieuwe race. Verder zijn er geen beperkingen.

POST http://localhost:8082/olympicgames/races

```
Request body:
{
    "dateTime": "2021-09-09 13:15:00",
    "discipline": "SPRINT_100M"
}
```

Voorzie nu een Rest endpoint waarmee je alle races kan opvragen uit het systeem. Maak in de business-laag gebruik van de klasse RaceDTO die reeds voorzien is in startcode.

GET http://localhost:8082/olympicgames/races

C. Atleet toevoegen aan of verwijderen uit een race

Implementeer een Rest endpoint (PUT) waarmee je een atleet toevoegt aan een race.

```
PUT http://localhost:8082/olympicgames/races/{raceId}/{athleteId}
Responses:201 CREATED, 404 NOT FOUND, 400 BAD_REQUEST
```

Zoek aan de hand van de athleteId de atleet op en controleer of hij mag deelnemen aan deze race voor deze discipline. Is dit niet het geval return je http status code 400 (BAD_REQUEST). Vind je helemaal geen race of atleet voor het meegeven ID, dan return je 404 (NOT_FOUND). Wanneer de atleet mag deelnemen aan de race, maak je een Score-entity aan met status ENROLLED en sla je alles op.
Tip: gebruik de methode addParticipant uit de klasse Race.

Om atleten terug te verwijderen uit de race voorzie je eveneens een Rest endpoint (DELETE).

```
DELETE http://localhost:8082/olympicgames/races/{raceId}/{athleteId}
Responses:
200 OK, 404 NOT FOUND
```

Tip: gebruik de methode removeParticipant uit de klasse Race.

D. Resultaat van een atleet voor een gelopen race registreren

Voorzie een Rest endpoint waarmee je de status van een atleet kan aanpassen voor een race. Bij de implementatie van dit Rest endpoint maak je gebruik van de bestaande methode registerResult in de klasse *OlympicGamesService*.

```
POST http://localhost:8082/olympicgames/races/{raceId}/{athleteId}
```

Request body indien disqualified:

```
{
   "state": "DISQUALIFIED"
}
```

Request body indien qualified:

```
{
   "status": "QUALIFIED",
   "time": "00:00:09.800"
}
```

```
Responses:
200 OK, 404 NOT FOUND
```

E. UNIT TESTEN

E1. Schrijf 3 unit testen voor Rest endpoints uit opgave B en maak hierbij gebruik van MockMvc.

- Schrijf 1 test voor een **POST request met status ok** (200).
- Schrijf 1 test voor een **POST request met een error** (500 of andere).
- Schrijf 1 test voor een **GET request** met status ok (200).

E2. Unit testen voor OlympicGamesService

In de klasse OlympicGamesService is de businesslogica voor het registreren van scores reeds voorzien in de methode registerResult.
Implementeer **3 relevante unit testen** voor deze methode *(public void registerResult(Long raceId, Long athleteId, RegisterScoreCommand registerScoreCommand))* in de klasse OlympicGamesService. Maak bij de implementatie van de testen gebruik van **Mockito**.

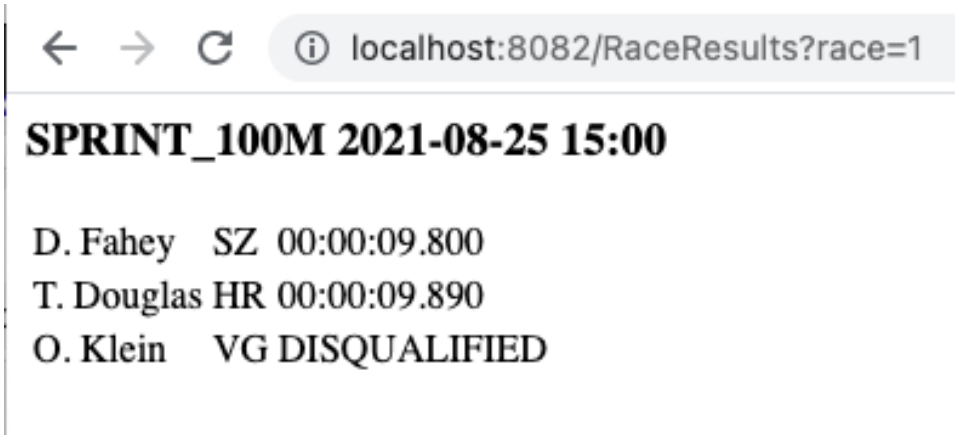
F. Servlet toont een scorebord van een race.

Zorg dat de servlet een info-boodschap toont (log4j) wanneer de servlet wordt aangemaakt. Zorg dat de servlet gebruikmaakt van de OlympicGamesService om de race op te zoeken. Indien er geen race bestaat met de opgegeven id, dan geef je een duidelijk boodschap. Indien de race wel gevonden wordt, dan toon je de datum, de discipline en de atleten in een lijstje gesorteerd op beste tijd.
Gediskwalificeerde atleten staan onderaan dit lijstje met "DISQUALIFIED" achter hun naam.
Wanneer de servlet wordt opgeruimd, toon je opnieuw een info-boodschap in de logging van je applicatie.

Voorbeeld van de output van de servlet:

```
2021-08-12 10:36:37.856  INFO 3026 --- [nio-8082-exec-2] b.p.p.o.servlet.OlympicGamesServlet: Initialize OlympicGamesServlet...
2021-08-12 10:36:37.858  WARN 3026 --- [nio-8082-exec-2] b.p.p.o.servlet.OlympicGamesServlet: Retrieving race with id [1]
2021-08-12 10:36:49.497  INFO 3026 --- [ionShutdownHook] b.p.p.o.servlet.OlympicGamesServlet: Destory OlympicGamesServlet...
```

Resultaat in je browser:



Veel succes!

Verwachtte output van het script

```
$ ./examen_valideren.sh
Enter port number:
8082

++++++++++++++++++++++++++++++++++++++++++++++++++

CONTROLE 1: OVERZICHT ATLETEN

[{"id":1,"firstName":"Darron","lastName":"Fahey","country":"SZ","age":44},{"id":2,
"firstName":"Ole","lastName":"Klein","country":"VG","age":36},{"id":3,"firstName":
"Toni","lastName":"Douglas","country":"HR","age":17},{"id":4,"firstName":"Amina","
lastName":"Stoltenberg","country":"LY","age":16},{"id":5,"firstName":"Burley","las
tName":"Conn","country":"EE","age":37},{"id":6,"firstName":"Norris","lastName":"Tu
rner","country":"VU","age":29},{"id":7,"firstName":"Hollis","lastName":"Koelpin","
country":"BN","age":35},…,{"id":25,"firstName":"Rafaela","lastName":"Gleason","cou
ntry":"KZ","age":40},{"id":26,"firstName":"Viva","lastName":"Ledner","country":"ES
","age":37},{"id":27,"firstName":"Jannie","lastName":"Hoppe","country":"ER","age":
44},{"id":28,"firstName":"Mose","lastName":"Abbott","country":"PE","age":31},{"id"
:29,"firstName":"Deondre","lastName":"Carroll","country":"SY","age":32},{"id":30,"
firstName":"Libby","lastName":"Reichel","country":"CU","age":25}]
++++++++++++++++++++++++++++++++++++++++++++++++++

CONTROLE 2: RACES TOEVOEGEN

++++++++++++++++++++++++++++++++++++++++++++++++++

CONTROLE 3: JOIN RACE VALID

++++++++++++++++++++++++++++++++++++++++++++++++++

CONTROLE 4: JOIN RACE INVALID

{"message":"Athlete [15] is not allowed to participate in race [1]"}
{"message":"Athlete [4] is not allowed to participate in race [2]"}
```

++

CONTROLE 5: LEAVE RACE

++

CONTROLE 6: RACE RESULTS

++

CONTROLE 7: SERVLET OUTPUT RACE 1

```
<html>
<body>
<h3>SPRINT_100M 2021-08-25 15:00</h3>
<table>
<tr><td>D. Fahey</td><td>SZ</td><td>00:00:09.800</td></tr>
<tr><td>T. Douglas</td><td>HR</td><td>00:00:09.890</td></tr>
<tr><td>O. Klein</td><td>VG</td><td>DISQUALIFIED</td></tr>
</table>
</body>
</html>
```

++

CONTROLE 8: SERVLET OUTPUT RACE 2

```
<html>
<body>
<h3>LONGDISTANCE_10000M 2021-09-03 22:22</h3>
<table>
<tr><td>A. Kihn</td><td>FI</td><td>ENROLLED</td></tr>
<tr><td>D. Hagenes</td><td>GR</td><td>ENROLLED</td></tr>
<tr><td>M. Wolf</td><td>EG</td><td>ENROLLED</td></tr>
</table>
</body>
</html>
```

++