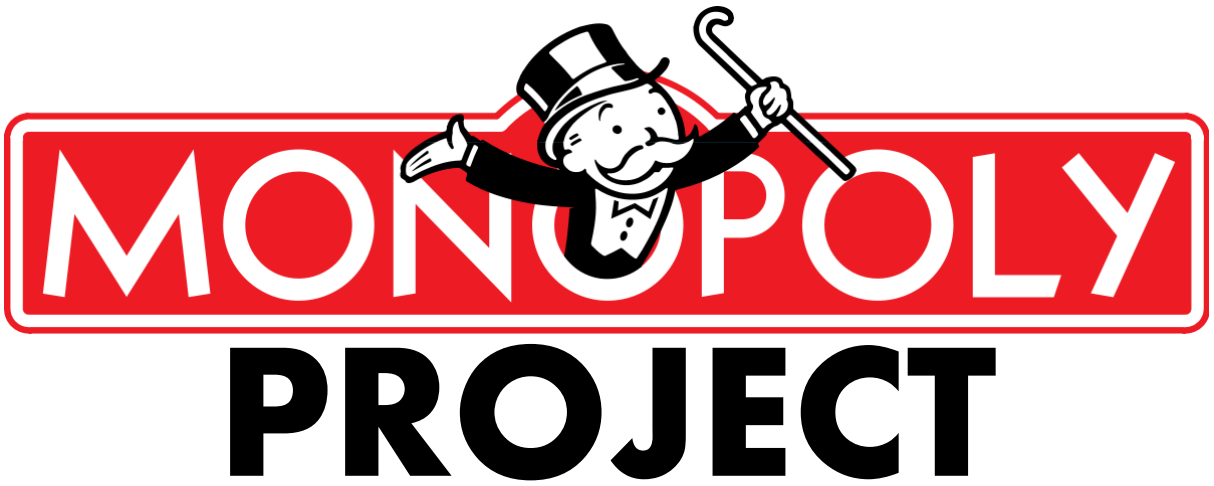
DESIGN PATTERNS AND SOFTWARE DEVELOPMENT PROCESS



PROJECT

BORDAS Alexandre – BUQUET Sébastien – CARVAL Nicolas

Table des matières

1. Introduction (a brief description of the problem)

We were tasked with the creation of a simplified Monopoly game.

We were asked to implement:

- A board of 40 cases (index from 0 to 39) with a case jail/simple visit and a case go to jail
- The roll of the dices (2 dices in this version of the Monopoly)
- The movements of the players on the board (several players may be on the same case), possibility of a player to play again if he made a double. The movement of a player is the sum of the value of his dices.
- The ways to go in and out of jail
    - Two ways to go to jail:
        - The player stops on the case “go to jail” (case 30)
        - The player realises 3 doubles in a row
    - Two ways to get out:
        - Realize a double
        - Wait three turns of the player (his previous two turns after moving to jail and his current turn)

    When the player goes to jail, the player position is updated to 10. When he is delivered from jail, the player moves from the sum of the roll of his dices.

2. Design Hypotheses (specification and explanation of details not explicitly indicated in the project assignments, but required for the solution's design and implementation)

Design patterns used

In this project we use two patterns: the singleton and the state pattern.

We use the state pattern to realise the prison process. We created two states for the player: a state free and a state jail that modify the rules of the turn. The state free state implements the general rules that are applied to the players and the state jail are the rules applied when the player is in prison. In a concrete way we have an abstract class state and two classes that inherit of this class stateFree and stateJail.

The second pattern used is a singleton pattern on the board to have the possibility to create only one board and not create error by referencing to different boards in the code of the game.

End of the game

As we did not have a rule to finish the game, we took the solution that the party ends when one player has realized five laps of the board. It is long enough to check if the game works well and short enough that the user doesn't need to click on roll the dices too many times.

Future development of our product

We have created several classes and attributes that were not necessary for the simplified Monopoly but would be useful if we wanted to improve and complete it one day.

The case class is indeed not useful because we can just stock the position of the player and it would be enough to realize the project. But if we want to add the price of the case, the cost of a house, the different rents that the player must pay when he walked on the case, the case class became useful. All these data could be either stocked in a file outside of the C# code or in an enum class. Therefore, we created an enum class nameCase to give an exemple of this.

Another example of this is the creation of the board class. That is list of case that contain forty cases. It became useful as a reference point if the player needs to know what case he falls on.

Visualization

Another hypothesis is the way to visualize the game. At first, we created a console program, but it was not easily comprehensible for the user, it is the Monopoly class of our project. Therefore, we realized a WPF with three windows.

The first window, the home, gave the user the possibility to choose between two, three or four players by entering a name in a case or not. Even if the user enters only one name a second will be added automatically, the monopoly is indeed played with at least two players.

The second window is the game window, we can see the board, the tokens, the dices. It is on this window that the game happens.

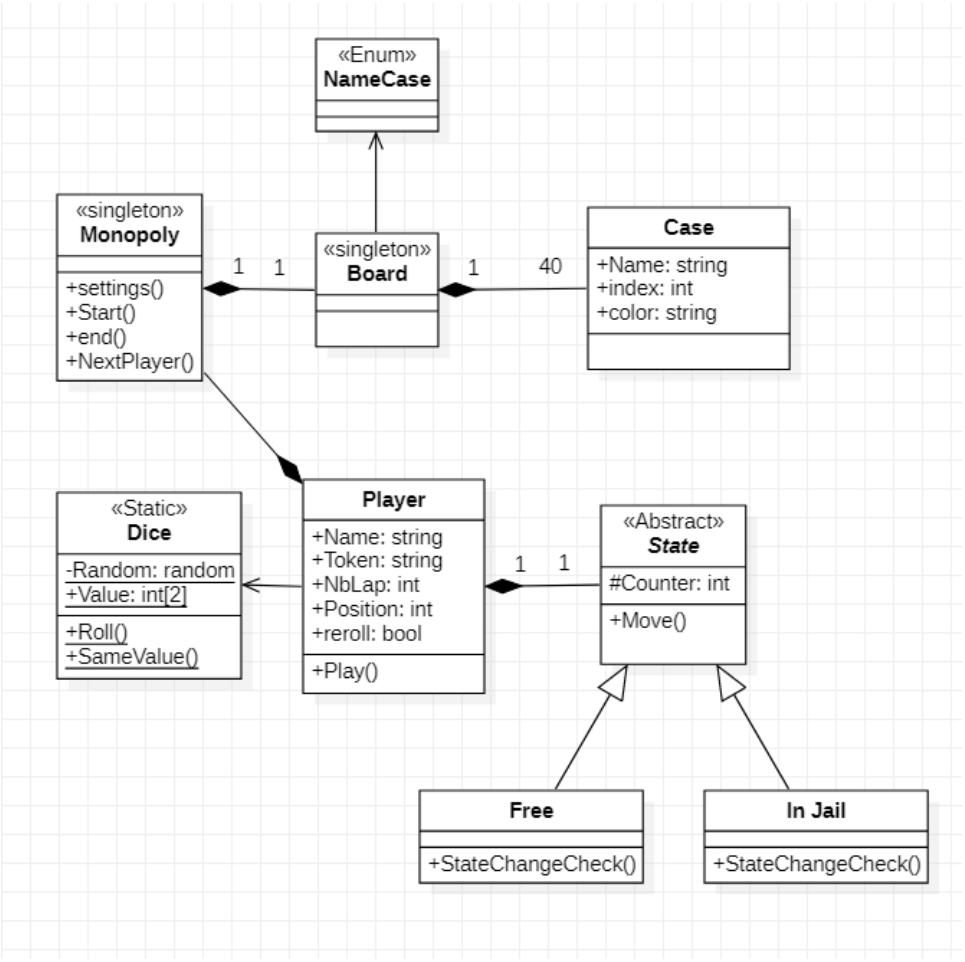
The third and last window is the end window. When one of the players realises five laps, the page automatically changes to the end page. On it the user can see a message that congratulates the winner of the game.

Number of players

The number of players was at first an arbitrary decision, but it became different with the creation of the WPF. We choose four players maximum with the idea of visibility. If we added more players the tokens became too small, and it was hard to see well the position of the player on the board. Moreover, we didn't need more players for testing the game.

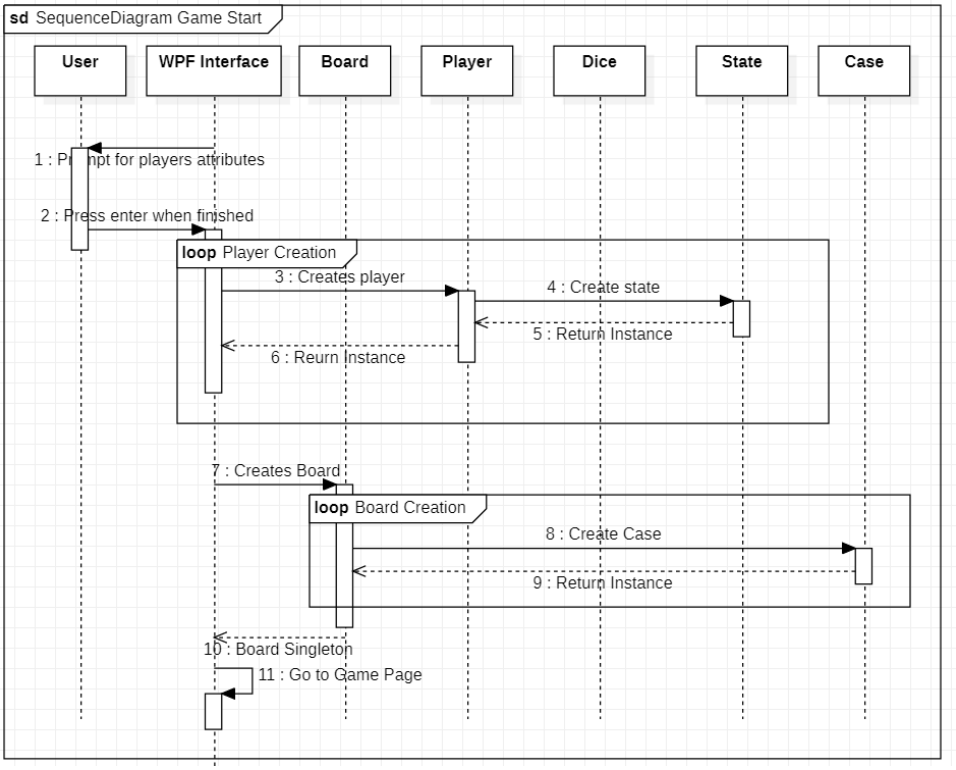
3. UML diagrams

a. Class diagram of the solution

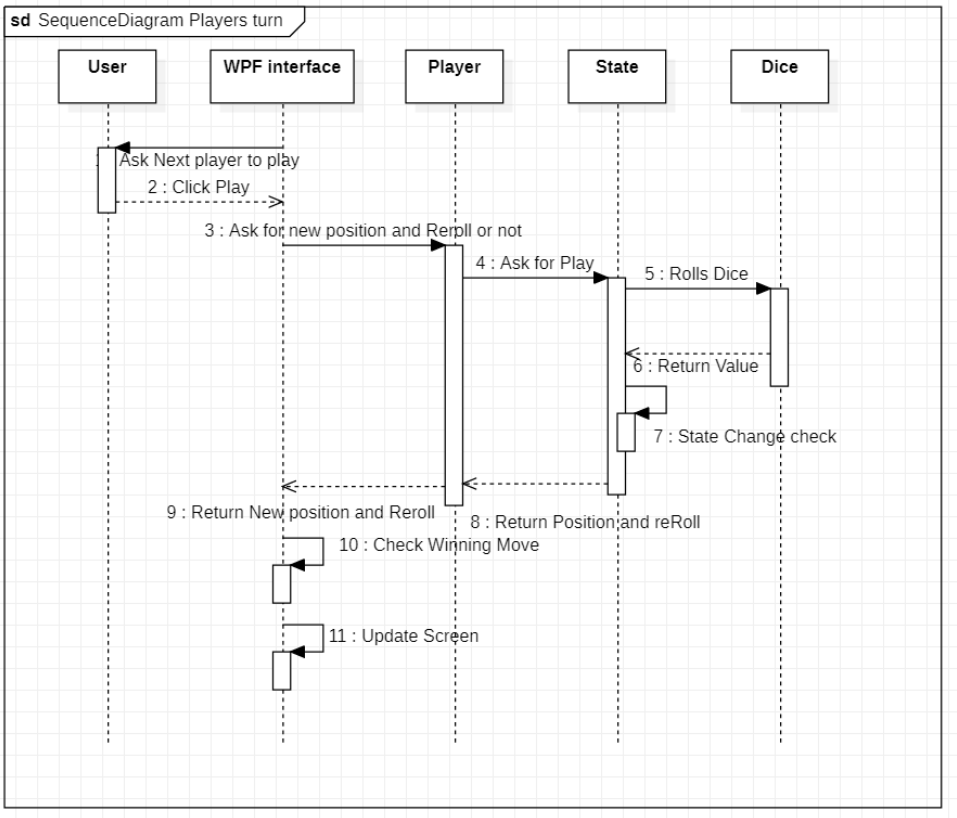


b. Sequence diagrams (at least, one sequence diagram of one solution's use case)

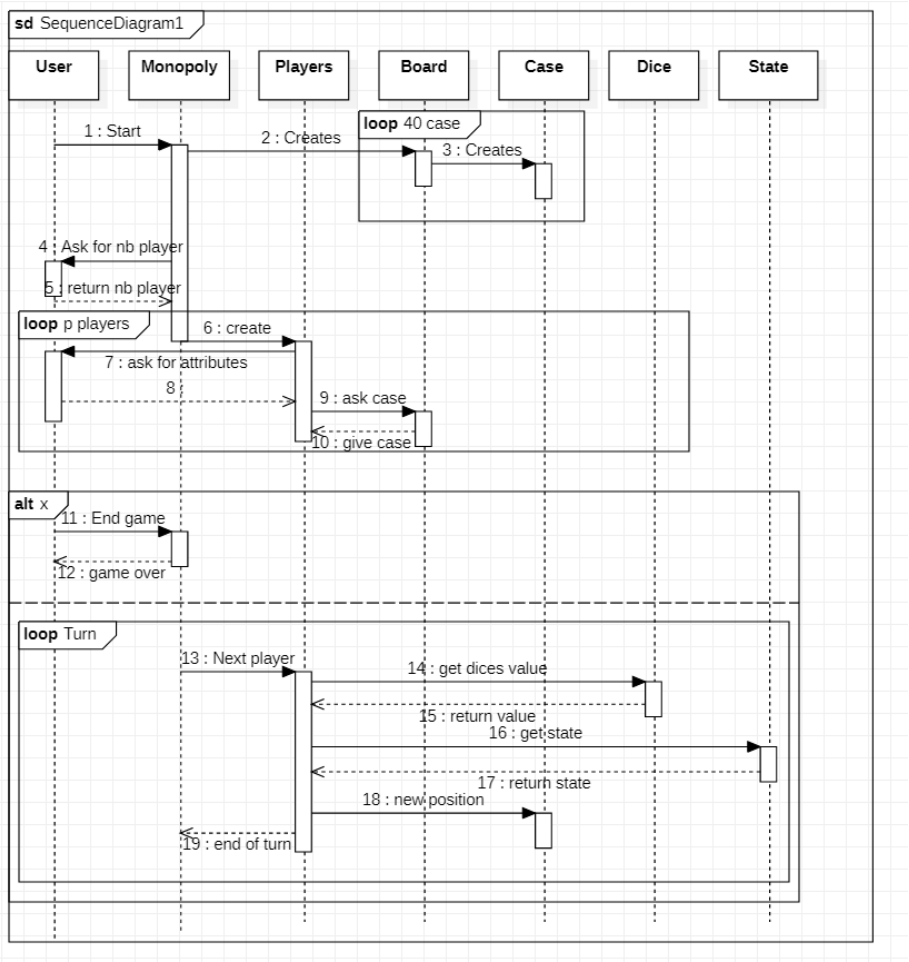
Start of the game:



Player turn :

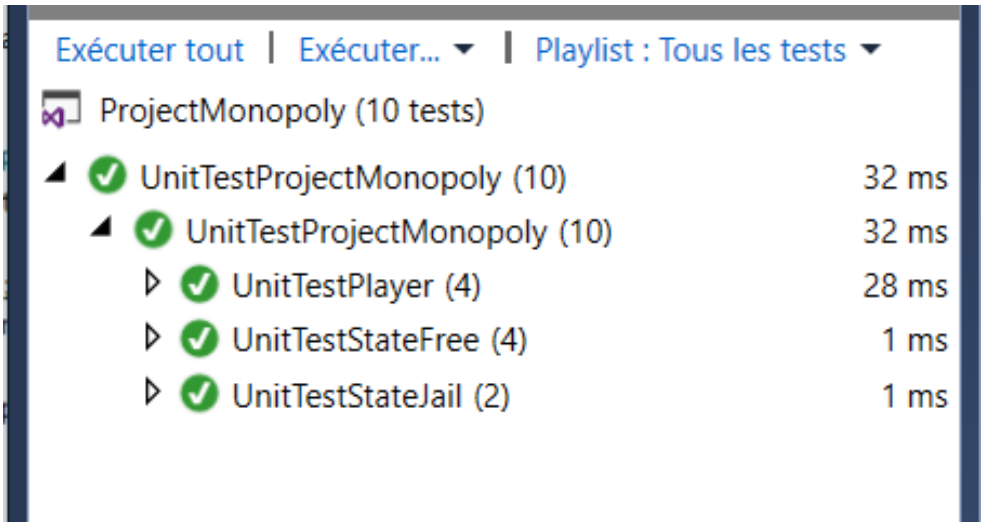


Or for the Console Project



4. Test cases (description of the employed techniques; specification of at least 2 executed test cases, a sample of input data (if present), expected and obtained results)

For this project we created a Unit Test project, in order to test the Player, StateFree an StateJail classes. Player class has 4 unit tests, StateFree class has 4 unit tests and StateJail 2 unit tests. Which adds up for a total of 10 unit tests.



Here are also some test cases that we created :

Test Case N°	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
N°1	Check Creation player with valid data	1- Start WPF program 2- Enter “p1” in first section 3- Enter “p2” in second section 4- Press button play	-Name for first player “p1” -Name for second player “p2”	Creation of the 2 players, resulting in the apparition of the names in colored boxes, and of colored token associated	As Expected	Pass
N°2	Check Creation player with invalid data	1- Start WPF program 2- Press button play without typing anything		Creation of 2 player by default, “Player1” and “Player2”	As Expected	Pass
N°3	Check screen update when player move	1- Execute every steps from test case n°1 2-press button Roll	Same data as test n°1	Token of first player has moved of X case corresponding to the sum of the displayed value for the dices	As Expected	Pass
N°4	Check end of game	1- Execute every steps from test case n°1 2- press Roll till one of the player has moved enough to complete 5 laps	Same data as test n°1	Screen has been updated and display the name of the winning player, the player who has completed 5 laps	As Expected	Pass

Here are examples of the unit tests created :

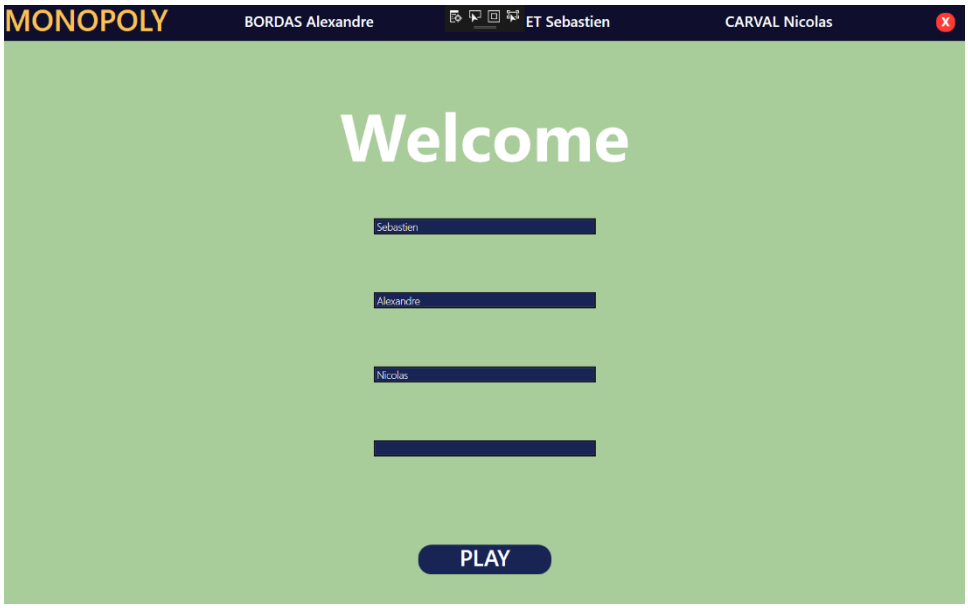
```
/// <summary>
/// This test aims to check that the players goes to jail when
/// played thrice in a row
/// </summary>
[TestMethod]
public void Test_InJail_When_Played3times_InARow()
{
    //Arrange
    Player p1 = new Player("player1", "car");
    StateFree current = new StateFree(p1);
    //Act
    current.Counter = 3;
    current.StateChangeCheck(p1.Position);
    //Assert
    Assert.AreEqual(p1.Position, 10);
}

/// <summary>
/// this test aims to check that a player
/// goes to jail when landing on "go to jail
/// </summary>
[TestMethod]
public void Test_InJail_When_On_GoToJail_Case()
{
    //Arrange
    Player p1 = new Player("player1", "car");
    StateFree current = new StateFree(p1);
    //Act
    p1.Position = 30;
    current.StateChangeCheck(p1.Position);
    //Assert
    Assert.AreEqual(p1.Position, 10);
}

/// <summary>
/// This test Aims to check that the position
/// of a player goes back to the beginning when
/// it is the end of a lap
/// </summary>
[TestMethod]
public void Test_PositionUpdated_When_EndOfLap()
{
    //Arrange
    Player p1 = new Player("player1", "car");
    StateFree current = new StateFree(p1);
    //Act
    p1.Position = 43;
    current.StateChangeCheck(p1.Position);
    //Assert
    Assert.AreEqual(p1.Position, 3);
}
```

5. Additional / Final remarks (optional)

Here is a presentation of how is displayed our work, it shows different cases that can be encountered during a party

Home page :

Input of the names of the players then « play » to begin the game. Here is an example with three players but this Monopoly can be played with either two, three or four players.



Start of the game :

All players are starting from the “Go” case. The dices are showing default values, It is Sebastien turns (name highlighted in yellow on the right side).

After clicking on “Roll”, we can see that the dices are showing the last values obtained, Sebastien got 4 twice, thus he is in the 8th case, his name is still highlighted as he continues to play thanks to the doubles.

After clicking again on “Roll” Sebastien made 7, it is not a double, therefore it’s Alexandre turn.

Further in the game, as the game continued playing, Alexandre went to jail because he dropped on “prison” case. He already waited 2 turns, without making doubles. It is now his third turn in jail.

Rolled the dice, even if it wasn’t doubles, he went out of jail, the position is updated.

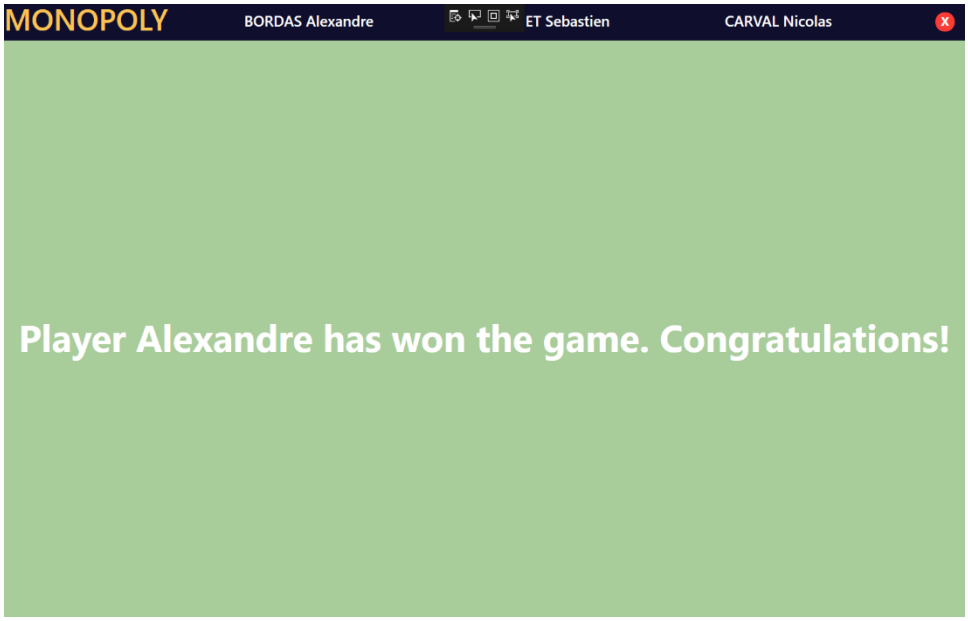
Here another possibility for the jail :

Dices showed that Nicolas didn't make double last turn, thus didn't move and had to wait.

Game over:

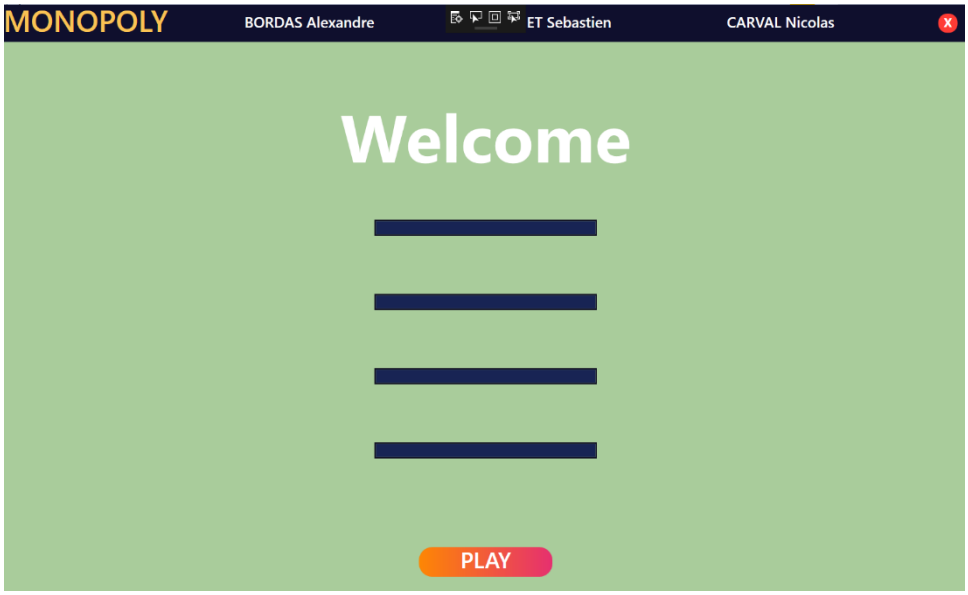
The first player to complete 5 laps won.

In this game Alexandre won. The Monopoly is now over.



Additional cases :

What happens if no player have been typed by the user ?



The game creates automatically enough default players in order to be able to play.

In this case 2 player were created by default.