Proejct 4: The Networked "Target Game" with GUI

a Java GUI Application with Networking Functionality

CISC 3120
Design and Implementation of Software Applications I

The project is to help students who have already had experience in programming in `C++` to achieve the following learning objectives in addition to the learning objectives defined in Projects 1, 2, and 3,

- to be able to design and implement File I/O;
- and to be able to design and implement Network I/O.

1 Project Description

We are now adding functionalities to the "Target Game" GUI application. The functionalities should include saving game statistics in a file, loading game statistics from the file, allow multiple players to communicate, and play on mulitple stations over the Internet (if no network firewalls in between).

The instructor wrote the start-up code for the project, and the start-up code is in a Maven project in the `sampleprograms` repository. You may browse the code from the following URL,

`https://github.com/CISC3120/sampleprograms/tree/master/TargetGameFXNet`

1.1 Project Requirement

This is a team project. You and your team collectively must complete the following requirements by making revision to the start-up code.

If you and your team complete *all* of the requirements in this section satisfactorily, the instructor considers the quality of the project as "A". Be aware whether you as an individual will receive the "A" grade for the project depending also on a project peer evaluation to be conducted anonymously after the submission of the project.

Note that the requirements in Sections 1.1.1 and 1.1.2 are identical to those in Project 3.

1.1.1 Maximum Guesses

This is in effect one of the requirement in Project 3. The start-up code permits the player to guesses any number of times before the player guesses correctly. In your project, you will set up a maximum guess threshold in the application. When the player exceeds the threshold without entering a correct guess, the player loses the game.

1.1.2 Add Game Statistics Pane

This is similar to one of the requirement in Project 3. You will keep track of a set of game statistics in the game. We call it a *round* when a player wins by guessing correctly or loses by entering too many guesses. The concept of *round* is used in the game statistics. The game statistics must include at least the following,

- the number of targets shot (i.e., the number of times the player produces a correct guess),
- the number of shots fired (i.e., the number of guesses the player makes in total),
- the accuracy of the player's guessing the target,
- the numbers of rounds that the player has won,
- and the numbers of rounds that the player has played.

In the start-up code, a `GameStatistics` class is written for this purpose, and a `TableView` is set up to display the game statistics. Your task is actually to make sure the display is up-to-date at any moment of the game.

1.2 Save and Load Game Statistics

The game has a simple menu where a user can save and load the game statistics. Once the statistics is loaded, the game should function as if there were no interruption in between after the player closes the application and restarts it.

1.3 Display Neighboring Online Players

The game will periodically *broadcast* the player's IP address and a listening TCP port number, perhaps, some other ancillary data, such as the player's name over UDP to port 62017. The game will also monitor UDP port, 62017, and receive broadcast datagram packets where the game can retrieve the IP address, the TCP port number, and other ancillary data of the senders (i.e., other instances of the game). The game has a `ListView`. The IP address and TCP port should be displayed in the `ListView`. This list constitutes the neighbor online players. Make sure that there is no duplicates in this list.

You are encouraged to add some ancillary data, such as, player's name in the broadcast messages. If you wish to include this, you should add an UI control or a JavaFX dialogue `Stage` (i.e., a window) to collect a player's name when the game states. However, this is not required.

TCP port number can be randomly chosen. The start-up code use a free TCP port reported by JVM. The purpose of this TCP port number is to support multiple players' online game playing as described in one of the bonus requirements below.

1.4 Bonus Project Requirement

If you and your team complete any one of the following *bonus* requirements *in addition to* the requirements above, the instructor considers the quality of the project as "A+".

1.4.1 Add a Messaging Functionality

Add UI component and networking functionality so that two players can write each other short messages. You can either use a second JavaFX `Stage` (i.e., a window) for this purpose, or add necessary UI components to the UI.

1.5 Add Multi-play over the Internet

The game should permits two players to play the game over the Internet in turns. That is to say, when one player connects to another player's game app via her own game app, the two players can take turns to play, and while one is playing, both players should see the target and guesses on their respective game user interface.

The instructor suggests the following design. Under the assumption that you have completed the functionality that displays neighboring players online, you (the initiator) can select (or click) a player from the list of online players, and the game will attempt to connect to the selected player's game that will in turn asks the selected player whether to accept the connection request. If the selected player accepts the connection, and the both players can play the game in turns as described. The selected player's game can randomly decide which of the two will play a round first, and send the choice to the initiator. After the selected player wins or loses a round, the turn is given to the initiator.

1.5.1 Display Shot when Target is Missed for Multi-Player Scenario

The start-up code uses an `Alert` window to display whether a guess is correct or wrong. To meet this requirement, you will display a "shot" in the main canvas (where the target is) when the player guesses wrongly, i.e., misses the target. You may choose any shape or size of your "shot", for instance, you may display the shot as a little box similar to the target, but with a different color, or better yet, you can use an image that resembles a smashed bullet.

Note that this requirement is similar to the one in Project 3. However, *the missed shot should be displayed in both players' interface*, which means to accomplish this, you must first complete the previous requirement.

1.5.2 Add a Shooting Pane

This is in effect one of the requirement in Project 3. When you run the start-up code, you can guess the location of the target and enter the guess only using the two `TextField` controls. Using the shooting pane, you can use mouse to guess where the target is. As shown in Figure **??**, when the mouse enters the shooting pane, you will display a cross that follows the mouse pointer. When you click the mouse button, a guess is entered and you will update the values in the two `TextField` controls and determine whether the guess is correct.

Note that this requirement is similar to the one in Project 3. However, the shooting pane should be synchronized when two players play the game over the Internet.

2 Project Invitation

Each team shall elects project coordinator for this project. The coordinator has the following responsibility,

- to accept the assignment invitation via the Github classroom;
- to clone the team project repository;
- to copy and add the start-up project to your own project repository
- to commit and push the project, and to inform team members that the project set-up is ready

The team members shall accept the invitation and clone the project repository. The collaboration and project development continues.

The project assignment invitation is at

`https://classroom.github.com/g/eQ0iXwnw`

3 Submission

Submit your project by pushing your project to Github by 11:59PM, Tuesday, December 5, 2017.

4 Appendix: Internationalization

The project is also to demonstrate the method to realize *internationalization*. The word internationalization is a very long word, and internationalization is an important issue to deal with in software application design. For convenience, we commonly refer it to as "i18n" where "i" is the 1st letter in internationalization, "n" the last , and "18" represents the 18 letters in between.

In the project start-up code, you should be able to find two resources files,

- `MessagesBundle.properties`,
- and `MessagesBundle_fr_FR.properties`,

which are for English and French, respectively. A resources file contains multiple lines, and each line is a key and value pair, e.g., "`keyname1=message to display`". The resources file is named in the format of `MessagesBundle_xx_YY.properties` where `xx` are two- (or three-) character language code (called ISO 639 alpha-2 or alpha-3 language code) and `YY` are two-character country code (called ISO 3166 alpha-2 country code). The resources file for US English is considered as the default, and can be named as `MessagesBundle.properties` without language and country codes' being embedded in the name.

The list of language codes are defined in ISO 639-2, and you can view the list at

`https://www.loc.gov/standards/iso639-2/php/code_list.php`

If the two-character code is defined for a language, you must use the two-character code. You may use the three-character code only when the two-character code is undefined.

The list of country codes is also an international standard, and you can view the list at

`https://www.iso.org/obp/ui/#search/code/`

The instructor recommends that you read Java API documentation on `Locale` at,

`https://docs.oracle.com/javase/8/docs/api/java/util/Locale.html`

You may consider these items to enhance i18n support of the game application, which helps you learn to write applications supporting multiple locales (languages and countries),

- To add more language to this game is to construct a message bundle file for a language and its corresponding country, called a `Locale`. For instance, to add the Russian language support for the country of Russia, we copy the US English message bundle file `MessagesBundle.properties` to `MessagesBundle_ru_RU.properties`, and translate the message for each key, and replace

Table 1: Example Message Bundle File Names

Language	Country or Region	File Name
Arabic (ar)	Saudi Arabia (SA)	`MessagesBundle_ar_SA.properties`
Chinese (zh)	China (CN)	`MessagesBundle_zh_CN.properties`
Chinese (zh)	Hong Kong (HK)	`MessagesBundle_zh_HK.properties`
English (en)	United States (US)	`MessagesBundle.properties` or `MessagesBundle_en_US.properties`
English (en)	United Kingdom (GB)	`MessagesBundle_en_GB.properties`
French (fr)	France (FR)	`MessagesBundle_fr_FR.properties`
German (de)	Germany (DE)	`MessagesBundle_de_DE.properties`
Hebrew (he)	Israel (IL)	`MessagesBundle_he_IL.properties`
Russian (ru)	Russia (RU)	`MessagesBundle_ru_RU.properties`
Spanish (es)	Mexico (MX)	`MessagesBundle_es_MX.properties`
Spanish (es)	Spain (ES)	`MessagesBundle_es_ES.properties`
Yiddish (yi)	Israel (IL)	`MessagesBundle_yi_IL.properties`

the English by the Russian translation. The examples of the message bundle files for a few more locales are in Table 1. This table is just a few examples, is far from complete, and your native tone and country may not be included.

- Whenever you display a message (a `String`), instead of writing a `String` literal, you retrieve a message via key look from the message bundle. In the Target Game start-up code, the `I18n` class is designed to support multiple languages. Below show a few steps you take to accomplish this. Be aware that these steps are more than often not written sequentially as below. Refer the game application start-up code and observe how the instructor handles it.

```
1  String bundleBaseName = "MessagesBundle";
2
3  Locale lc = new Locale("es", "MX"); // language and country
4
5  ResourceBundle bundle = ResourceBundle.getBundle(bundleBaseName, lc);
6
7  String msgKey = "numOfRoundsWon";
8
9  String msg = bundle.getString(msgKey);
10
11 System.out.println("Message is " + msg);
```