**UTCN**

**Computer Science Department**

**Software Design 2018**

**ASSIGNMENT A3**

**====================================================================**

1. **Objective**

The objective of this assignment is to allow students to become familiar with the client-server architectural style, the Observer design pattern and the Bridge design pattern.

1. **Application Description**

Use Java/C# API to design and implement a client-server application for managing online show visualization such as movies, theatre performances and sport events. The application has three types of users: the basic user, the premium user and an administrator.

The basic user can perform the following operations:

* Search show, select a show and view details of a show
* View history of all shows he has seen
* Give a rating to the show
* Add a comment to the show

1p

The premium user can perform the following operations:

* All operations from basic user
* Recommend a show to a friend or a group of friends who also have accounts on the site and are premium users (the recommendation will also appear as a notification on the friends page)
* Add interests in a show he wants to see when it will be uploaded on the site and receive notification from application that the show was uploaded so that he can watch it

1p

The administrator can perform the following operations:

* CRUD on shows (for ex. movie information: name, description, actors, release date, imdb rating).
* CRUD on user accounts.

1p

In addition, when a new show is uploaded on the site and there are users interested in that show the application should inform all the interested users about that show by sending them an update about the show and let them know they can watch it.

1. **Application Constraints**

* The application should be client-server and the data will be stored in a database. 2p
* Use an ORM (hibernate) for database operations 1p
* Use the Observer design pattern for notifying the users when the movies they are interested in have been uploaded. 1p
* Use the bridge design pattern to implement user relationship and show relationship.

(or you can implement a different scenario) 1p

* All the inputs of the application will be validated against invalid data before submitting the data and saving it. 0.5p
* Use Spring (optional and will be graded separately with max 1p)

1. **Requirements**

* Create the analysis and design document (see the template).
* Implement and test the application. 0.5p

1. **Deliverables**

* Analysis and design document.
* Implementation source files.
* SQL script for creating and populating the database with initial values.
* Readme file that describes the installation process of the application and how to use it:
  + how to install your application on a clean computer
  + how to access your application and with what users
  + images with all use cases and their scenarios implemented
* assignments with only 2 or 3 classes in the class diagram will not be accepted!
* assignments with only 2 or 3 tables in the database model diagram will not be accepted! Will be more complex than that!

1. **References**

<http://download.oracle.com/javase/tutorial/networking/sockets/index.html>

<http://docs.oracle.com/javase/tutorial/uiswing/>

<http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

<https://refactoring.guru/design-patterns/bridge>

<https://refactoring.guru/design-patterns/observer>

<https://www.tutorialspoint.com/hibernate/orm_overview.htm>

<https://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-annotations-example/>

<https://spring.io/docs/reference>

1p oficiu

Grade: 0.4 \* Documentation + 0.6 \* Implementation