Block 3 - Exercise for Units 1 and 2

For this exercise, you are asked to create an IntelliJ project called **TourDeFrance**, in which we are going to manage some statistics regarding this cycling race. Inside the source define a package called `race`, in which we are going to place every class of this project.

1. Cycling stage class

In order to store the main information of every cycling stage, we are going to define a `CyclingStage` class. The main information that we want to store is the date of this stage (a string with the format *yyyy-mm-dd*, such as *2021-05-21*), the total number of kilometres and the full name of the winner. Add a constructor to set all these parameters, the corresponding getters and setters and a `toString` method to show the information of the stage with this format:

```
date (km) winner
```

For instance:

```
2021-05-21 (186 km) Tadej Pogacar
```

2. The FileUtils class

Besides, you are asked to add a class called `FileUtils` with these two static methods: `loadStats` and `saveStats`

loadStats method

`loadStats()` method will read a text file called `stats.txt` containing some cycling stages with the following format:

```
date1;km1;winner1
date2;km2;winner2
...
```

For instance:

```
2021-05-21;186;Tadej Pogacar
2021-05-22;198;Primoz Roglic
...
```

This static method must read the contents of the text file and return a generic `ArrayList` of cycling stages (objects of type `CyclingStage`), parsing the contents of the file and creating one `CyclingStage` object for each line. We assume that the information stored in the file is correct, but if file doesn't exist, then this method must return an empty list (not null).

saveStats method

Regarding `saveStats` method, it must receive an `ArrayList` of `CyclingStage` objects as a parameter, and it must store them in the text file `stats.txt` with the same format explained above, sorted by date in ascending order. The previous contents of the file, if any, must be completely erased when saving the information again.

Error management

In both methods, you must make sure that you catch every possible exception that may occur. Also, you need to close the corresponding file, and show the appropriate error message, if any.

3. The main application

Finally, add a third class file called `Main` to the project, with the `main` function on it (along with any other additional function that you may need). At the beginning, the program must load the cycling stages from `stats.txt` in an `ArrayList` (using `loadStats` method explained above). Then, it will show this menu to the user:

- **1. Add stage**: user will be asked to enter the information of a new stage (date, kilometres and winner name) and, if date is not repeated in the list, it will add the new stage to the list. Then, it will save the new stage list into `stats.txt` file (using `saveStats` method explained above).
- **2. Show winners**: it will show the names of the different winners of the stages (with no repeated names).
- **3. Top winner**: it will show the name of the winner with more victories in the race. If there are more than one cyclist with this maximum number of victories, then the program must show only one name (any of them).
- **0. Exit**: to exit the application.

4. What to submit?

You must submit a ZIP or RAR file containing the whole IntelliJ project. In order to get to the `stats.txt` text file properly, you should place it in the root folder of your project. You don't need to remove this file when you submit the project.

5. Evaluation criteria

This exercise will be evaluated as follows:

- `CyclingRace` class: 1 point
- `FileUtils` class: 3 points (1,5 points for each static method)
- Main program - add a new stage: 2 points
- Main program - show winners (avoiding repetitions): 1,5 points
- Main program - top winner: 1,5 points
- Code cleanliness, modularity and reusability: 1 point

> **VERY IMPORTANT**: every exercise that does not compile will be automatically evaluated to 0.