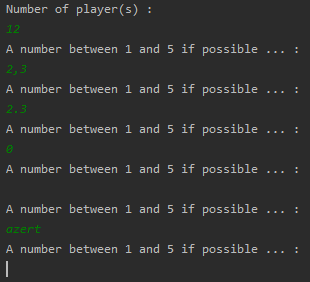


About Testing:

Here we’re going to talk about the security of the user inputs, the way we handled the code, so it can’t crash mid-game.

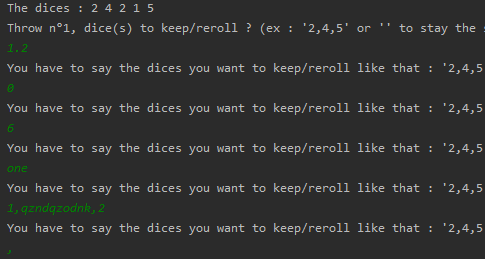
The main concern is to forgive the user if he does a bad input, if he’s writing an integer instead of a string or something like that.

Every check is done by a ‘try / catch’ of exceptions so we’re sure that the user can’t make our game crash with one of them.

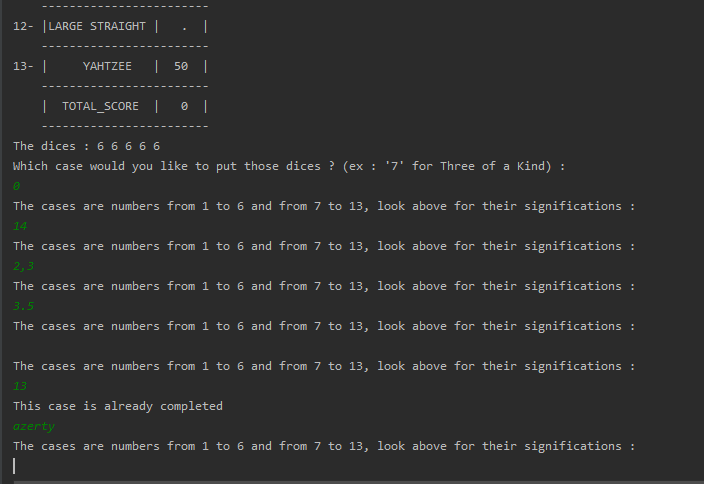


At the start of the game, we want to make sure that the number of players isn’t too excessive. Also, we take care of any wrong input like a ‘null’ entry, a random string or a double.

Everything is going to be rejected except an integer between 1 and 5 included.

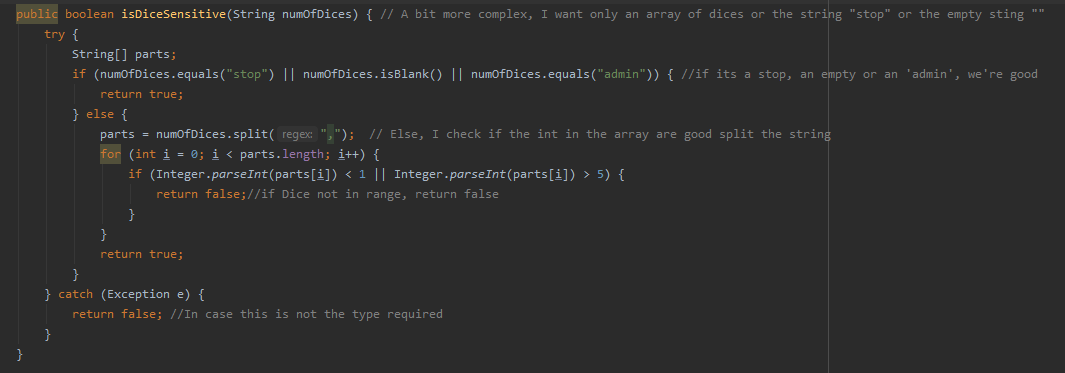
Here we’re waiting for more than an integer, in fact the user can write 2 types of answer, an array of integer (like: ‘1,2,3’) or a string with specified commands (which are: ‘stop’, ‘admin’ or null).

If one of the inputs in the array is false, the code will reject all the input (like in the last example) because we check each cases of the array before running our function on it.



Here there is also the case were the user is trying to put a score in a case that is already completed so we made a custom answer, so the user know that the input should work but in that case it doesn’t because of the game rules.

Once again, we check for all wrong entries like an unexpected string, float or null entry. All that managed with ‘try and catch’ function that are used in Boolean and called in while functions so the user



Here is the example of one of the ‘Try Catch’ functions, this one, ‘IsDiceSensitive’ is used when we’re asking the user to choose an array of dice he wants to keep or reroll, or one of the commands ‘admin’, ‘stop’ or the null String. (seen in picture 2).

The command returns a Boolean that is true if the input is one of the expected answers. In the others cases its false and we’re asking the user to write another answer that match the expectations.