Pokemon game Graph.

In the Pokemon game you have to manage a group of agents whose goal is to catch as many Pokemon as possible before time runs out. The more Pokemon you catch, the more points you earn. The game is played on a game board

Varies, depending on the stage you choose to play (there are 24 different stages, with increasing difficulty.) The movement on the board is on a weighted directed graph, on which Pokemon are randomly scattered, the same

Agents can catch by moving close to them on the graph.

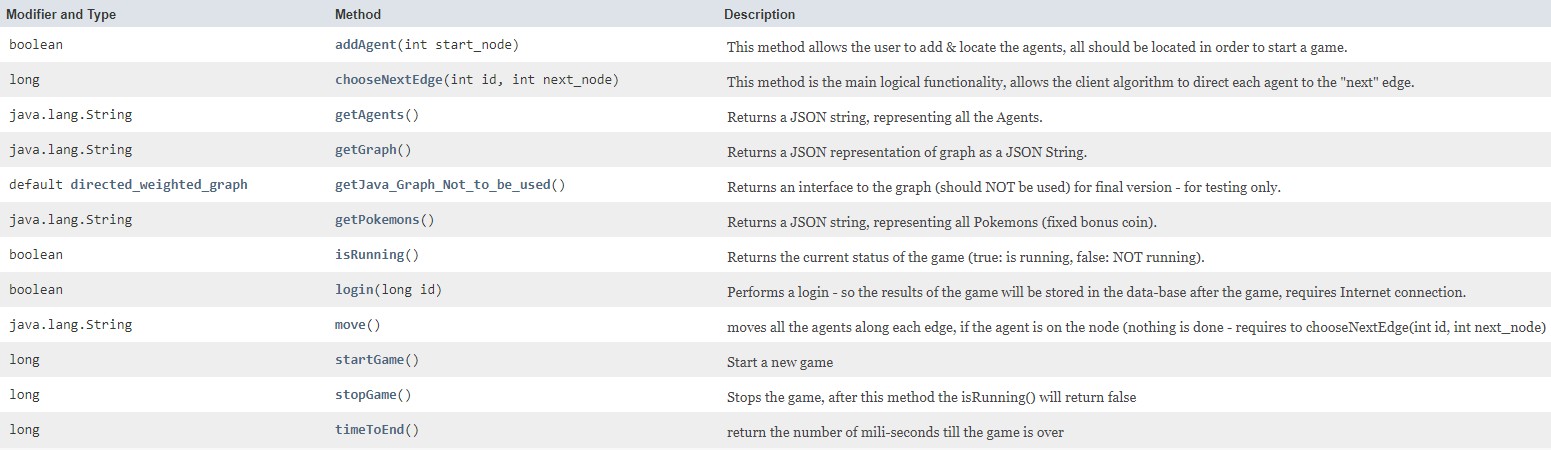
Game management will be conducted in the following steps:

1. Stage selection.

2. Location of agents to their origin points.

3. Start of the game.

4. Continuous management of targeting agents until the game is over.

The game will be managed by a server, with which you can communicate via the following API:

Step details:

1. **Step Selection:** To get a reference to the game server one has to run the following method along with the step of the game you want to play:

game\_service game = Game\_Server\_Ex2.getServer(level\_number);

Following this action, the server builds the game board within it according to the requested stage and returns a reference to the server, a reference on which we will apply any method related to communication with the server. The you can get information about the board, information like how many Pokemon and agents there are at that stage. It is also possible to know what the graph of the board looks like (information about the sides and vertices), and to know where the Pokemon are located, what type they are and how much they are worth. To get this information one has to use the different get-methods offered by the server API as detailed in the image above.

all the information that comes from the server comes in JSON format. Note that when there are two vertices with id 4 and 9 for example, which are connected in a two-way way, that is, there are two sides between them, from 4 to 9 and vice versa, and if there is a Pokemon between these vertices, the Pokemon belongs to only one Of them. This means that crossing the other side will not allow the Pokemon to be grasped, but only crossing the correct side between the two. How do you know which side of the Pokemon is associated with? By the value return type- of the same Pokemon. type with a value of "1" means that the Pokemon is on the side from the lower vertex (for you the id value of the vertex) towards the higher vertex. In our example, the Pokemon will be on the side from 4 to 9 and if in the type of the Pokemon the value was "-1" then the Pokemon is placed on the rib protruding from the higher vertex, towards the lower vertex. That is - the side from 9 to 4.

1. **Positioning agents to their starting points**: Thanks to the information you have from the previous step, you can make a strategic decision on where to place your agents. The positioning is done by using the method addAgent(int StartNode). Note that this method should be read as the number of agents that should be active at the stage you have chosen to play. Call for placement of each agent.
2. **Start of the game**: The start of the game is done by using the method startGame(). From that moment on the clock ticks backwards. Once the time runs out, the game will end and no more points can be scored. You can check if the game is still active or check how much time is left by using the appropriate functions offered by the API.
3. **Ongoing management of agents until the game is over:** Each agent has current location and destination values ​​that the agent plans to reach, defined as src and dest. That will contain id values ​​of vertices on the graph. Notice that dest can only be a vertex which his neighbor vertex is src. These values ​​can be seen, as are additional agent settings (such as speed) by calling the method getAgents() that will provide the relevant information in a JSON format.

In order to update the value of dest of the agent. The method you must run is chooseNextEdge(int agentID, int nodeID).

The game server can be told to move the agents on the graph by using the method move(). Once this method is activated, the server will move each of the agents in the game, on the graph, from where the agent is currently located, to the target vertex set for it at the value dest, according to the speed set for that agent and according to the last time the method move() was activated. For example, if an agent moves at a speed of 1 unit of distance per second, and 3 seconds have passed since the last time you requested the server to move the agent, then now the request will cause the agent to move a distance of 3 units of distance. You must update the dest according to the traffic strategy you choose. The agent's movement will be according to the dest- defined for him during the game. Note that while an agent is on the edge it is not possible to update its dest, as it is currently on its way to to another dest that was set by earlier. You can update the agent's dest only with if the current dest is 1- (which means the agent is currently missing a target for the following movements).   
And this will happen when the agent has finished reaching some destination vertex. Be careful not to accidentally skip Pokemon and thus miss them (to capture a Pokemon you must go close to it) an action that can happen if you use the method move() with too large time intervals, which will result in too int the steps being to large in the agent movements on the edge. After Pokemon is caught - another will appear in its place in a random position on the graph.

pay attention! All the methods described here, is not a graphical interface at all. The construction of the graph on the server and the movement of the agents on it is purely logical. Therefore, apart from the logical management of the game, you must take care to present the entire occurrence of the course of the game visually. Starting with and during the game, so you can watch your agents' progress on the graph and their Pokemon perception. And of course answer all The requirements specified in the official assignment document.

After your code and algorithm work - add a login execution code with the ID. After step number 1 in order for the results of your runs to be saved. Note that the score is affected by the amount of Pokemon you catch (the more you catch the higher the score) and the number of times you run the method move() The more you call the move() method, the lower your score is going to be.   
Tip: Don't skimp on the move () method calls for starters. After your game runs and its stable you will start thinking about how you can reduce amount of calls you used the move () method so you can improve you score.

Get help (but not really required) from the sample classes you received with the assignment. They demonstrate an active course of play and form of communication with the server. They also have an example of using a graphical interface and display of the game. All this code is to serve as an example and give help and guidance, but there is no obligation to use it either See it as a binding guideline, one way or another, for the form of realization of the task.

Good Luck!