Given: October 15th
Due: November 12th at 11:59pm
Grade: 15% Assignment 1 - The Collectible Trading Card Game

Submit source code and running instructions to EAS[1]. Do it in Java, use eclipse (as in the lab, jdk8, tomcat8, eclipse WTP).

Do your work individually, do not program it with anyone else, and cite in comments any snippets you grab from the web (url, date accessed, what modifications you made). While you may find yourself using a few of these snippets, they should only make up a small portion of your work. If they make up most of your work, you are at risk of being in violation of our code of conduct, and there's just no good reasaon to end up in that sitation.

The code will be marked in eclipse with the included Test Suite, in a lab, with you there to give guidance if there is trouble. You will have five minutes to show the marker that it can run smoothly, and then he will grade you based on the results of the run tests (it gives a number). I will subsequently review all code to make sure that you used the patterns identified for this assignment.

1 Learning Objectives

Learn to Critically Approach a Problem. Getting it done isn't enough. You need to look at every problem, think about its qualities, and come up with an appropriate solution given any constraints. It is possible to get a working solution to this assignment while skipping this step, but achieving this learning objective is incredibly valuable and will serve you well in all future endeavours. Think about what you need to do and why you need to do it!

Learn to Create a WEA from Start to Finish. Many of you have only experience creating snippets of working code. This assignment may give you your first taste of putting it all together, from accessing the database to returning something that can be seen in a web browser. Once you know that you can do this, you should be more confident with all software development, as you know getting this far is definitely achievable (even easy, with practice).

Apply Basic WEA Patterns. Primitive Page Controllers/Transaction Scripts and Row Data Gateways may not be the best thing to use, but they are basic building blocks, and understanding them will help you see their limits, and give context for the more specific patterns we will see in the rest of the course. You will also undoubtedly come across these patterns in practice, and being able to identify them and understand them from a practitioner's perspective will be helpful for managing, maintaining or migrating such code.

2 Description

You will make the beginnings of a Pokemon game. Players will be able to:

- register
- login
- logout
- list players
- challenge players
- list challenges
- accept challenges
- refuse challenges
- upload decks
- view decks
- draw cards
- play pokemon to the bench
- view the board for a game
- view your hand
- list games
- retire from a game

2.1 Implications

This implies a few things:

- you must have more than one player to play a game
- you cannot issue or accept a challenge without a deck
- you can only retire from games you are playing
- you can only view your hand in a game you are still playing
- you cannot have users with the same identification
- you cannot challenge yourself

There may be other implied details. Ask questions and list implications!

[1]`https://fis.encs.concordia.ca/eas/`

2.2 Constraints

- There are no turns
- a deck must have **40** cards
- there is no shuffling! Decks start in the order they are uploaded, and when you play with them, you draw cards (first draw is the first line of the uploaded deck, etc).
- a newly started game will not have a concept of turn and no cards start drawn!
- there are no active pokemon!

2.3 Data

2.3.1 The Deck "Upload" Format

When uploading cards, they are identified by a line of text per card. The format is:
<type> "NAME"
Where type is either e for energy, p for pokemon or t for trainer. e.g.:
e "Fire"
p "Charizard"
t "Misty"

2.3.2 The Board Data

The board consists of a list of players involved, whether they have retired or not, how many cards are in each player's hand, deck, discard pile and the list of either player's benched pokemon.

```
{
        "board": {
                "players": [12, 14],
                "decks": [1, 2],
                "play": {
                        "12": {
                                "status": "playing",
                                "handsize": 5,
                                "decksize": 32,
                                "discardsize": 0,
                                "bench": [7, 6, 2]
                        },
                        "14": {
                                "status": "playing",
                                "handsize": 3,
                                "decksize": 32,
                                "discardsize": 0,
                                "bench": [13, 16, 15, 12, 5]
                        }
                }
        }
}
```

2.3.3 The Deck Data

A deck would be an ordered list of cards:

```
{
        "deck": {
                "id": 1,
                "cards": [
                        {"t": "e", "n": "Fire"},
                        {"t": "e", "n": "Fire"},
                        {"t": "p", "n": "Charizard"},
                        {"t": "e", "n": "Fire"},
                        {"t": "e", "n": "Fire"},
                        {"t": "e", "n": "Fire"},
                        {"t": "p", "n": "Charizard"},
                        {"t": "p", "n": "Meowth"},
                        {"t": "e", "n": "Fire"},
                        {"t": "t", "n": "Misty"},
                        {"t": "t", "n": "Misty"},
                        {"t": "e", "n": "Fire"},
                        ...
                ]
        }
}
```

2.3.4 Viewing the Players

A List of Players, who are actually just Users right now:

```
{
        "players": [
                        {"id": 1, "user": "alice"},
                        {"id": 2, "user": "bob"},
                        {"id": 3, "user": "chuck"},
                        {"id": 4, "user": "darcy"}
        ]
}
```

2.3.5 Viewing the Challenges

Each challenge consists of a challenger, who initiated, and a challengee that they wish to play against. Status starts open (0), but it can be refused by the challengee (1), or the challenger can withdraw by refusing their own challenge (2), or lastly it can be accepted (3).

```
{
        "challenges": [
                        {"id": 1, "challenger": 1, "challengee": 2, "status": 3},
                        {"id": 2, "challenger": 2, "challengee": 1, "status": 2},
                        {"id": 3, "challenger": 3, "challengee": 1, "status": 0},
                        {"id": 4, "challenger": 4, "challengee": 3, "status": 1}
        ]
}
```

2.3.6 General Success and Failure

Many actions just return success of failure. The status is important, as it will be specifically tested against, but don't neglect making useful error messages!

```
{
        "status":"fail",
         "message":"Something went horribly wrong, but make your message more helpful!"
}
```

```
{
        "status":"success",
         "message":"Things went okay! We should probably say specifically what."
}
```

2.4 Calls

Register

path
  /Register

method
  POST

params
  user
  pass

params
  Returns success or failure (2.3.6)

Login

path
: /Login

method
: POST

params
: user
: pass

params
: Returns success or failure (2.3.6)

Logout

path
: /Register

method
: POST

params
: Returns success or failure (2.3.6)

List Players

path
: /ListPlayers

method
: GET

params
: Returns a list of players (2.3.4)

List Challenges

path
: /ListChallenges

method
: GET

params
: Returns a list of players (2.3.5)

Challenge Player

path
: /ChallengePlayer

method
: POST

params
: challengee

params
: Returns success or failure (2.3.6)

Accept Challenge

path
: /AcceptChallenge

method
: POST

params
: challenge

params
: Returns success or failure (2.3.6)

Refuse Challenge

path
: /RefuseChallenge

method
: POST

params
: challenge

params
: Returns success or failure (2.3.6)