Input/Output notes.

The player open the game. He can :

- Play against another person
- Play against the computer
- Load an old game
- Quit the game
- Lose the game

Play against another person will be the same as play against the computer, because the computer just will make the piece move.

At any time on the game, we will be able to see :

- Each piece captured
- Time consumed

At any time of the game, we will be able to :

- Quit the game
- Quit the game, saving it
- Pause the game (hide the board)
- Play the game

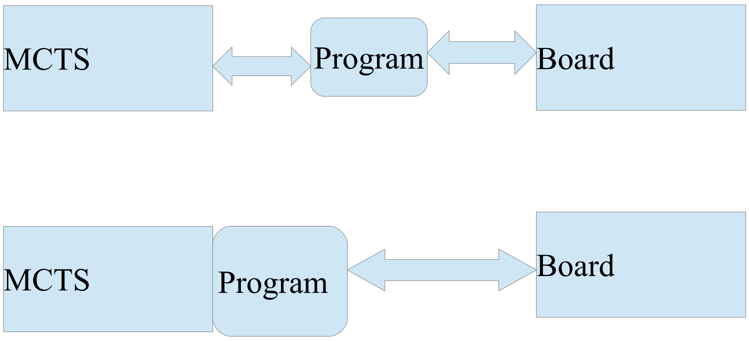
The API should be able to recognize (this is input):

- Keyboard moves
- Mouse moves
- Maybe console moves (see later)

The output will be the display of the board, and confirmation windows (do you want to save)

The question still to be developed is the relation between MCTS and the application (it's the display so we'll call it board)
We have to create a program that makes communicate the two entities. We'll use Maps (type Hashmap in Java) to translate a language to another.

Implementations :



Hashmap :
Key : Rabbit, Value = 0001
Key : Cat, Value = 0010
Key : Dog, Value = 0011
Key : Horse, Value = 0100
Key : Camel, Value = 0101
Key : Elefant, Value = 0110

In the first implementation,
the board and the MCTS program are communicating with a program that translate and copy again all data. It's the simplest but it cost more, it copies data twice.
The second implementation handles the translation in the MCTS side, not in the same class, but it will use a Interface that will be filled by Baptiste Set of rules. So there will be only one copy and no more.
There will be active waiting, a method from the board will call the MCTS program by sending data that the program will translate and give it to the MCTS. The MCTS program will process it. During

that time, the Board will wait (Active Wait) I think interruptions are not really useful here, it doesn't cost a lot to just wait, but who knows ?. Then the MCTS program will send data to the program, and the program will answer the board.