Faculty of Engineering, Alexandria University
CS 221: PROGRAMMING-2
Dept. of Computer and Systems Engineering

Year: 2nd year
Duration: 180 Minutes
Final Exam: 2018

answer ALL questions:

It is required to build our own version of the IMDb movie portal. IMDb (www.imdb.com) is the world's most popular source for movie and TV. The system stores information about movies and people working in the movie business. A movie can be a single movie (e.g., "gone with the wind"), a series (e.g., "James Bond"). TV-Shows can be a single documentary (e.g., "The Mysteries of the Pyramids") or a season (e.g. "Friends"); in which each season contains several episodes (e.g., "My first Christmas with my Friends"). Series have attributes; which are common to the whole series, such as one or more genres (i.e., types, "comedy", "thriller", etc.), a description, main actors, director(s), writer(s), etc. The single movie in a series can have its special attributes too; such as: publishing date, additional actors, special description, etc. The same applies to the episodes. For all persons in the database, and regardless of their roles, the system keeps their master information; such as name, birthdate, nationality, and place of birth. Note that a person can play several roles in a single movie. "Mel Gibson" was the director and main actor in "Braveheart".

A normal user is allowed to register him-/herself to the portal. Registered users are allowed to comment on movies. S/He can like or dislike a comment. Additionally, s/he can comment on a comment creating a thread of comments.

We want to have a single search textbox (similar to Google), in which the user can type any search term (e.g., "Smith"). The search should be forked into three separate searches: searching for the term in the movies, the persons, and the comments. The search results should be merged in one search result page having the hits from the movies first ("Mr. & Mrs. Smith"), followed by the persons (e.g., the actor "Will Smith"), followed by the comments (e.g., "I would have preferred that Smith would marry Susy at the end of the movie").

Creating a movie can be done by any registered user. The creation screen is complex since all data must be filled out before the objects are created (e.g., descriptions, choosing genres, creating seasons and episodes in case of TV-shows, etc.). A newly created movie is not available at once. It is stays in the editorial area, where users with editorial rights can change the data and finally one of them approves the movie for publishing.

The site should send the user an email saying "Hi <username>, you might consider watching this new movie <movieName>" if a movie is published having the same genre as a movie on which he commented. The user has the option to stop these notifications.

The system administrator manages the master data such as the genres and the persons.

Question 1 (20 grades):
Draw a simple use case diagram illustrating the interaction between the normal users, editorial users, system administrator and your system.

Question 2 (24 grades):
Identify at least:

- 1 fundamental design pattern,
- 1 creational design pattern,
- 2 structural design patterns, and

K.N.

- 2 behavioral design patterns.

Explain briefly (2-3 sentences) the usage of each design pattern in the system.

Question 3 (15 grades):

Draw a detailed UML class diagram for each of the design patterns you applied in Questions 2.

Question 4 (10 grades):

Draw the state diagram for a normal *movie* from creation till its publishing.

Question 5 (21 grades):

Consider the following code snippet.

```
1  public class UserValidator {
2      private Cryptographer cryptographer;
3
4      public boolean checkPassword(String userName, String password) {
5          boolean caseSensitive = true;
6          User user = UserGateway.findByName(userName, caseSensitive);
7          if (user != User.NULL) {
8              String codedPhrase = user.getPhraseEncodedByPassword();
9              String phrase = cryptographer.decrypt(codedPhrase, password);
10             if("Valid Password".equals(phrase)) {
11                 Session.initialize();
12                 return true;
13             }
14         }
15         return false;
16     }
17 }
```

Answer the following questions

I) Identify a **Side-Effect** of this method? How can you avoid it?

II) Identify a design pattern referenced by this code.

III) What is the best way to initialize the private member `cryptographer`?

a) Pass it on in the **constructor**?

b) Pass it in a **setter**?

c) Use the **Singleton** design pattern.

Justify your answer.

IV) Is it better to write `if("Valid Password".equals(phrase))` than `if(phrase.equals("Valid Password"))` or is it the same? State why.

V) In your opinion, should the method `UserGateway.findByName()` throw an exception if the user is not found instead of the current behavior. Justify your answer.

VI) In your opinion, should the method `UserGateway.findByName()` throw an exception if the database throws a `DatabaseConnectionException` or should it deal with it within the `UserGateway.findByName()` method?

VII) What is wrong with the input arguments of the method UserGateway.findByName()?

GOOD LUCK

K.N.