GUI

Sławomir Aleksander Dańczak, M.Sc. Project No. 2

Write an application that implements a game in the style of "Pacman". An example of the game is available **here**. In the game, every 5 seconds, enemies have a 25% chance to create upgrades (e.g. +50% movement speed, etc.) that the player can collect. Implement min. 5 different significant complex upgrades. After launching the application, it displays the main menu consisting of the following options:

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

When starting a new game, the player is asked for the size of the board. You need to implement the ability to create any window from 10 to 100 rows/columns. The generated game board is then displayed in a new window. It must be implemented based on the *JTabel* component, in which we will show the board fields in the cells. Implementing your own *JTable* model based on *AbstractTableModel* is mandatory.

A fully functional graphical interface should be provided. *CLI* can only help the programmer, but there can be no interaction with the user.

During the game, the score counter, time counter, life counter and other necessary elements of the graphical interface must be visible and will be constantly updated during the game.

You also need to use graphic files and create a cohesive look of the entire application, taking into account the appearance of windows such as *Menu*, *High Scores* and *game window*.

Using graphic files to visualize each element of the window is mandatory. In addition, you have to make the game characters move and perform tasks through simple animations (e.g. motion animation, food animation).

All things related to time must be done using the Thread class (the *Timer*, *Executor* and other classes are not allowed). Careful and correct synchronization of all threads must be ensured. You cannot combine different functionalities into one thread.

The game is played according to the rules mentioned above. It should be possible to interrupt the game at any time through the ***compound keyboard shortcut - Ctrl+Shift+Q***, which will return you to the main menu. Ensure the keyboard shortcut works at anytime during gameplay.

At the end of the game, the player is asked for the name under which they want to be saved in high scores. Provide high scores persistence using the ***Serializable*** interface.

You should save the ranking, so you do not lose saved records after closing the application. You must use the interface ***Serializable***.

After selecting the High Scores option from the main menu, it is displayed to the user. Because the High Score window can be large, take care of the scrollbars.

Implement the application using the MVC programming pattern with complete event handling implemented by the delegated event handling model.

Hints:

- Take care of exceptions in the program. If any occurs, display its message to the user.
- High Scores list must be implemented using the ***JList*** component.
- You should take care of the scalability of application windows.
- Not all windows need to be implemented via the JFrame class. Dialogs can be used.

The project is based on GUI material.

Attention:

- ***In the case of receiving a project that does not comply with the requirements or with significant deficiencies in implementation or a non-compiling solution, the result for such a project will be 0 points.***
- ***It is impossible 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).)***