CORE MANAGEMENT

The `Core` part of the project is the gate between our graphic libraries and our game libraries.

This class is contained in the `arcade` namespace, and is initialized inside the `Arcade` class. The latter will parse the lib folder and the user arguments and create the `Core` class.

Roles of the Core class

Loads the libraries

The game and graphic libraries are dynamic, so that they can be changed during runtime. The `Core` class stores two `Loader` instances, so that it can open, close and load one graphic and one game library during the program's execution.

Error handling

If a library is not loaded correctly, of if during the execution of the games there is an issue, the `Core` will throw a custom exception called `Error` inheriting from the `std::exception` class. It will then stop the execution of the program in order to ensure the program's safety.

Handles events during execution

According to the player's actions, it will handle all events related to the program's execution. All events are triggered by key bindings in the program, but note that mouse events are set in our graphic libraries but not handled in the `Core` . You can check the list of keybindings availables in the `README` file at the root of the repository

Handles the program loop

As the 'gate' of our program, the Core has the responsibility to handle the execution of the program. The method `executeLoop()` handles the different scenes as well as gets the events and key pressed by the user. The methods

```
void displayMenu(Events& event);
void displayGame(Events& event);
```

handle the display of the scenes, as well as the `menu` and `game` loops.

Handles the scoreboard

Scores are automatically updated and saved at the program's end and when the player goes back to the menu after a game. A scoreboard system is also set up with the 5 top scores.