Assignment 1

Student: Danila Lucia-Diana
Group:30433

Table of Contents

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.

- **Functional Requirements**

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)

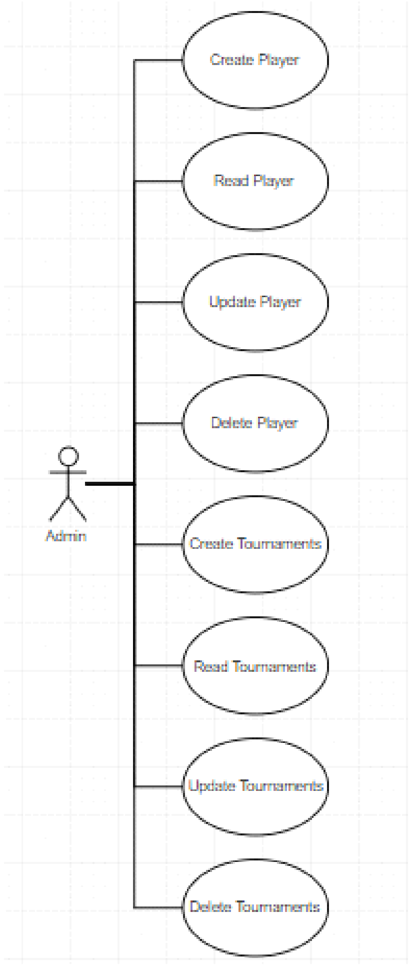
The administrator user can perform the following operations:

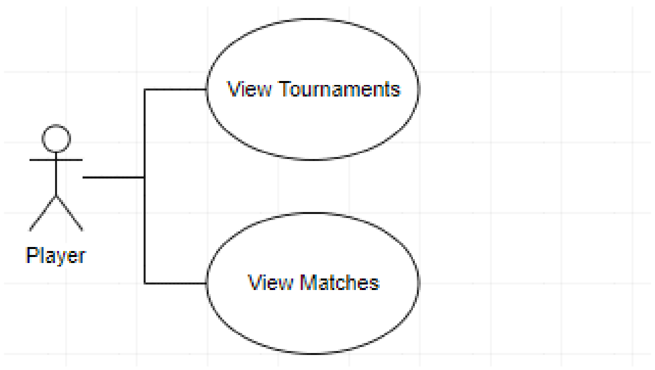
- CRUD on player accounts
- CRUD on tournaments: He creates the tournament and enrolls the players manually.

- **Non-functional Requirements**
  The data will be stored in a database. Use the Layers architectural pattern to organize your application. Use a domain logic pattern (transaction script or domain model) / a data source hybrid pattern (table module, active record) and a data source pure pattern (table data gateway, row data gateway, data mapper) most suitable for the application.

  All the inputs of the application will be validated against invalid data before submitting the data and saving it in the database.

2. Use-Case Model





Use case: delete player
Level: user-goal level
Primary actor: admin
Main success scenario: admin logs in -> views the player table -> choses the id of the player to be deleted -> insert id -> player deleted
Extensions: failure -> to log in or invalid id for player to be deleted

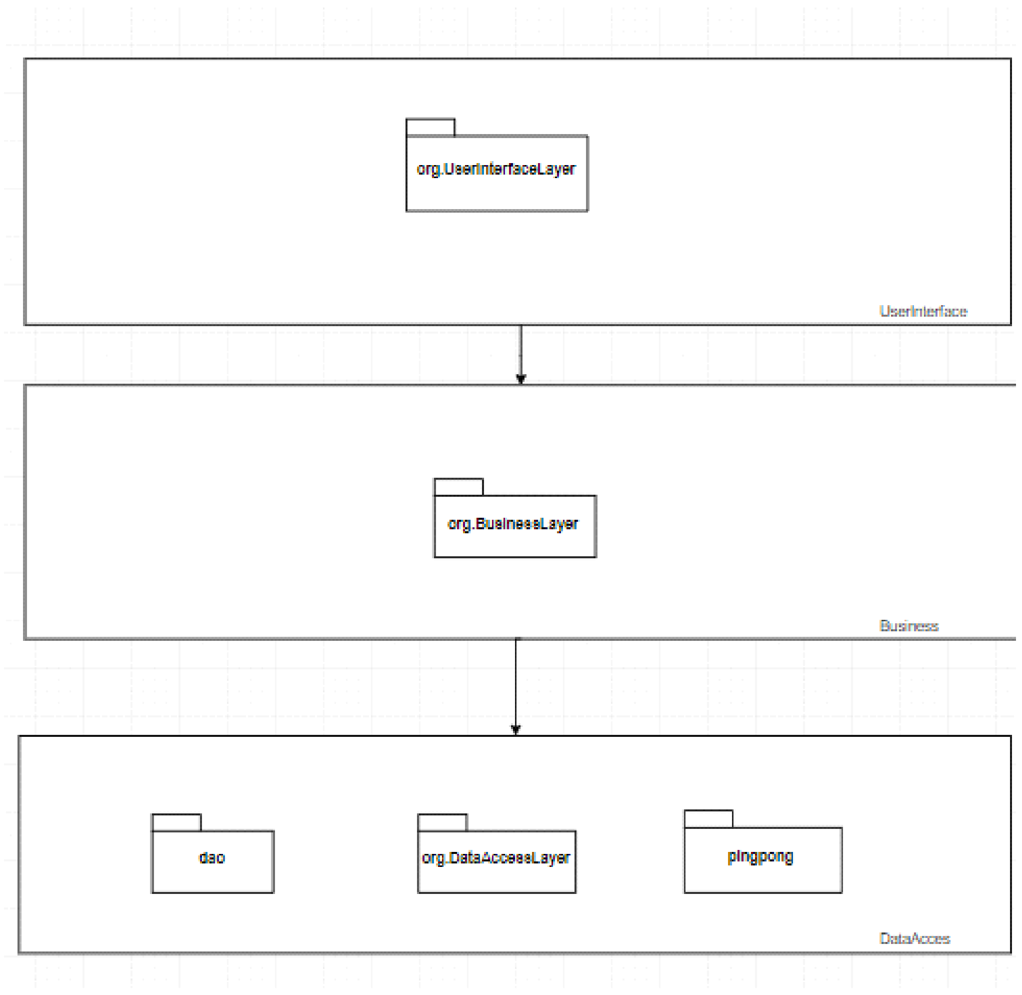
3. System Architectural Design

3.1 Architectural Pattern Description

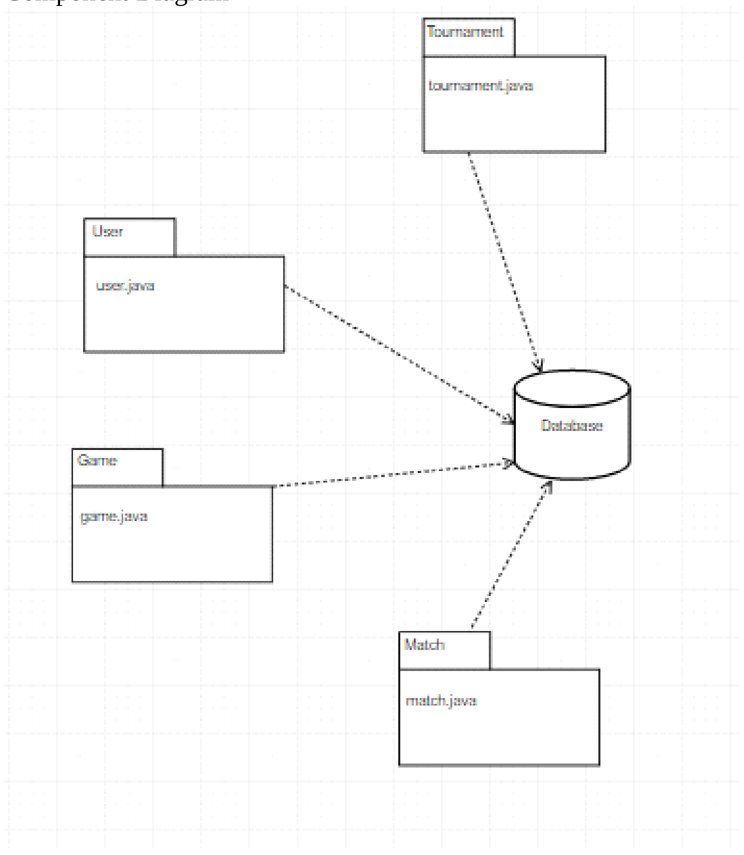
In a typical 3-tier application, the application user's workstation contains the programming that provides the graphical user interface (GUI) and application-specific entry forms or interactive windows. (Some data that is local or unique for the workstation user is also kept on the local hard disk.)
Business logic is located on a local area network (LAN) server or other shared computer. The business logic acts as the server for client requests from workstations. In turn, it determines what data is needed (and where it is located) and acts as a client in relation to a third tier of programming that might be located on a mainframe computer.
The third tier includes the database and a program to manage read and write access to it. While the organization of an application can be more complicated than this, the 3-tier view is a convenient way to think about the parts in a large-scale program.
A 3-tier application uses the client/server computing model. With three tiers or parts, each part can be developed concurrently by different team of programmers coding in different languages from the other tier developers. Because the programming for a tier can be changed or relocated without affecting the other tiers, the 3-tier model makes it easier for an enterprise or software packager to continually evolve an application as new needs and opportunities arise. Existing applications or critical parts can be permanently or temporarily retained and encapsulated within the new tier of which it becomes a component.

3.2 Diagrams

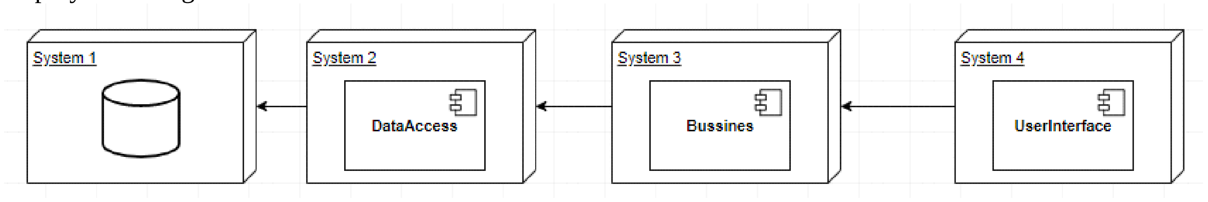
Package diagram


org.UserInterfaceLayer
UserInterface
org.BusinessLayer
Business
dao
org.DataAccessLayer
pingpong
DataAcces

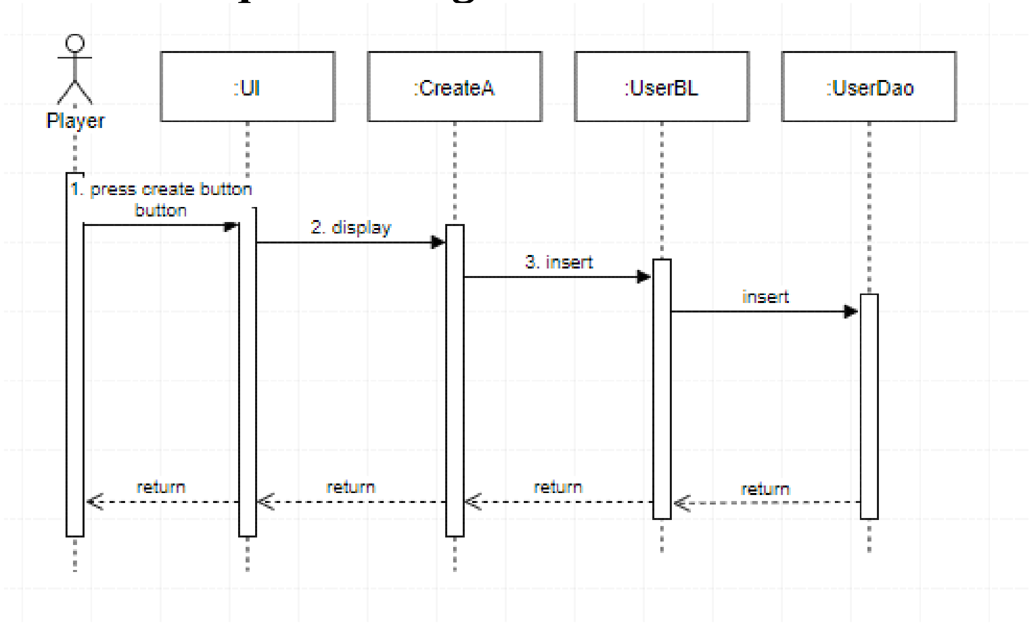
Component Diagram



Deployment Diagram



4. UML Sequence Diagrams



5. Class Design

5.1 Design Patterns Description

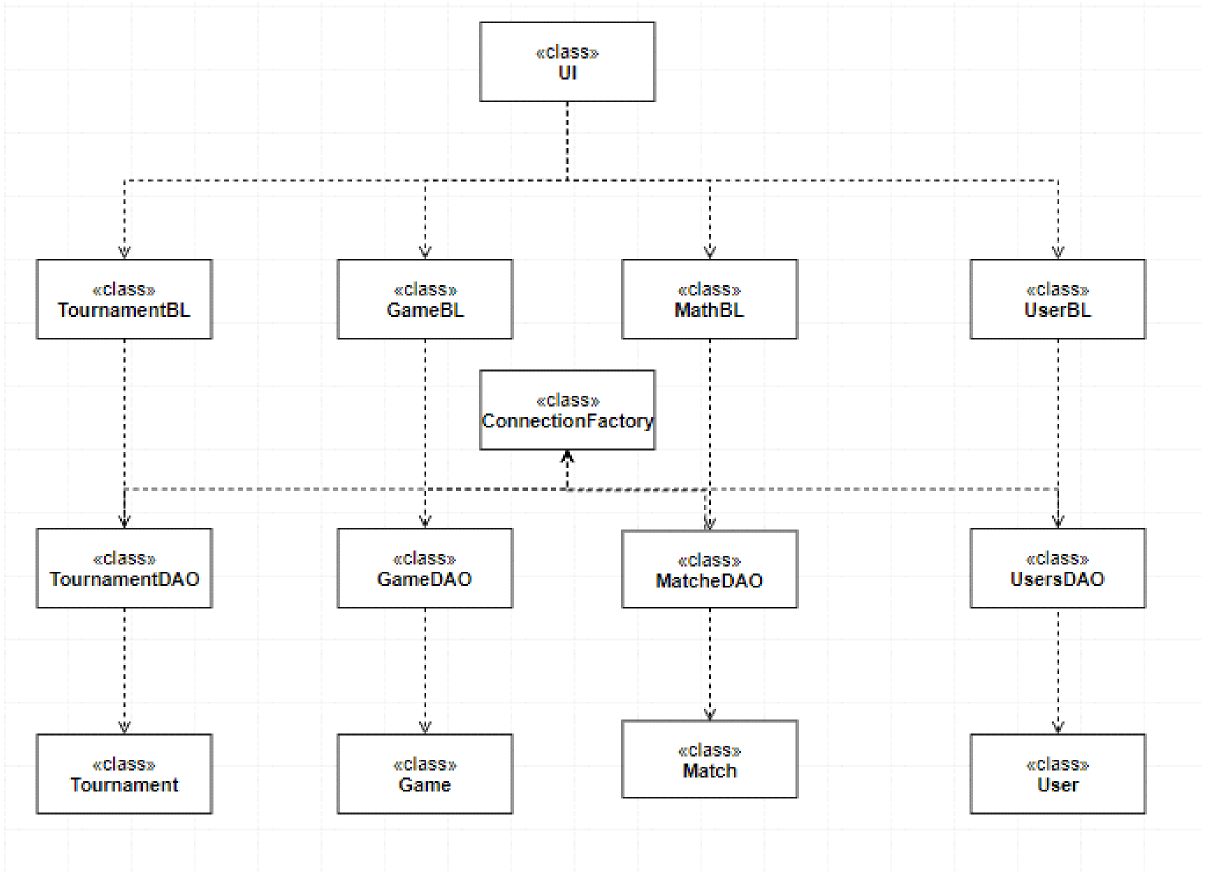
The used design Patterns are Dao, Table Module and 3-tier architecture.

Dao: ccess to data varies depending on the source of the data. Access to persistent storage, such as to a database, varies greatly depending on the type of storage (relational databases, object-oriented databases, flat files, and so forth) and the vendor implementation.

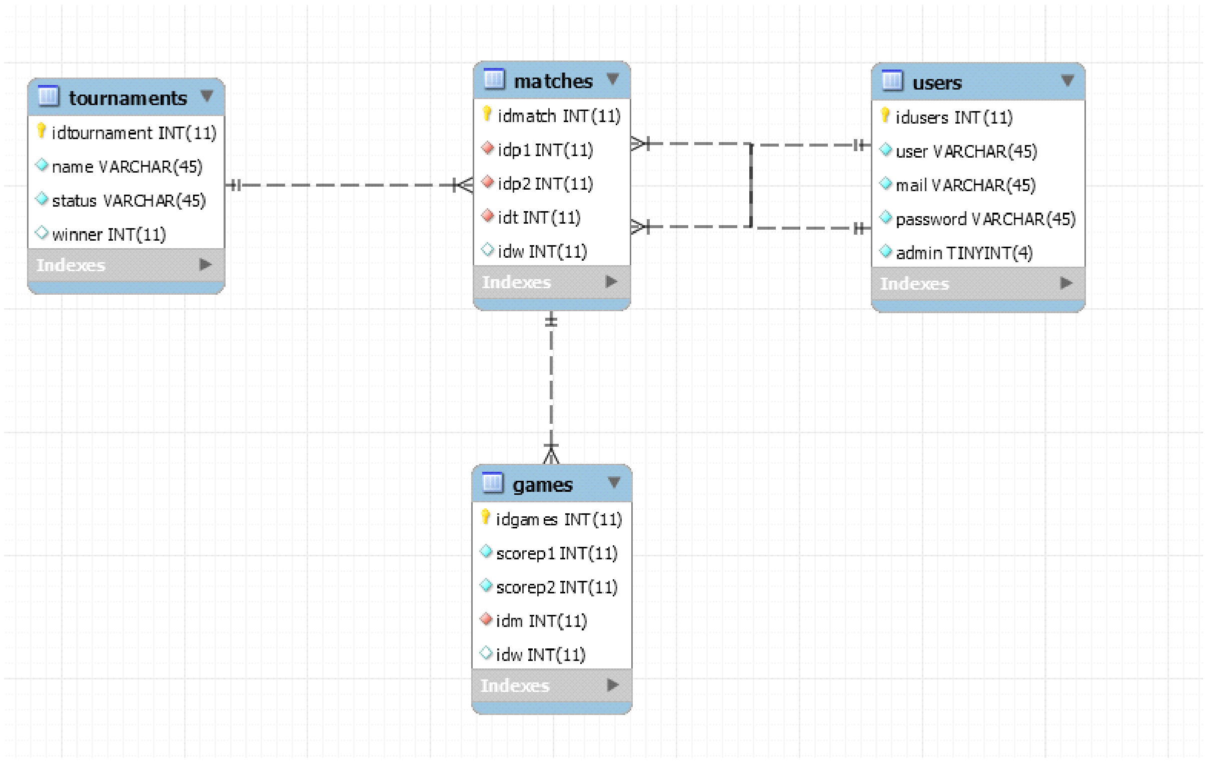
Table Module: A single instance that handles the business logic for all rows in a database table or view.

3-tier: In a typical 3-tier application, the application user's workstation contains the programming that provides the graphical user interface (GUI) and application-specific entry forms or interactive windows. (Some data that is local or unique for the workstation user is also kept on the local hard disk.)

5.2 UML Class Diagram


«class» UI
«class» TournamentBL
«class» GameBL
«class» MathBL
«class» UserBL
«class» ConnectionFactory
«class» TournamentDAO
«class» GameDAO
«class» MatcheDAO
«class» UsersDAO
«class» Tournament
«class» Game
«class» Match
«class» User

6. Data Model



7. System Testing

Junit tests implemented for find by id.

8. Bibliography

http://searchsoftwarequality.techtarget.com/definition/3-tier-application
http://www.oracle.com/technetwork/java/dataaccessobject-138824.html
https://martinfowler.com/eaaCatalog/tableModule.html