GUI

Project 2

Sławomir Aleksander Dańczak, M.Sc. Full-time studies 25 April 2021

AntiPlague Coronavirus Game

Write a program that is a game like" Plague Inc.". During the game a map of the world will be presented divided into countries (select minimum 10 countries). This map has various ways of traveling between countries (airlines, buses, boats, trains, cars, etc.). Traveling methods are optional, but there must be at least 3 (not all must be between all countries, but at least 2). The game is to prevent the infection of all people in all countries with a virus.

Each country has different criteria for hanging each of the transport routes between countries (criteria according to your creativity - e.g. according to the number of infected). Each of the criteria must be achievable and be able to be modified with purchased upgrades. You must create at least 9 improvements, the selection and operation of which depends on your imagination.

Upgrades can be purchased for points collected for anyone who has been saved from infection or cured. The virus begins to infect China in the Hubei province, and the manner, speed and spread effects depend on the level of difficulty and implemented at your discretion.

The game ends when we manage to protect the entire world population, stop the infection or when everyone is infected. Of course, progressive virus infection and its spread throughout the world should also be implemented.

A fully functional graphical interface should be provided. Transport between countries should be visualized using an animation, e.g. with an airplane icon. The command line console (*CLI*) can only be used as a help, but no user interaction with the program can occur there.

After starting the program should display the main menu consisting of the options:

- *New Game*
- *High Scores*
- *Exit*

After starting a new game, player will be asked in a separate window about the game difficulty level (at least 3 levels). After selecting level of difficulty, game window is displayed in the new window and time counter starts (it's worth noting that the time counter, like the virus behavior and others, must be ***implemented in separate threads***). During game must be visible points and time counter, which are constantly updated. The game is played according to the rules mentioned above. It should be possible to interrupt game at any time through the compound ***keyboard shortcut*** selected by you (***Ctrl+Shift+Q***), which will return you to the main menu.

After finishing the game, in the new window the player is asked for his name under which he will be saved in the ranking. Ranking is calculated based on the time, effect obtained and difficulty level (any implementation). You should save the ranking so that you do not lose saved records after closing the application. The form of stored data is optional (you can use e.g. interface ***Serializable***).

After selecting the ranking option from the main menu, it is displayed to the user. There may be a relatively large number of saved results, so you should take care of scrollbars in case it does not fit in the window of a reasonable size.

Hints:

- Take care of exceptions in the program. If any occurs, display its message to the user.
- High Scores list must be implemented using ***JList*** component.
- Countries can be implemented using buttons, but you can also design your own component.
- Not all windows need to be implemented via the JFrame class. Dialogs can be used.
- Take care of the appearance of the application

The project is based on GUI material.

The MVC design pattern should be used in the project.

Attention:

- ***It is not possible to use WYSIWYG tools to generate windows (e.g. Window Builder).***
- ***Lack of knowledge of any line of code or plagiarism will result in obtaining 0 points for this project with the possibility of failing the entire subject.***
- ***Not only the practical and substantive correctness of the solution will be assessed, but also the optimality, quality and readability of the code written by you.***
- ***An important part of the project is the use of: inheritance, collections, interfaces or abstract classes, lambda expressions, Java Generics, additional functionalities or structures and other characteristic elements presented in the classes and lecture (but only in a natural way, nothing by force).***