COMP3021: Java Programming (Fall 2016)

Programming Assignment #2: GUI Pokémon

Contents

Background

Pokémon Go is a free-to-play, location-based augmented reality game developed by Niantic for iOS, Android, and Apple Watch devices. It was initially released in July 2016 and got extremely popular worldwide. In this game, players can acquire random numbers of Poke Balls at different Supply Stations. When wild Pokémons hidden in the surrounding environment are spotted, the players can use Poke Balls to catch these Pokémons. The following shows some screenshots of this game.

In this assignment, you are required to enhance your text-based Pokémon Go implemented in Assignment 1 with GUI display.

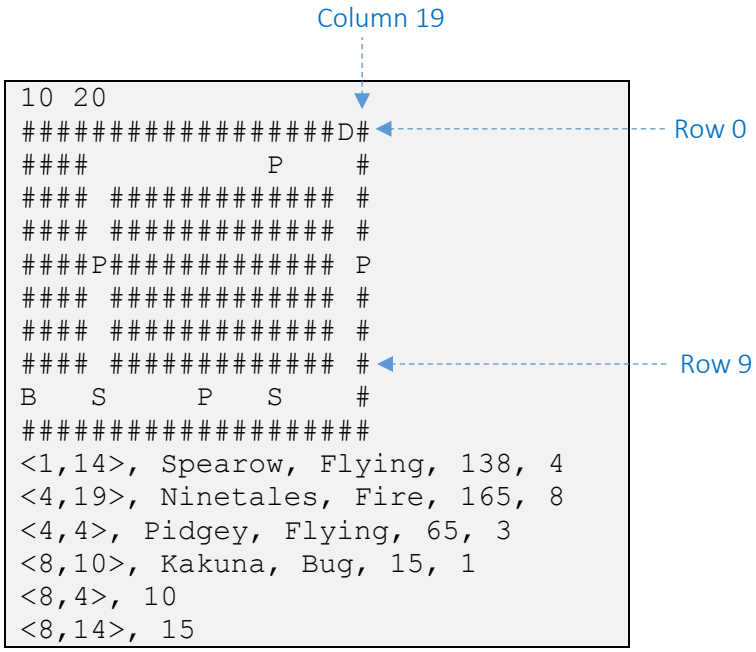
GUI-Based Pokémon

You are required to enhance your text based Pokémon with GUI display in this assignment. The input file is the same as Assignment 1. There is no output file for this assignment. All the information is displayed with GUI using JavaFX, which is updated simultaneously with the movement of the player.

Descriptions of the input file
Describes the map
Describes the positions and properties of all Pokémons.
Describes the positions and configurations of all supply stations.

Specifications of the Input File

The input file, describes the map, the properties of Pokémons and Supply Stations. We are going to describe these aspects one by one as the following:



The screenshot above shows an example of the input file. The first line contains two numbers **M** and **N**, which define the number of rows (i.e. 10 rows) and the number of columns (i.e. 20 columns), respectively. Both M and N are positive integers greater than zero. The following M lines define the map. Each line contains exactly N characters. Each cell <R,C> in the map is indexed by its row R and column C (Both R and C are indexed from 0). There are six different types of characters. "#", " ", "B", "D", "S", "P". The meanings of these characters are shown as followings:

Character	Meaning	Can Pass or Not?
"#"	Wall cell, which any player cannot pass.	No
" "	Empty cell	Yes
"B"	The starting point of the player	Yes
"D"	The destination point of the player	End of the Game
"S"	Supply Station.	Yes
"P"	Pokémon	Yes

This file then describes the properties of all Pokémons. Suppose there are N_P Pokémons in the map (N_P = 4 in the map shown above), there will be N_P lines after the map. Each line describes the position of the Pokémon (marked as <row number, column number>), the name of it, the type of species (Water, Bug, Flying, Ground, Poison, …), the combat power of the Pokémon and the number of Poke Balls needed to catch it. These properties are separated by commas, and there may be extra spaces between them.

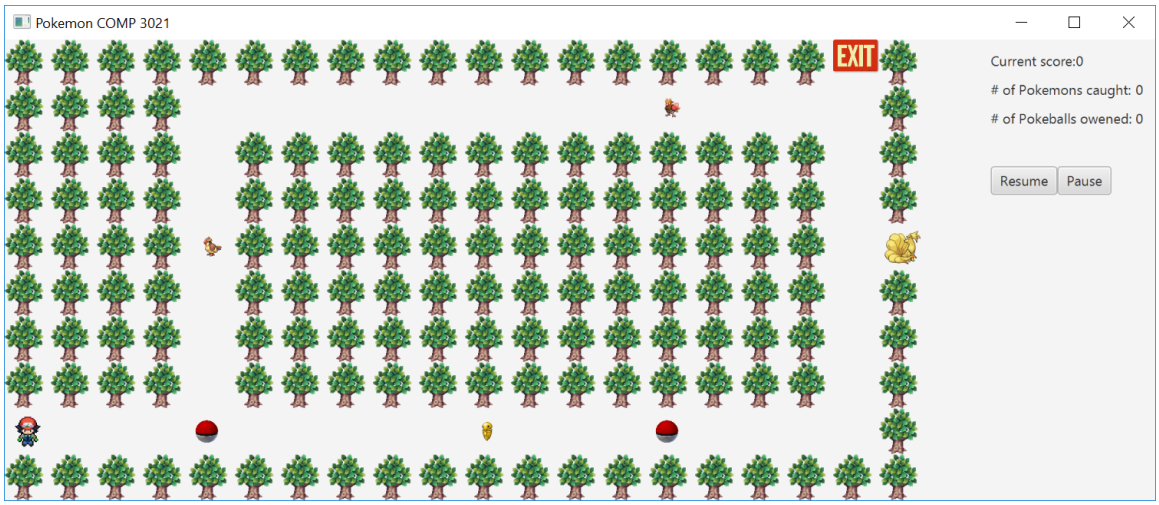
$$< Row\ Number, Column\ Number >, Name, Type, Combat\ Power, Number\ of\ Required\ Balls$$

Finally, the file describes the information of supply stations. Suppose there are N_S supply stations in the map (N_S =2 in the map shown above), there will be N_S lines after the definition of Pokémons. Each line describes the position of the supply station and how many Poke Balls you can obtain in this station. They are separated by commas and may contain spaces between them.

$$< Row\ Number, Column\ Number >, Number\ of\ Provided\ Balls$$

The Expected GUI Display

You need to read the information of the whole map (this should be already implemented for Assignment 1) and display it with GUI using JavaFx. The initial status of the display is the same as the description in the input file. For example, for the input file shown above, the initial interface is shown as the follow picture. The GUI is mainly composed of two parts. The left part shows the map and the right part shows the status of the player. Besides, there are two buttons “RESUME” and “PAUSE” in the right panel.



For the map displayed in the left, Pokémons and stations are displayed in the position as descripted in the map. The player is displayed at the starting point of the map. In the destination point, there displays an exit mark. Walls are displayed with tree images and there are no images displayed for all the empty cells. All the icons can be found in our provided package.

For the information panel displayed in the right, you need to show the current score for the player. The scoring function is calculated the same as Assignment 1 as the following:

$$scoring\ function = < NB + 5 * NP + 10 * NS + MCP - Steps >$$

The table below shows the meanings of these symbol:

NB	*<Number of Poke balls of the player when he arrives at the destination>*
NP	*<Number of Pokémons that the player has caught>*
NS	*<Number of distinct types of all Pokémons that the player has caught >*
MCP	*<The maximum combat power of all Pokémons that the player has caught >*
Steps	*<The steps of the move from the starting point to the destination>*

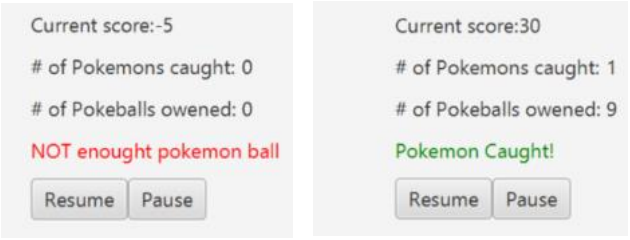
You also need to display the number of the Pokémons (NP) that the player has caught, and the poke balls (NB) the player owns right now.

There are two buttons displayed bellow this information. The functions of these two buttons will be described in the following section.

Make the Player and All Pokémons Mobilizable

All the Pokémons and the player should be mobilizable in this assignment, and the rules are specified as the following:

1. The movements of the player should be controlled by keyboard, it can move "up"," down"," left" and "right" by pressing the key of "up", "down", "left" and "right". However, he cannot move if the next cell is ***Wall*** (displayed as tree in the GUI. This part is covered by Lab9).
2. All the Pokémons can move by themselves automatically. They can move "up", "down", "left" and "right" randomly to a cell C, if and only if cell C is ***Empty*** or is the ***Player***. Two Pokémons cannot appear at the same cell simultaneously.
3. If the player meet any Pokémon, which means the player and the Pokémon appear at the same cell at the same time, the event of catching will be triggered.
   - If the player has enough poke balls, the Pokémon will be caught, and it will disappear forever.
   - If the player does not have enough poke balls, the Pokémon will not be caught. Then it will disappear first and will reappear at any of the ***Empty*** cells randomly later. The time for reappearance is determined randomly within the range of 3 seconds to 5 seconds.
   - You need to show the information of the result in the information panel as the followings to indicate whether the Pokémon has been caught or not.



4. All the Stations cannot move. However, they will disappear when the player reaches them and obtains the poke balls they provided. They will reappear randomly in any empty cell at a certain time later. The time required for reappearance is determined by randomness within the range

from 5 seconds to 10 seconds. If the station reappears, the player can obtain the poke balls again.

5. The function of the "PAUSE" is to pause the game. When this button is clicked, all the movements, including the player and all the Pokémons are disabled. The player will not respond to the keyboard and all the Pokémons are still. The function of "RESUME" is to continue the game.
6. The information panel needs to be updated at each movement of the player. When the player reaches the "***EXIT***", the game will be terminated.

Implementation Notes

- Each Pokémon and station should be handled by its own thread. To do this, the Pokémon class and the station class need to extend class ***java.lang.Thread***. However, since Pokémon is already extending class Cell and Java class can only extend at most one class. Therefore, you need to make class Cell extends ***java.lang.Thread***.
- To make sure no two Pokémons appear at the same cell at the same time, you are required to use synchronization between multiple threads. You need to protect the code region that access and or modified the map.
- Before you start the thread for each Pokémon and station, you have to set the daemon as true, for example, pokemon.setDaemon(true). So that when you close the application, all threads will be interrupted.
- All the icons are provided which are named with the ID of the Pokémons. We also provide a class PokemonList.java. You can use PokemonList. getIdOfFromName(String name) to retrieve the id of a Pokémon based on its name. For walls, exit and station you can use the following images:

```
String wall = new File("icons/tree.png").toURI().toString();
String exit = new File("icons/exit.png").toURI().toString();
String station = new File("icons/ball_ani.gif").toURI().toString();
```

- To add delays you can use ***java.util.Random*** to generate a number random between two values like this:

```
Random r = new Random();
int Low = 10;
int High = 100;
int delay = r.nextInt(High-Low) + Low;
```

Then you can use Thread.sleep(milliseconds)

```
try {
    Thread.sleep(delay);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

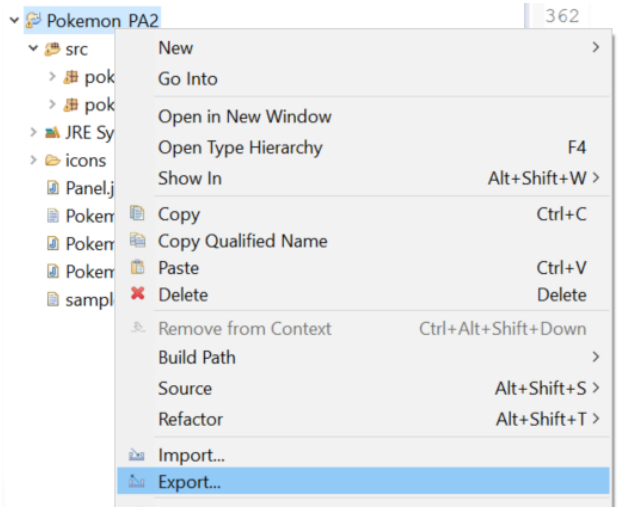
Marking Schemes

- *Please observe the submission deadline, later submission will not be accepted or evaluated.*
- *This assignment mush be finished individually; plagiarism is not allowed. Please refer to the plagiarism policy on the web page(*https://course.cse.ust.hk/comp3021/index.html#policy*)*
- *We will use other input files to start the game.*

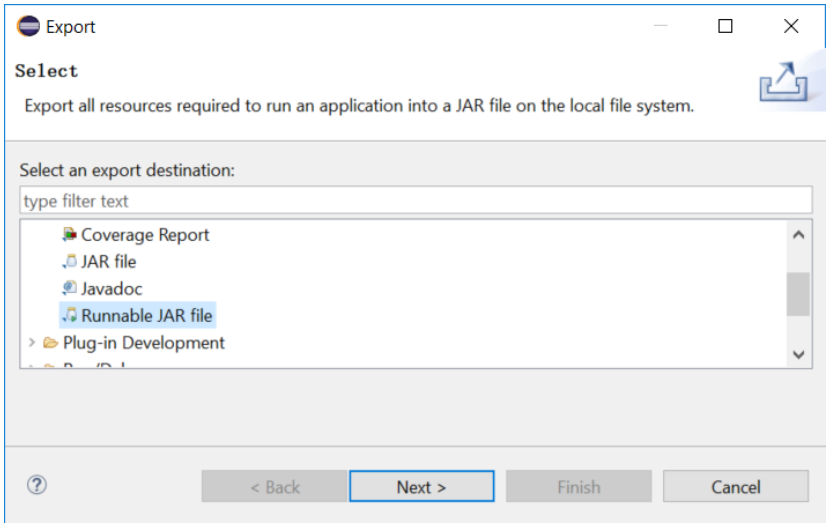
Description	Percentage
The correct format of the submission with the name <Your student Name>.zip, which contains both your source files and a runnable jar. The runnable jar should read the file from "sampleIn.txt" in the same folder and can be successfully run without any exceptions.	15%
The map is correctly created based on the input file (walls, empty spaces and exit).	10%
The game panel can be correctly updated	10%
The player can successfully respond to keyboard and make valid moves. (cannot walk through walls).	15%
All the Pokémons can make valid moves automatically (cannot walk through walls).	10%
If the Pokémon has not been caught, it will disappear and reappear at a random empty cell 3 seconds to 5 seconds later randomly.	10%
Exactly one Pokémon can occupy a cell at a given time, which is implemented using Synchronization.	10%
Stations can reappear and regain their balls between 5 to 10 seconds randomly after the station was visited.	5%
The game can be paused and resumed.	10%
The game terminates when the player reaches the exit.	5%

Submission Details

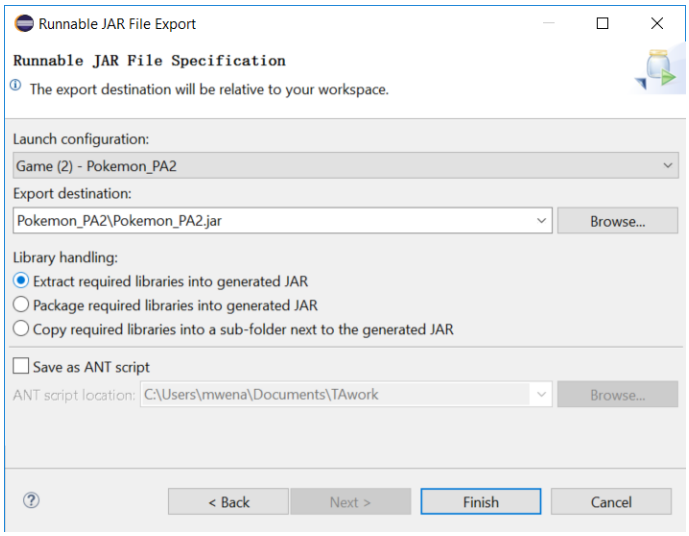
You need to submit both your source codes and a runnable jar for this assignment.
The following describe the steps of extracting a runnable jar:



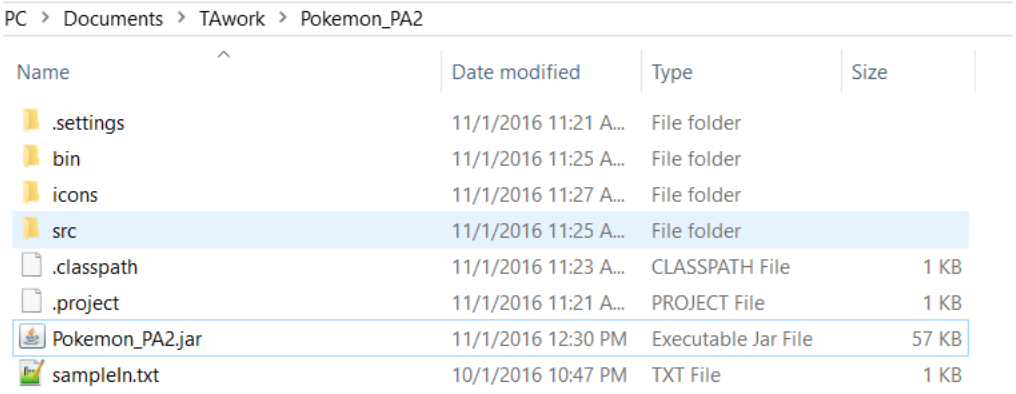
Please choose Runnable JAR file:



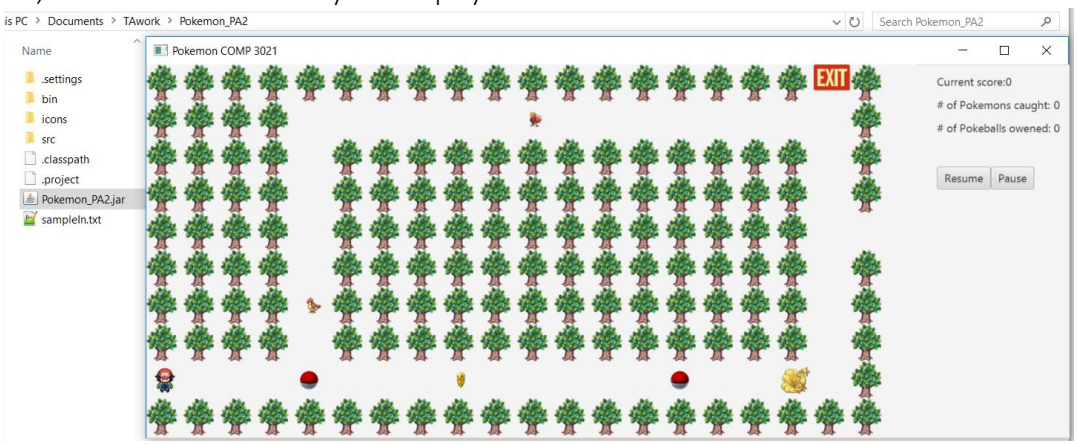
Please specify the launch configuration as the entry point of your whole program. It should be the same as Assignment 1, which is the main function in source file Game.java.



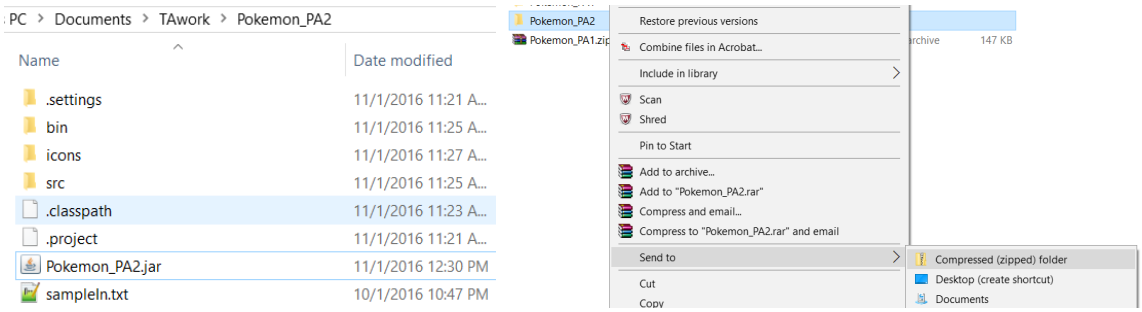
Please name the jar file as “Pokemon_PA2.jar” and put it under the folder of your project:



By default, it reads from the input file “sampleIn.txt” as Assignment 1. When you double click this jar file, it should run successfully and display the correct GUI.



Then you need to submit your project “Pokemon_PA2” as well as the runnable jar file in a zip file with the name “<Your student ID>.zip”.
For Windows users, you can do it by right clicking the “Pokemon_PA2” folder, and choosing ***Send to*** > ***Compressed(zipped) folder***. Please rename the archive file to <Your student ID>.zip.
For Mac users, you can do it by right clicking the folder “Pokemon_PA2”, and selecting ***Compress items***, and then you can find the new created .zip archive in the same directory. Please rename the archive file to <Your student ID>.zip.



Please submit the zip file ***“<Your student Id>.zip”*** (i.e. 20202795.zip) via the CASS system (https://course.cse.ust.hk/cass/student/#/submit).

Bonus

You are welcome to do bonus for this assignment, and we can give a bonus up to 10%.

The bonus for Assignment 2 is to enrich the action of catching. Currently, when the player meets any Pokémon, there is only one message in the information panel indicating whether the Pokémon has been caught or not. To get the bonus, you need to enrich the action of catching by popping up a fighting window. You can put any animations for the catching in this window. For example, you can create animations like throwing a poke ball. During the fighting, the original window should be frozen, which means all the Pokémons do not move. The whole game resumes after this catching action is finished.

Deadline

The assignment will be due at: 11:59pm 30 Nov 2016.
Submit a single file named as <Your Student Id>.zip.
Please follow the steps listed in the Submission Details.