







Implement a one-window application analogous to "Monkeytype" ([look here](#)). The application is designed to check the efficiency of fast and precise typing and to display statistics.

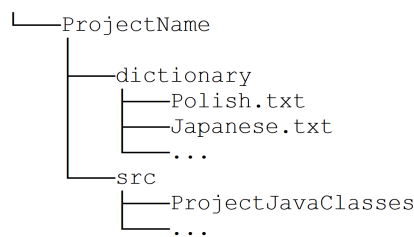
A menu with available options is displayed on the top panel (language selection, test time selection) as part of the application.

On the bottom panel, we display a footer with information about the available keyboard shortcuts (implement the indicated functionalities):

-  +  - restart test
-  +  +  - pause
-  - end test

Within the application, the user selects one of the available languages. The list of languages depends on the number of text files provided. In the attached *dictionary.zip* file, there are 19 text files (encoded in UTF-8) representing different languages (50k-600k words for each language). Each file is named according to the language it represents. The application should be designed so that if the number of files changes, the application will adjust the number of available languages without any modification.

The *dictionary* directory should be placed directly in the project structure:



During the test, the user tries to write the text visible in front of them as quickly as possible.

We consider each word separately. We move on to entering the next word after pressing the space button. Make sure to change the color of the characters, as in the following example:

- Text that has not yet been entered is grayed out:  
make turn interest write time present little around eye
- Correctly entered characters appear green:  
make turn interest write time present little around eye
- Incorrect characters appear in red:  
make turn interest write time present little around eye
- Redundant characters appear in orange:  
make turn interest write time present littleeee around eye
- Omitted characters appear in black:  
make turn interest write time present littleeee around eye

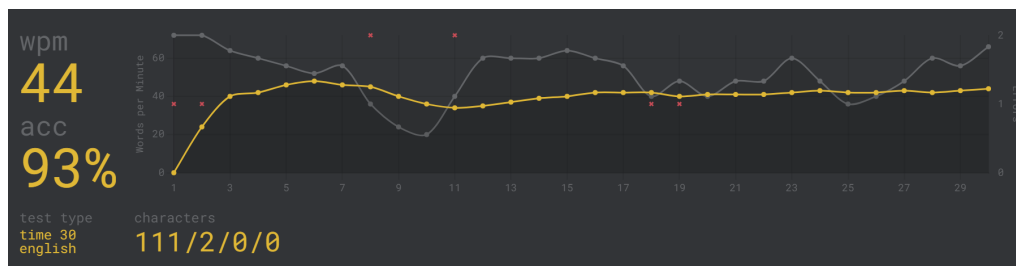
Implement the option to select the test's duration: 15, 20, 45, 60, 90, 120 or 300 seconds. We display one paragraph to the user each time, and when we reach the end of it, we replace the completed paragraph with a new one. Each paragraph has 30 random words from the selected language's dictionary.

Implement the animation of jumping letters of the entire text, creating a 'wave' (I warn against using ready-made solutions from the Internet). Also, implement any other two animations to any window elements.

After the test, you should display a chart presenting (broken down into successive seconds):

- current WPM (word per minute)
- average WPM

You should also calculate the average WPM for the entire test and the statistics of the entered characters in the form (correct/incorrect/extra/missed) and specify what % of characters were entered correctly. An example of such a chart and statistics looks like this:



As a result of the test, a file with the name containing the date and time of the test must be created in the project directory. The file will present a list of subsequent words and the calculated WPM for a given word.

The contents of such a text file must look like this:

```
become -> 70wpm
will -> 83wpm
begin -> 99wpm
no -> 106wpm
see -> 80wpm
keep -> 58wpm
...
```

#### ***Remember:***

- Take care of exceptions in the program. If any occurs, display its message to the user.
- You should take care of the scalability of the application window.
- ***Implement the application using JavaFX technology and the MVC pattern***

#### ***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 (e.g. Window/Scene Builder).***
- ***FXML files cannot be used in the application***
- ***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 will be assessed, but also the optimality, quality and readability of the code written by you.***