UTCN
Computer Science Department
Software Design 2018

ASSIGNMENT A2

---

1. Objective

The objective of this assignment is to allow students to become familiar with the Model View Controller architectural pattern and the Abstract Factory design pattern.

2. Application Description

Since the Ping-Pong Tournaments application is such a big success, the owners of the application wish to add a new feature to it: Paid Tournaments. They differ from the free tournaments available until now by the fact that they require an enrollment fee and offer a cash prize to the winner. For the moment, the player that finishes on the 1st position receives all the money from the enrollment fees of the tournament. From the beginning of the tournament until a player wins 1st place, the prize money is kept in the account of the Ping-Pong Association or in the account of the Tournament itself.

Based on Assignment A1, adapt your application to fit the new requirements.

The regular user can perform the following **additional** operations:

- Enroll into upcoming Tournaments, by paying the enrollment fee out of their account.
- View Tournaments by category: Finished, Ongoing, Enrolled, Upcoming.
- Search Tournaments by name and by type (free / paid)

The administrator user can perform the following **additional** operations:

- CRUD paid tournaments.
- Add money to any player's account.
- Withdraw money from any player's account

Every new upcoming tournament must be created with at least one month before its start date and must have an enrollment fee specified at creation time.

3. Application Constraints and Technical Requirements

- Use the Model View Controller Pattern for all views.
- The Data Access Layer (DAL) will be re-implemented using an ORM framework.
- The application will use a config file from which the system administrator can set which DAL implementation will be used to read and write data to the database.
- Use the Abstract Factory design pattern to switch between the new and the old DAL implementation (implemented with JDBC or equivalent)
- All the inputs of the application will be validated against invalid data before submitting the data and saving it. (e.g. Tournament start date, Mandatory fields, The player has enough money in his account, etc.)
- Write at least one Unit Test for each new method in the business layer (e.g. Test that you can transfer money from the Player's account to the account of the Ping-Pong association).

4. Deliverables

In the same Github repository as Assignment 1, add a new folder with the following files:

- Analysis and design document.
- Implementation source files.
- SQL script for migrating (altering) the database. When writing the analysis and design document, **specify if the players lose any data**. (e.g. they lose their entire account, they lose the won games, etc.)
- Readme file that describes the **installation process** of the application **and how to use it**.

5. Grading

Grade	Functionality
5	Analysis and design document MVC Structure Abstract Factory for DAL ORM Implementation (no new queries) View Tournaments by category
6	Add money to account Withdraw Money from account
7	CRUD on paid tournaments Search Tournaments
8	Enroll into tournaments Transfer money into winner's account
9	Read Config file and choose DAL New Unit Tests
10	Quality of Implementation and Documentation (Validated Input Data + Code + UI)

6. References

https://en.wikipedia.org/wiki/Abstract_factory_pattern#Structure
https://sourcemaking.com/design_patterns/abstract_factory/java/1
https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm
http://www.austintek.com/mvc/
https://docs.oracle.com/javafx/2/best_practices/jfxpub-best_practices.htm#sthref17
https://www.youtube.com/watch?v=LMdjhuYSrqg&t=0s&list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG&index=32
https://www.developer.com/java/data/understanding-java-observable-and-javafx-observable.html
https://en.wikipedia.org/wiki/Object-relational_mapping
https://www.journaldev.com/3793/hibernate-tutorial (1-5)