Given: November 14th
Due: December 3rd at 11:59pm
Grade: 25% Assignment 2 - The Collectible Trading Card Game

Submit source code (.java .jsp any necessary .jar files) and running instructions to EAS[1]. Do it in Java, use eclipse (as in the lab, jdk8, tomcat8, eclipse WTP). Your work will be evaluated almost entirely on its ability to pass the provided, evolving test suite.

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 reason to end up in that situation.

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. Please be kind and include a database teardown/setup script or a programmatic means to do so for your marker.

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. In this assignment, you must use the full set of mature Architectural patterns, and any related patterns they imply.

- Front Controller
- Dispatcher
- Command
- Template View (or Transform View)
- Domain Object
- Input Mapper
- Output Mapper
- Finder
- Table Data Gateway
- Proxy (Lazy Load)
- Identity Map
- Domain Object Factory
- Unit of Work
- Optimistic Concurrency Management

You will also come across most of 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.

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

2 Description

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

1. register
2. login
3. logout
4. upload decks
5. view decks
6. view deck
7. list players
8. challenge players
9. list challenges
10. withdraw challenges
11. refuse challenges
12. accept challenges
13. list games
14. view board
15. view hand
16. view discard pile
17. bench pokemon
18. evolve pokemon
19. play energy
20. play trainer
21. end turn
22. retire from a game

2.1 Constraints

The game has a few rules to keep in mind:

1. you must have more than one player to play a game
2. you cannot issue or accept a challenge without a deck
3. you can only retire from games you are playing
4. you can only view your hand in a game you are playing
5. you cannot have users with the same identification
6. you cannot challenge yourself
7. both players in a game may look at either discard pile
8. a deck must have **40** cards
9. 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).
10. It is the challenger's turn first.
11. The challenger takes the first turn, starting the game with one card drawn
12. The challengee gets their first card when the challenger ends their first turn
13. there are no active pokemon!
14. one card is drawn at the beginning of each turn after the first (you can think of the start if the first player's turn as happening when a challenge is accepted)
15. you may only play one energy per turn
16. when a pokemon is evolved, it keeps its energy, the basic pokemon does not go into the discard pile, but is still "visible"
17. when a trainer is played, it is moved to the top of the discard pile (nothing else happens)
18. Each hand can have at most 7 cards
19. If a turn ends and that player has more than 7 cards, the oldest card in their hand is moved to the top of the discard pile
20. Optimistic Concurrency Management should cause failure whenever submitted versions don't match for all Use Cases except Retire
21. A player may retire without the need of a version: Optimistic Concurrency Management should just allow retiring whenever, as long as the player is in the game.

22. The same deck may be used in multiple games
23. A player may only have a single open challenge against another player, but may challenge as many different players as they wish
24. A player is able to have more than one active game with a single player

2.2 Data

2.2.1 The Deck "Upload" Format

When uploading cards, they are identified by a line of text per card. The format is:
<type> "NAME"
Additionally, the format for "Stage One" pokemon will be:
<type> "NAME" "BASIC"
Where type is either e for energy, p for pokemon or t for trainer. e.g.:
e "Fire"
p "Charmander"
t "Misty"
p "Charmeleon" "Charmander"

2.2.2 The Board Data

The board consists of a list of players involved, whether they have retired or not, the player whose turn it currently is, how many cards are in each player's hand, deck, discard pile and the list of either player's benched pokemon, including any energy attached and the id of basic pokemon that have evolved into "stage one" pokemon.

```
{
    "game": {
        "id": 1,
        "version": 37,
        "players": [12, 14],
        "current": 14,
        "decks": [1, 2],
        "play": {
            "12": {
                "status": "playing",
                "handsize": 7,
                "decksize": 28,
                "discardsize": 2,
                "bench": [
                    {"id": 7},
                    {"id": 6},
                    {"id": 2}
                ]
            },
            "14": {
                "status": "playing",
                "handsize": 3,
                "decksize": 28,
                "discardsize": 0,
                "bench": [
                    {"id": 13},
                    {"id": 16, "e": [3, 4], "b": 1},
                    {"id": 15, "e": [2]},
                    {"id": 12},
                    {"id": 5}
                ]
            }
        }
    }
}
```

2.2.3 The Deck Data

A list of a player's decks:

```
{
    "decks": [1, 4, 5]
}
```

2.2.4 The Deck Data

A deck would be an ordered list of cards:

```
{
    "cards": [
        {"id": 1, "t": "e", "n": "Fire"},
        {"id": 2, "t": "e", "n": "Fire"},
        {"id": 4, "t": "p", "n": "Charmander"},
        {"id": 5, "t": "e", "n": "Fire"},
        {"id": 6, "t": "e", "n": "Fire"},
        {"id": 7, "t": "e", "n": "Fire"},
        {"id": 8, "t": "p", "n": "Charmeleon", "b": "Charmander"},
        {"id": 9, "t": "p", "n": "Meowth"},
        {"id": 10, "t": "e", "n": "Fire"},
        {"id": 11, "t": "t", "n": "Misty"},
        ...
    ]
}
```

2.2.5 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"}
    ]
}
```

2.2.6 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). The deck used by the challenger to initiate the challenge is also shown.

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

2.2.7 General Success and Failure

Many actions just return success or 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."
}
```

2.2.8 Games

When listing current games!

```
{
    "games": [
        {"id": 1, "version": 1, "players": [1, 2]},
        {"id": 2, "version": 1, "players": [2, 1]},
        {"id": 3, "version": 1, "players": [2, 3]},
        {"id": 4, "version": 1, "players": [3, 4]},
    ]
}
```

2.2.9 Hand

When viewing your hand!

```
{
    "hand": [13, 16, 15, 12, 5]
}
```

2.2.10 Hand

When viewing a discard pile!

```
{
    "discard": [13, 16, 15, 12, 5]
}
```

2.3 Calls

2.3.1 Register

path
/Poke/Player/Register

method
POST

params
user
pass

returns
Returns success or failure (2.2.7)

2.3.2 Login

path
/Poke/Player/Login

method
POST

params
user
pass

returns
Returns success or failure (2.2.7)

2.3.3 Logout

path
: /Poke/Player/Logout

method
: POST

returns
: Returns success or failure (2.2.7)

2.3.4 Upload Deck

path
: /Poke/Deck

method
: POST

params
: deck - the String of cards, one per line in the format from section 2.2.1

returns
: Returns success or failure (2.2.7)

2.3.5 View Decks

We only want to show decks uploaded by the current player.

path
: /Poke/Deck

method
: GET

returns
: Returns deck (2.2.4) or failure (2.2.7)

2.3.6 View Deck

This will show the deck specified. Any player may view any deck... because.

path
: /Poke/Deck/(\d+)

method
: GET

in-url-param
: The id of the deck to be viewed

returns
: Returns deck (2.2.4) or failure (2.2.7)

2.3.7 List Players

path
: /Poke/Player

method
: GET

returns
: Returns a list of players (2.2.5)

2.3.8 Challenge Player

path
: /Poke/Player/(\d+)/Challenge

method
: POST

in-url-param
: The id of the player challenged

params
: deck - the deck to be used by the challenger if the challenge is accepted

returns
: Returns success or failure (2.2.7)

2.3.9 List Challenges

List Challanges that the current player is involved in.

path
: /Poke/Challenge

method
: GET

returns
: Returns a list of players (2.2.6)

2.3.10 Withdraw Challenge

path
: /Poke/Challenge/(\d+)/Withdraw

method
: POST

in-url-param
: The id of the challenge

params
: version - the version of the challenge

returns
: Returns success or failure (2.2.7)

2.3.11 Refuse Challenge

path
: /Poke/Challenge/(\d+)/Refuse

method
: POST

in-url-param
: The id of the challenge

params
: version - the version of the challenge

returns
: Returns success or failure (2.2.7)

2.3.12 Accept Challenge

path
: /Poke/Challenge/(\d+)/Accept

method
: POST

in-url-param
: The id of the challenge

params
: version - the version of the challenge
: deck - the deck to be used by the challengee

returns
: Returns success or failure (2.2.7)

2.3.13 List Games

List games that the current player is involved in.

path
: /Poke/Game

method
: GET

returns
: Returns games (2.2.8) or failure (2.2.7)

2.3.14 View Board

path
: /Poke/Game/(\d+)

method
: GET

in-url-param
: The id of the game

returns
: Returns board (2.2.2) or failure (2.2.7)

2.3.15 View Hand

path
: /Poke/Game/(\d+)/Hand

method
: GET

in-url-param
: The id of the game

returns
: Returns hand (2.2.10) or failure (2.2.7)

2.3.16 View Discard Pile

path
/Poke/Game/(\d+)/Player/(\d+)/Discard

method
GET

in-url-param
The id of the game
The id of the player whose discard is to be returned

returns
Returns hand (2.2.10) or failure (2.2.7)

2.3.17 Play Pokemon to Bench

Play a "basic" pokemon (those without the extra field) to the bench.

path
/Poke/Game/(\d+)/Hand/(\d+)/Play

method
POST

in-url-param
The id of the game
The pokemon to play from the hand

params
version - the version of the game

returns
Returns success or failure (2.2.7)

2.3.18 Evolve Pokemon

The current player evolves a "basic" pokemon on the bench with a "stage one" pokemon from their hand. The stageone pokemon is removed from the hand and replaces the benched basic pokemon, keeping any attached energy.

path
/Poke/Game/(\d+)/Hand/(\d+)/Play

method
POST

in-url-param
The id of the game
The id of the stage one pokemon to be played

params
basic - the id of the basic pokemon on the bench to be evolved
version - the version of the game

returns
Returns success or failure (2.2.7)

2.3.19 Play Energy

The current player attaches an energy onto a benched pokemon. The energy card is removed from the player's hand. Can only be done once per turn.

path
/Poke/Game/(\d+)/Hand/(\d+)/Play

method
POST

in-url-param
: The id of the game
The id of the energy to be played

params
: pokemon - the id of the pokemon on the bench to attach the energy to
version - the version of the game

returns
: Returns success or failure (2.2.7)

2.3.20 Play Trainer

The current player plays an trainer. The trainer card is removed from the player's hand and placed into their discard. Can be done any number of times per turn.

path
: /Poke/Game/(\d+)/Hand/(\d+)/Play

method
: POST

in-url-param
: The id of the game
The id of the trainer to be played

params
: version - the version of the game

returns
: Returns success or failure (2.2.7)

2.3.21 End Turn

The current player's turn ends, player status changes accordingly, and the next player draws a card.

path
: /Poke/Game/(\d+)/EndTurn

method
: POST

in-url-param
: The id of the game

params
: version - the version of the game

returns
: Returns success or failure (2.2.7)

2.3.22 Retire

path
: /Poke/Game/(\d+)/Retire

method
: POST

in-url-param
: The id of the game

returns
: Returns success or failure (2.2.7)