<Ping Pong Application>

Analysis and Design Document

Student:Ispas Ovidiu Mihai

**Group:30433**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Use JAVA/C# API to design and implement an application for a ping pong association that organizes tournaments on a regular basis. Every tournament has a name and exactly 8 players (and thus 7 matches). A match is played best 3 of 5 games. For each game, the first player to reach 11 points wins that game, however a game must be won by at least a two point margin. The application should have two types of users: a regular user represented by the player and an administrator user. Both kinds of uses have to provide an email and a password in order to access the application.

The regular user can perform the following operations:

- View Tournaments

- View Matches

- Update the score of their current game. (They may update the score only if they are one of the two players in the game. The system detects when games and matches are won)

- Enroll in a tournament

- Search a tournament

- View tournaments by category (Upcoming ,Ongoing, Enrolled, Finished, Free, Paid).

The administrator user can perform the following operations:

- CRUD on player accounts

- CRUD on tournaments: He creates the tournament and enrolls the players manually.

- Withdraw money

- Deposit money

# Functional Requirements

* Players and administrator have the possibility to log in
* Players can view the available tournaments
* Players can view the matches of that tournaments
* Players can update the score of the match they are currently in
* Players can enroll themselves in a tournament
* Players can search for a tournament
* Players can view filtered tournaments
* Administrator can create players
* Administrator can update, view and delete the players already in the system
* Administrator can create tournaments (free or paid)
* Administrator can update, view and delete the tournaments already in the system
* Administrator can enroll the available players in the available tournaments
* Administrator can deposit money into players’ account
* Administrator can withdraw money from players’ account

# Non-functional Requirements

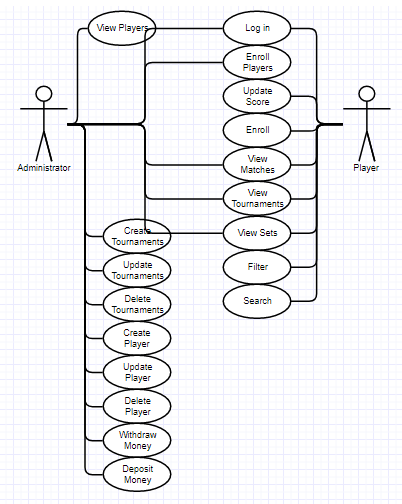
- The data will be stored in a database

- Use Layers and Model-View-Controller architectural design pattern to organize the application

- Use a domain logic pattern or a data source hybrid pattern and a data source pure pattern

- All the inputs will be validated

2. Use-Case Model



Use case : Create a player

Level : user-goal level

Primary actor : Administrator

Main success scenario : 1. Complete the log in fields with prerequisites

2. Click on Log in button

3. Go to players window by clicking on Players button in the window that show up

4. Complete all the fields necessary : Name, Age, Username and Password

5. Click on Insert button

Extensions :

User didn’t write the prerequisites right in the log in fields :

1. Complete the log in fields with prerequisites
2. Click on Log in button
3. Complete the log in fields with the right prerequisites
4. Click on Log in button

5. Go to players window by clicking on Players button in the window that show up

6. Complete all the fields necessary : Name, Age, Username and Password

7. Click on Insert button

User didn’t complete all the necessary fields for creating a player :

1. Complete the log in fields with prerequisites

2. Click on Log in button

3. Go to players window by clicking on Players button in the window that show up

4. Complete all the fields necessary : Name, Age, Username and Password

5. Click on Insert button

6. Click on the wrong created player and select the Delete button

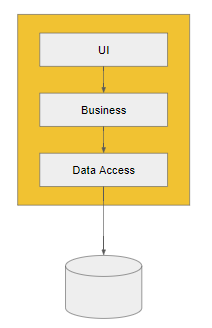
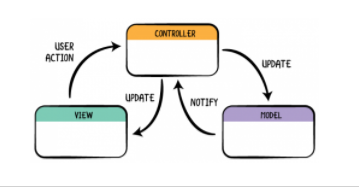
7. Complete all the fields necessary

8. Click on Insert button

3. System Architectural Design

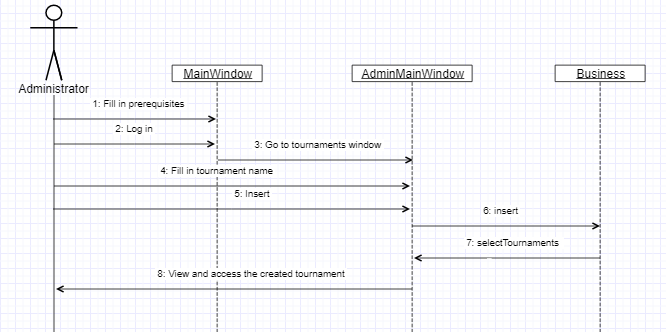
**3.1 Architectural Pattern Description**

The architectural design pattern used for this project are the Layers pattern and the Model-View-Controller pattern. The first one’s principle is separating the data access part from the logic business part and both of them from user interface part. So, eventually, we have three layers : Data Access Layer, Business Layer and Presentation Layer. This segmentation leads to the fact that Presentation Layer has no direct access to the Data Access Layer, so everything needed will be passed through Business Layer. The second one is based on having a Controller class which makes the relations between the business part (entities/DTOs and logic methods) – Model and the graphical user interface – View.

**3.2 Diagrams**

4. UML Sequence Diagrams

Scenario : Administrator creates a new tournament



5. Class Design

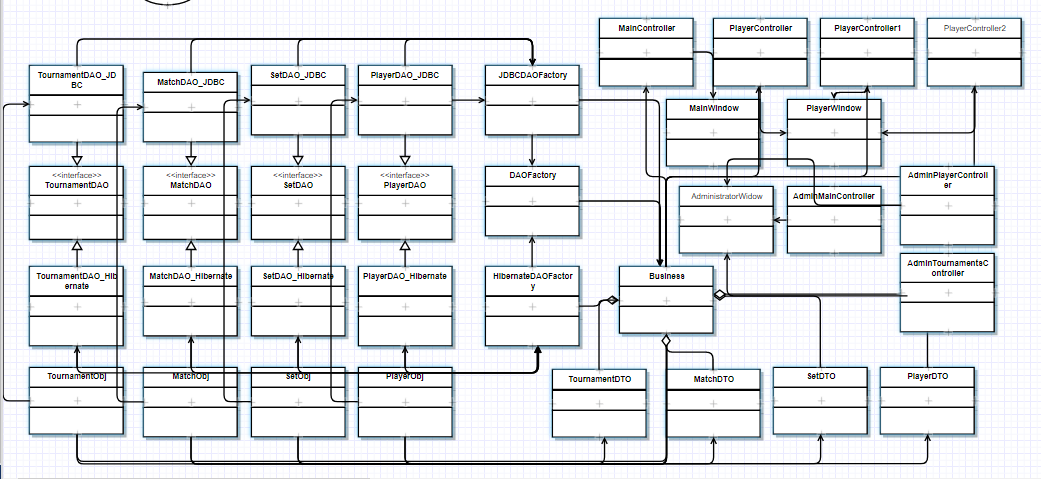
**5.1 Design Patterns Description**

At the Data Access Layer, I used Table Data Gateway pattern, which is based on having a Data Access Object for every table in the data base. A Data Access Object is an object that contains all the methods of inserting, updating, deleting and selecting all the necessary data.

Observable design pattern is a pattern I used in order to update the graphical user interface with the updated data whenever an action that requires it is done.

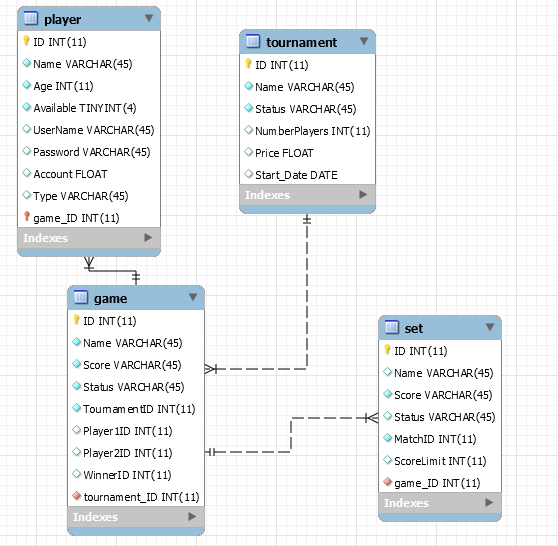
Abstract factory design pattern is a pattern I used in order to help switching easier between the two data access frameworks used : JDBC and Hibernate.

* 1. **UML Class Diagram**



1. Data Model

The Data Model I used for organizing the data for this project is the Relational Model , which is based on storing all the necessary data in 2 dimensional tables which are linked between each other by using an additional field which contains the ID from the table you want to make the one-to-many relationship.



1. System Testing

For the system testing , I used a unit test on the only method which contains some logic , which is the one that updates the score of the match in which a player with a given ID is enrolled.

Another 2 unit tests are on the deposit and withdraw money methods from a player’s account.

8. Bibliography