1. Get the movement to execute by the current player.
2. Check if the movement follows the notation system.

In case the notation is not respected, or it is not a possible movement, the input is asked again to the player.

In case the notation is good, we proceed with the program logic:

1. By using the Player class, we check if the current player has the piece to move and if this could realize the desired movement:
   1. Check for Bishops if notations demands a bishop (e.g. “Bf6”)
   2. Check if the movement of this piece allows to arrive from its current position (stored with the ID of the piece in a list for the player) to the final position (stated in the input).

In case one of the two conditions is not good, we return a FALSE and we should restart by asking a different movement, giving the reason to add the player to take a proper decision.

In case the piece exists and is able to realize the movement, we proceed with the program logic returning a TRUE:

1. By using the Board class, we check if the final position is occupied or not and if it is coherent to the movement to execute (it will change the result depending if we want to do a simple move, eat a piece…)
2. By using the Board class (positions to be traversed could be returned together with the TRUE and passed to Board class), we check if the path to use is free or not.

If the movement is feasible, it is performed and we continue with the step 6. Else, we need to ask for a new movement giving a reason for it.

1. We change the position of the used piece on the list of the player.
2. In case of eating a piece, we should change the pieces list of the opponent.
3. We implement the changes on the Board class before printing it in the console with the opponent perspective.
4. In case of check or check mate, a message is displayed !