Evaluation of `https://github.com/m-mato/sem`

Jan Stourac, 113869

November 2, 2016

Checklist evaluation

[**OK**]	`mvn clean install` without problems, all tests are passing	
[**OK**]	every team member contributed to entitity, DAO and tests (as requested)	
[**NOK**]	`hashCode()` and `equals()` 1. possible `NullPointerException` in `Result` entity when checking `note` (both methods, even without constraint violations) 2. if `Results` entity is not yet persisted in DB (e.g. `id` is null) then `equals()` is always false with the same data except when testing the same instances (`obj.equals(obj)`)	-1
[**NOK**]	unit tests for DAO 1. testing `EventDAO` should also contain tests for `findBy*()` using `null` parameters (necessity of these tests is emerged by explicit mentioning `IllegalArgumentException` in javadoc and explicit `null` checking in method implementation) 2. test `findBy*NotFound()` in `SportsmanDAOTest` is correct in functionality, but incorrect according to javadoc – such methods should return `null` when sportsman(s) matching criteria are not present in DB and not an empty list	-1
	Total points:	**8**

General comments

- `@author` tag should be more readable than `xaksamit`
- is there any reason for not using `@Transactional` in DAO implementations?
  - if *not*, then why only `SportsDAO` is transcational? The reason should be at least documented...
  - if *yes*, then there will be problem in `delete()` when using from transactional class (detached entity)
- unify ways of writing (or auto-generating) `equals()` and `hashCode()` – e.g. difference between `getClass() == o.getClass()` and `instanceof` if there is no explicit reason for different behaviour
- consider using `Objects.hashCode()` and `Objects.equals()` which handles `null` to prevent unnecessary complex code (like in `Sport`) or mistakes (`NullPointerException` in `Result`)
- maybe unnecessary handling `null` values in DAO implementation (context manager did it and throws exception by default)
- catching exception, which should not be thrown (in DAO when using `getResultsList()`)
  - and even if this catching is not an issue, why sometimes is returned empty list and sometimes `null` (and the documentation saying "if nothing is found, then `null` is returned" is wrong – try failing `assertNull(sportsmanDAO.findBySurname("some nonexistent surname"))`)
- test for `findAll()` method for should be more powerful than just `assertNotNull(results)`
- I don't see the reason for using `@Table` annotations in every entity except `Sport`
- I don't see the reason for naming find-all method as `getAll()` in `ResultsDAO` while elsewhere it is `findAll()`

Strong points

- complex DAO (containing advanced `findBy*()` methods)
- universal entity generators (no code duplication) in tests
- connections between tables
- suitably named variables/attributes

Weaknesses

- not complete testing
- unnecessary exception/null testing
- confusing usage of annotations (`@Transactional`, `@Table`)
- not unified ways for writing/generating `equals()` and `hashCode()` methods

Other notes

- consider using autoformatting functionality from IDE to achieve similiar code formatting, spacing, enclosing in brackets and indentation