Object-Oriented Programming & Design - Ex4 (very last)

Abstract
This document presents the final (last) assignment for the OOP course (CS.Ariel 2021), In this assignment, you will be asked to "put into practice" the main tools covered along the course, in particular, you are expected to design a "Pokemon game" in which given a weighted graph, a set of "Agents" should be located on it so they could "catch" as many "Pokemons" as possible. The pokemons are located on the graph's (directed) edges, therefore, the agent needs to take (aka walk) the proper edge to "grab" the pokemon (see below for more info). Your goal is to maximize the overall sum of weights of the "grabbed" pokemons (while not exceeding the maximum amount of server calls allowed in a second - 10 max)

The Pokemon game

Here is a list of directions regarding the "Pokemon game":

1. The game is being played on a "server" that is given to you, you should design and implement the client-side (only).
2. The server is a simple .jar file that can be run on any java machine (JDK 11 or above) in a command line, e.g., <java -jar Ex4_Server_v0.0.jar 0> (where the "0" parameter is a case between [0-15]).
3. After the server is running a client can connect to it (play with it) - you have two platforms to choose from: java or python (there is NO need to implement in both).
   The java example is a very simple cmd example while the python includes a basic GUI (neither have any algorithms).

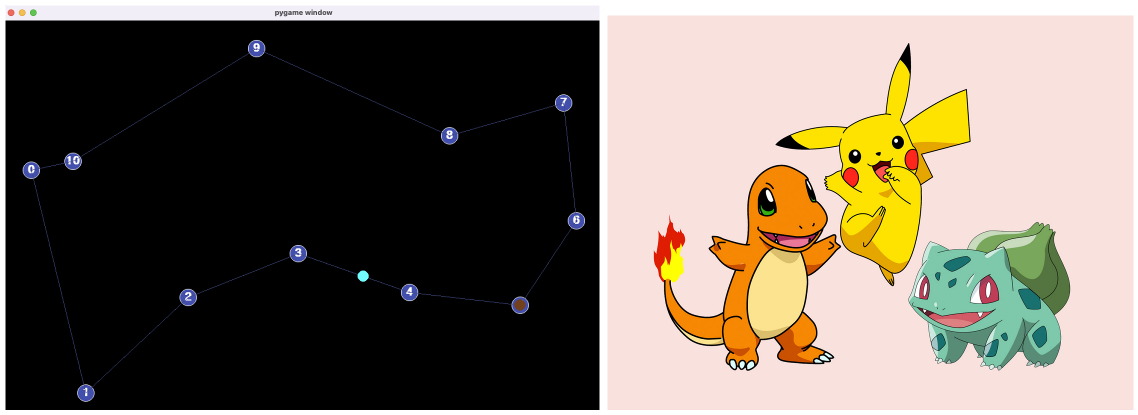
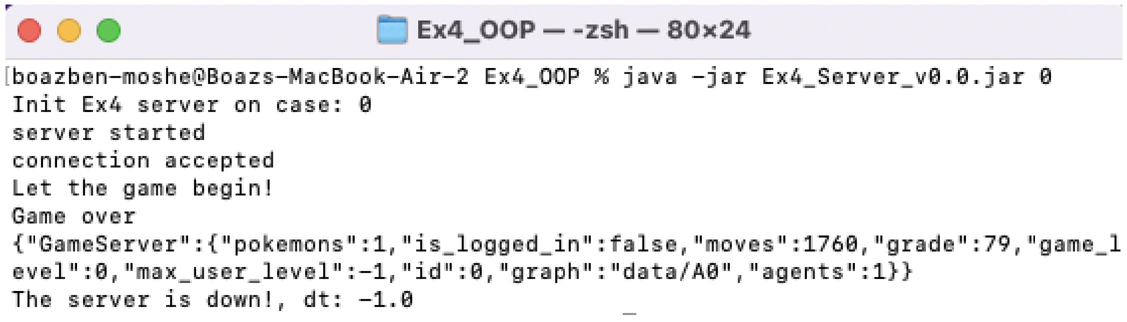


Figure1 (left) case 0, the game as played using the python (pygame) example, the "agent" is marked in brown, while the Pokemon is marked in a light blue.

About the GUI:
 Clearly, there is a long way "to go" to make the UI on the left somewhat resemble the one on the right - yet keep in mind, this assignment is mainly regarding properly designing and implementing a software package - the GUI is (always) secondary to the algorithms.

However, it is important to create a clear scalable gui with a resizable window.
Make sure overall points, moves counter, and time to end in seconds are presented, as well as a “stop” button to gracefully stop the game at any time point.
Pokemons lying on downward edges should be differ from ones on upward edges.
It is recommended (for algo debug) to draw the nodes and agents ids as well as the pokemons values.

4. After the game ends (each game has a fixed time - commonly 30-120 seconds) the results are printed - by the server.



Ex4_OOP — -zsh — 80×24

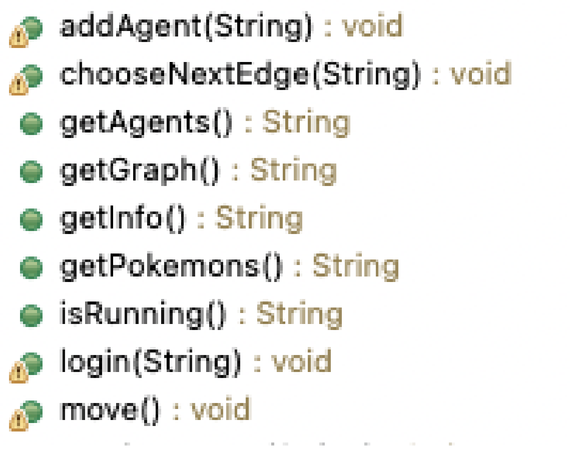
```
[boazben-moshe@Boazs-MacBook-Air-2 Ex4_OOP % java -jar Ex4_Server_v0.0.jar 0
 Init Ex4 server on case: 0
 server started
 connection accepted
 Let the game begin!
 Game over
 {"GameServer":{"pokemons":1,"is_logged_in":false,"moves":1760,"grade":79,"game_l
 evel":0,"max_user_level":-1,"id":0,"graph":"data/A0","agents":1}}
 The server is down!, dt: -1.0
```

Figure2: the server cmd - the results are printed as a json string below the “Game over”

5. In this assignment, we are mainly interested in maximizing the overall score - which is denoted as “grade” - the sum of all the pokemon weight as caught by all the “Agents”. E.g., in Figure 2, the grade is 79, with 1760 moves (30 seconds game).
6. The objective of this assignment is to maximize the grade - but without exceeding the maximal 10 calls to move per second (on average, that said, the above result is not valid as the number of “moves” is strictly above 300 = 30*10).
7. The client (both in java and python) has the following api (all json based):
    a. Init the server with a case [0-15] from a command line.
    b. Get the underlining weighted (directed) graph - you can assume it is strongly connected (getGraph).
    c. Get the list of Pokemons (getPokemons), e.g., {"Pokemons":[{"Pokemon":{"value":5.0,"type":-1,"pos":"35.197656770719604,32.10191878639921,0.0"}}]} where value is the weight (grade), type is positive if the edge_dest > edge_src (else negative), and pos is the 2D position of it (here you need to find the edge - by the position & type).
    d. Get a list of all the Agents (in case an agent is on a node its “dest” is -1).
    e. Locate each Agent on a node (addAgent) - before the game starts.
    f. Start a game - each game has a fixed time - mostly 30-120 seconds.
    g. Get the remaining time (in mili seconds) for the game to be played
    h. Direct each agent to the next destination (chooseNextEdge) - this can be done only when the agent is on a node (not on an edge) - this is the main api for the algorithm.
    i. Move the agents: this is the main method that “plays the game” - in order

for an Agent to grab a pokemon, the agent needs to be on the same (directed edge) & the server should be called (move) when the Agent is “close-enough” to the pokemon - Note: the exact distance is not given.

j. Get the game info (getInfo): returns the grade, and the number of moves of the current game. This data is printed at the end of each game.



```
addAgent(String) : void
chooseNextEdge(String) : void
getAgents() : String
getGraph() : String
getInfo() : String
getPokemons() : String
isRunning() : String
login(String) : void
move() : void
```

Figure3: the main (java) api - the python is the same.

Working Tasks:

Sage 1 - get it running + github the project.
Stage 2 - design the general algorithm (write it down properly in your github docs).
Stage 3 - implement the simplest version of a working code (this should include a debuggable GUI)
Stage 4 - Improve your code and start uploading your results to the Google form.
Stage 5 - Make sure you have all the needed documents including readme, algorithm definition, class diagram, “how to run”, results, tests (unitest / junit), and some images (and a short clip) as a wiki (github) project. Make sure there is a working release available for download, and a short clip
Stage 6 - report your results here, make sure you report all the 16 cases [0-15].
Stage 7 - upload your github link here.

Concluding remarks:

- The assignment should be submitted by Jan 9th to a form (here.).
- Your code will be tested for plagiarism!!
- You are welcome to use your previous graph implementation (i.e., Ex2, Ex3).

It is a lot of work - but have fun designing, refactoring and implementing a “game to remember”.

Why is no one having a good time?
I specifically requested it.

Q & A

1. Q:My jetbrains idea implies I have an updated java jdk but the terminal can’t run the jar file.
   A:The problem may be with your ambiguous java versions. Your system has a “JAVA HOME” path variable with an older version of java, while your idea knows the updated one. To ensure this is your situation, try running the jar from the jetbrains terminal. If this works, you may want to update your java home path variable to the new version on your computer. Either way, you can leave everything as is and run from jetbrains terminal and everything should be just fine.
2. Q: We’ve already got a decent algorithm which outputs nice results in terms of scores, but our code design doesn’t follow some important fundamentals of OOP such as MVC, SOLID, design patterns ect. Are those issues critical for the assignment?
   A: Yes.