TIC TAC TOE

Language: python

File:
or_game_AI.py
test.py

Explanation:

Tic-tac-toe is a paper-and-pencil game for two players, X *and* O*, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game*

First player first to play.
The board size 3X3.
First player is - X
Second player is - O

Board

The board:

```
 --- --- ---
|   |   |   |
 --- --- ---
|   |   |   |
 --- --- ---
|   |   |   |
 --- --- ---
```

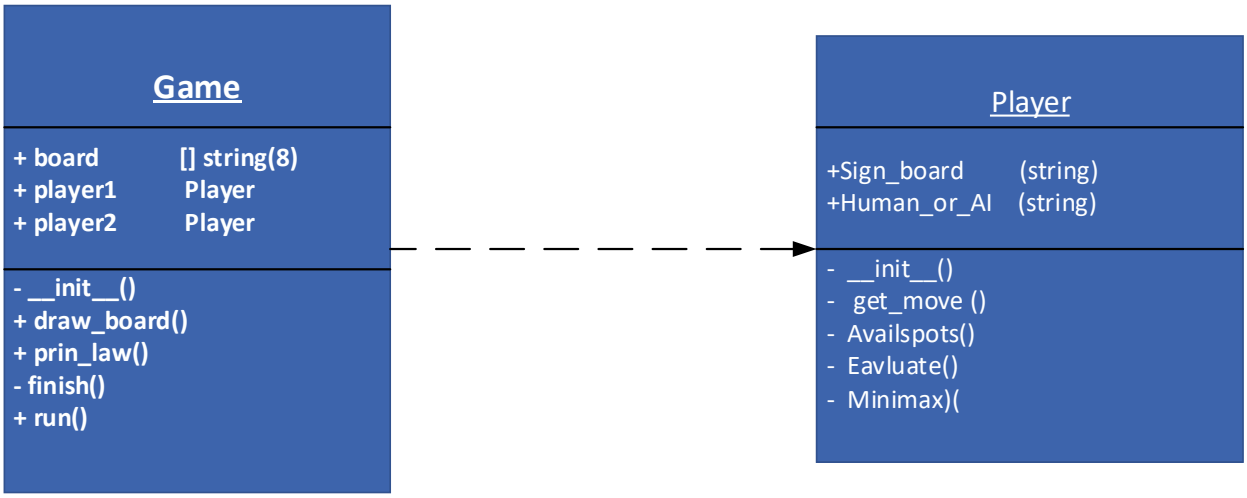
0	1	2
3	4	5
6	7	8

First, the program asks you to enter to opponent type:
1 for play with the computer
2 for play with the person

The program run 9 times.
Every one of the players will write the number he wants to put his mark on.
when someone win the game the program stops and print the winner.

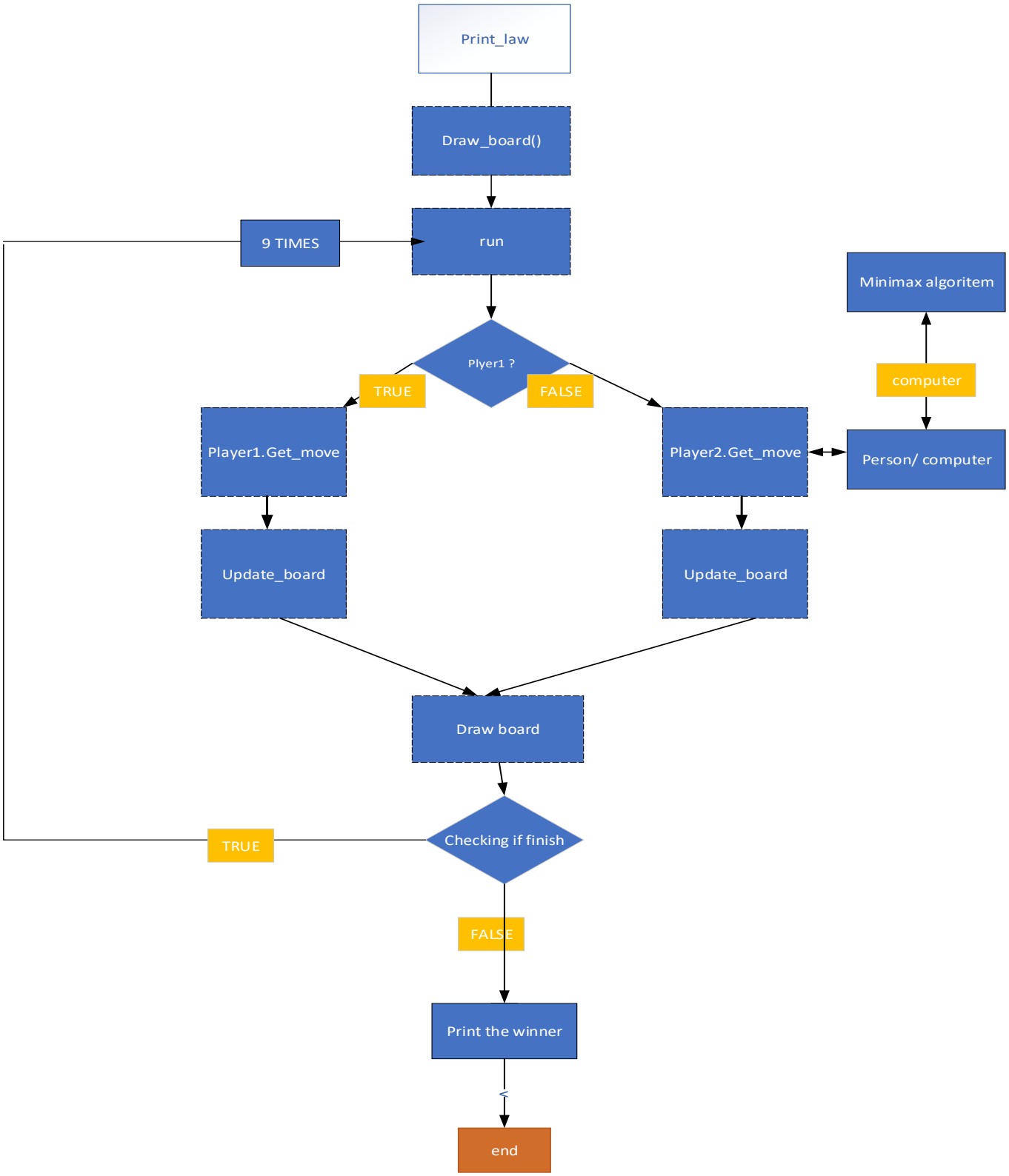
Uml Diagram

The relation between the class:



The main driver

The program process:



Minimax algorithm

Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the opponent is also playing optimally.
As its name suggests, its goal is to minimize the maximum loss (minimize the worst case scenario).

Minimax pseudo code

function Minimax (game, depth, player):

```
if player sign board = O
        best move, best = [-1, -1000000]
else
        best move, best = [ -1, 1000000]
```

For each state, we will initialize the worst-case value for the player's perspective.

Our move at this point is undefined and initialized to -1

```
if current game state is a finish state or depth = 0:
        return [-1, value of the game]
```

If we finish the game we will return -1 and the value we finished with

```
for each cell in available spots in game:
```

Everywhere blank on the board - consider it as an optional movement

```
        game board [cell] = player sign
```

Update the board by the empty place

```
        if player sign board = O
                move, score = minimax (game, depth-1, X)
        else
                move, score = minimax (game, depth-1, O)
```

We will be called to the next state and consider it his best value

```
        game board in cell = " "
        move = cell
        if player == "O"
                if best < score
                        best move, best = move, score
        else
```

For a given state, we find the optimal state from a computer perspective

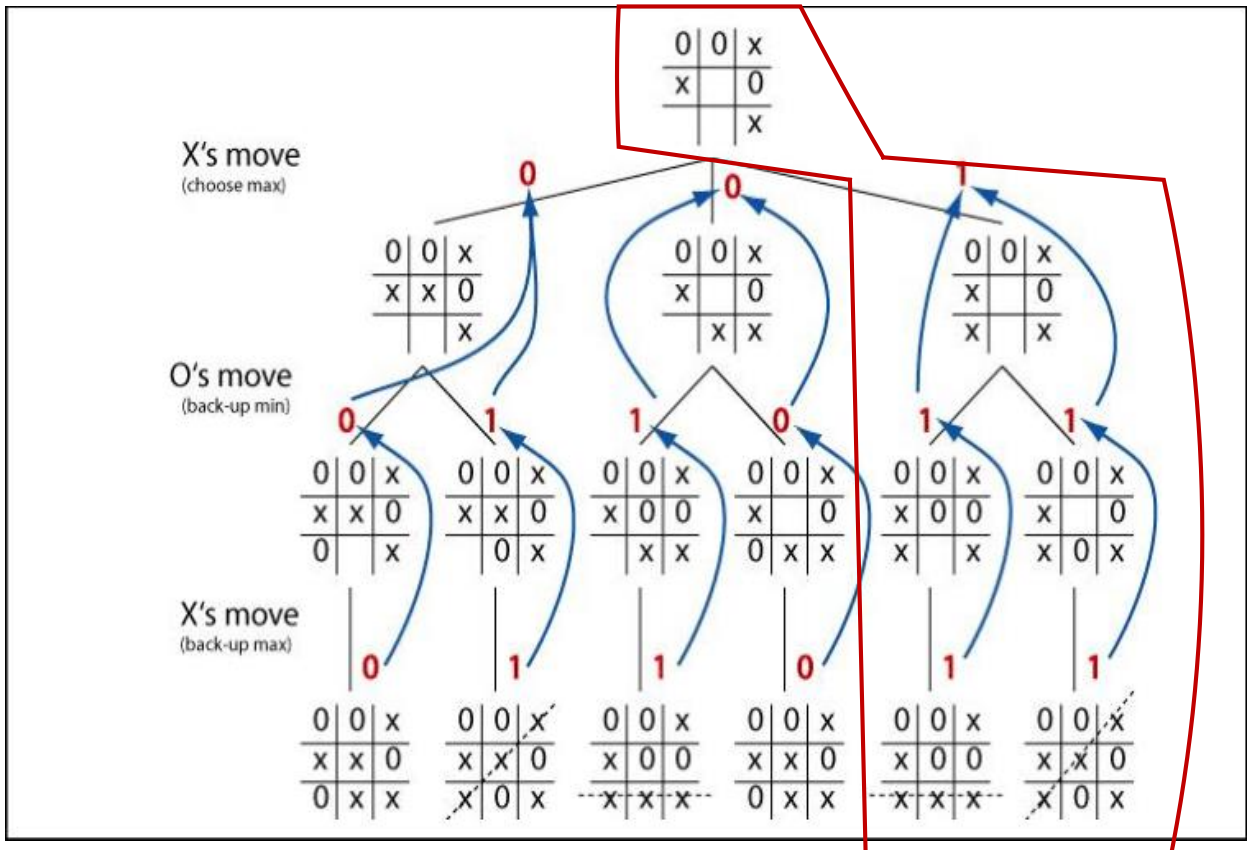
if **best > score**

best move, best = move, score

return **best**

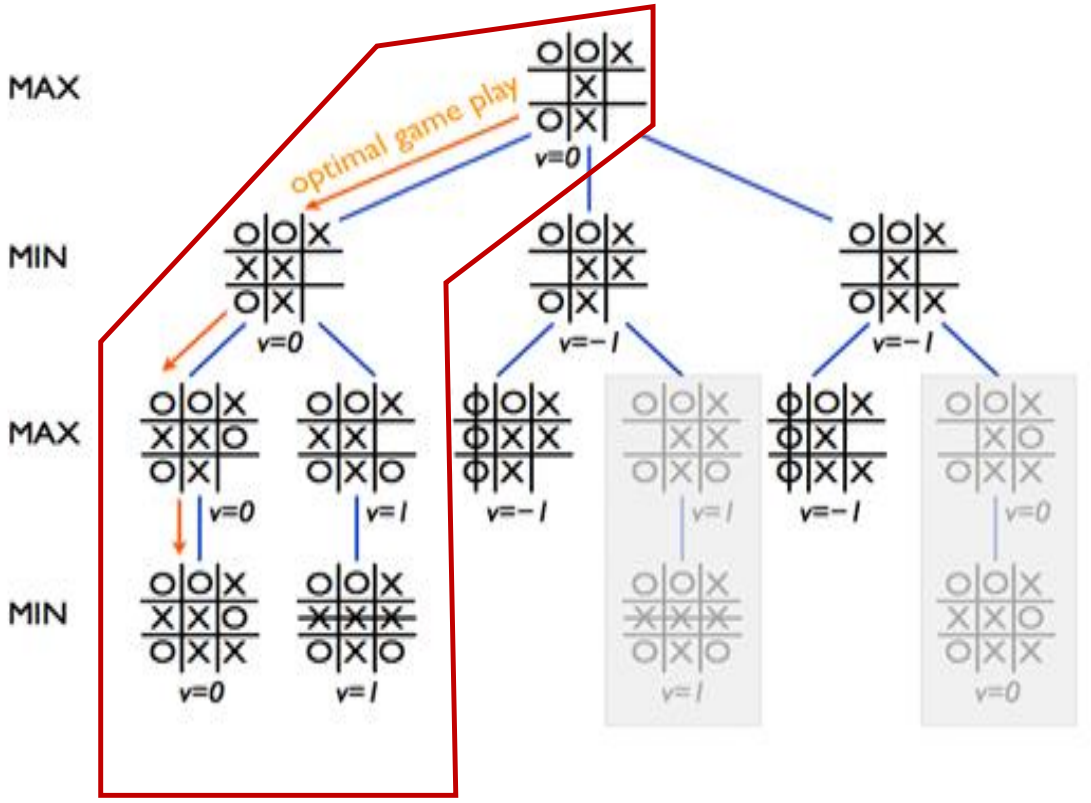
some example

1.

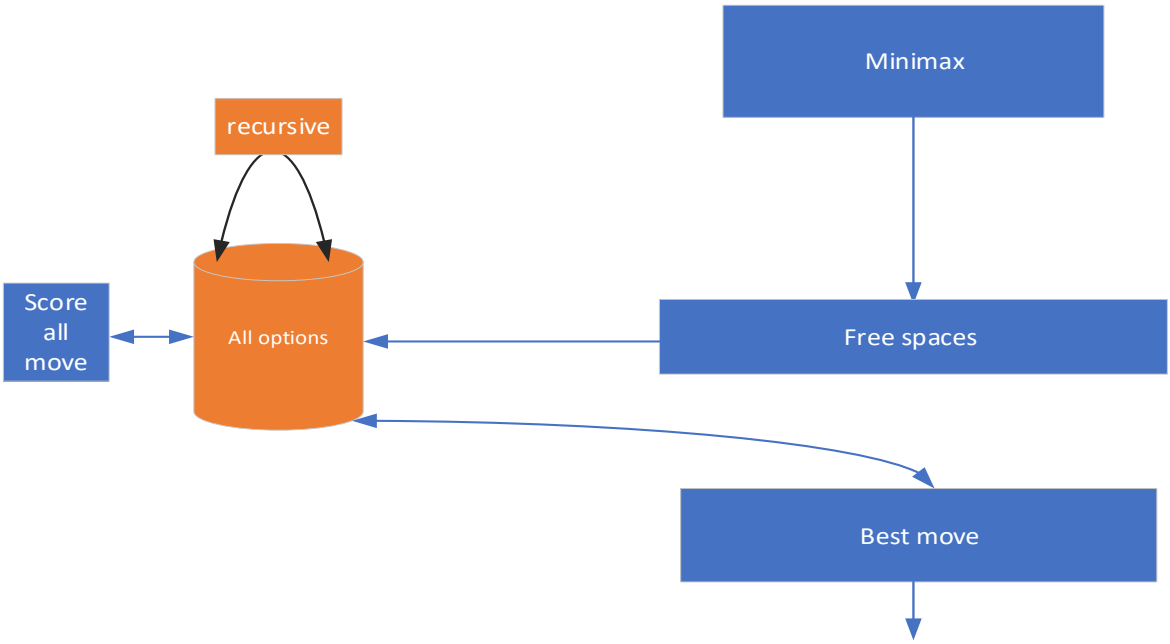


For each recursive call we can see that we followed the algorithm and so we took the good option for the computer perspective

2.



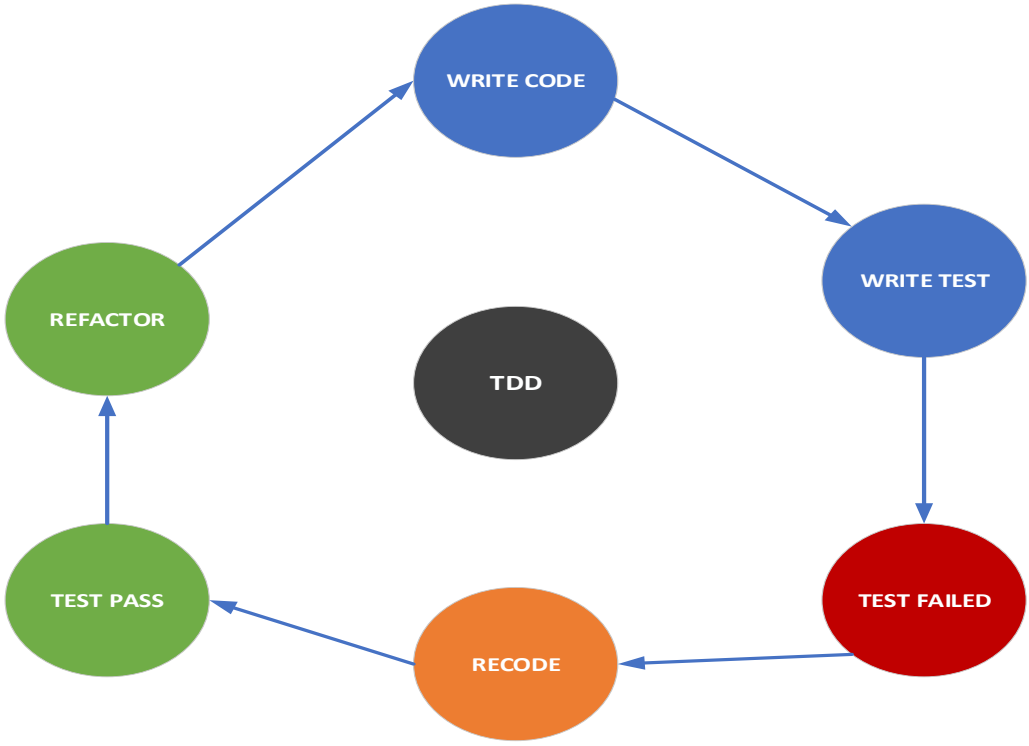
Uml Minimax



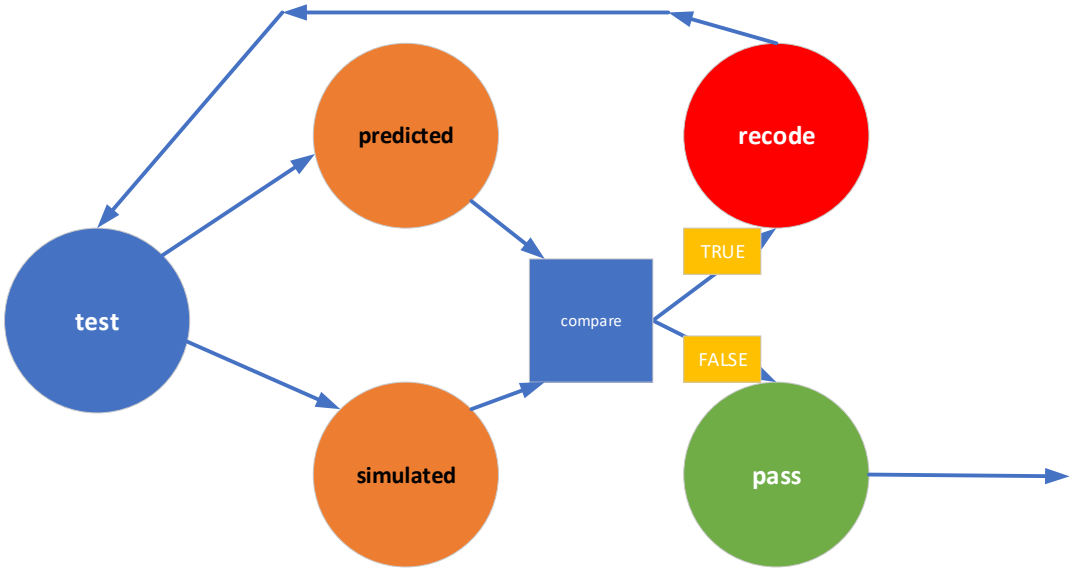
Test Drive Development

The test software tested only for things that can't be checked manually.
The part that no checked- can be checked manually by they display

TDD chart

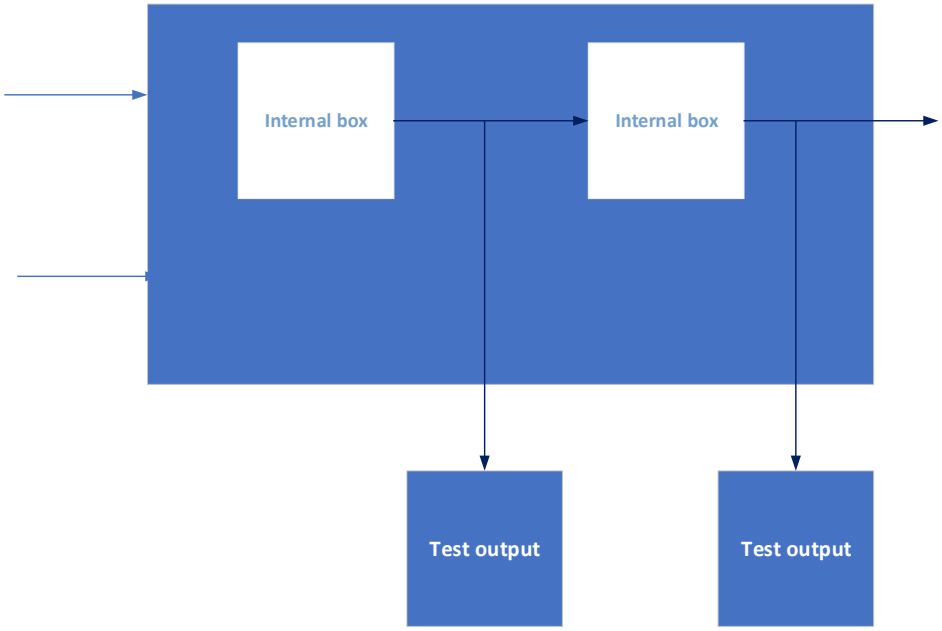


Test chart



“White Box” looking

1. Board properties
2. Player properties
3. Game board draw
4. Game print law
5. Finish function correctness
6. evaluate function correctness
7. avalispots function correctness
8. Minimax function correctness



1.Game properties

We checked if the size of the board is 9 and make sure that the property of Player1 and Player2 is initial correct.

check	predicted	simulated	comment
Length of board	9	9	Need to be length of 9
Player1 sign board	X	X	Sign of player 1
Player2 sign board	O	O	Sign of player 2
Human or AI player 1	Person	Person	Player 1 is the user -a person
Human or AI player 1	Person/ computer	Person / computer	Player 2 is a person or a computer – need to check twice

2.Player properties

By checking the game properties, we can check the Player class properties because Player class in sub class of Game class.

check	**predicted**	**simulated**	**comment**
Player1 sign board	X	X	Sign of player 1
Player2 sign board	O	O	Sign of player 2
Human or AI player 1	Person	Person	Player 1 is the user -a person
Human or AI player 1	Person/ computer	Person / computer	Player 2 is a person or a computer – need to check twice

3.Game board draw

Check by print the board and see if correct

check	**predicted**	**simulated**	**comment**
["O", "O", "O", "O", "O", "O", "O", "O", "O"]	-	-	Full board **Check by print**
["X", "X", "X", " ", " ", " ", " ", " ", " "]	-	-	Half board **Check by print**
[" ", " ", " ", " ", " ", " ", " ", " ", " "]	-	-	Empty board **Check by print**

4.Game print law

check	**predicted**	**simulated**	**comment**
Print law	-	-	**Law Standards Check by print**

5.Finish function correctness

we checked the code by examine the case that can be a winners :
(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6))

First tested for first player with X and after that for second player with sign O.

check	predicted	simulated	comment
[" ", " ", " ", " ", " ", " ", " ", " ", " "]	0	0	Special case
PLAYER 1			
["X", "X", "X", " ", " ", " ", " ", " ", " "]	X	X	Winner player 1
[" ", " ", " ", "X", "X", "X", " ", " ", " "]	X	X	Winner player 1
[" ", " ", " ", " ", " ", " ", "X", "X", "X"]	X	X	Winner player 1
["X", " ", " ", "X", " ", " ", "X", " ", " "]	X	X	Winner player 1
[" ", "X", " ", " ", "X", " ", " ", "X", " "]	X	X	Winner player 1
[" ", " ", "X", " ", " ", "X", " ", " ", "X"]	X	X	Winner player 1
["X", " ", " ", " ", "X", " ", " ", " ", "X"]	X	X	Winner player 1
[" ", " ", "X", " ", "X", " ", "X", " ", " "]	X	X	Winner player 1
PLAYER 2			
["O", "O", "O", " ", " ", " ", " ", " ", " "]	O	O	Winner player 2
[" ", " ", " ", "O", "O", "O", " ", " ", " "]	O	O	Winner player 2
[" ", " ", " ", " ", " ", " ", " ", " ", "O", "O", "O"]	O	O	Winner player 2
["O", " ", " ", "O", " ", " ", "O", " ", " "]	O	O	Winner player 2
[" ", "O", " ", " ", "O", " ", " ", "O", " "]	O	O	Winner player 2
[" ", " ", "O", " ", " ", " ", "O", " ", " ", "O"]	O	O	Winner player 2
["O", " ", " ", " ", "O", " ", " ", " ", "O"]	O	O	Winner player 2
[" ", " ", "O", " ", "O", " ", "O", " ", " "]	O	O	Winner player 2
No winner			
["0", "X","0", "O", "X", "X", "X", "O", "O"]	0	0	No winner
["X", "O","X", "O", "O", "X", "X", "X", "O"]	0	0	No winner
["O", "X","X", "X", "X", "O", "O", "O", "X"]	0	0	No winner
["O", "X", "O", "X", "X", "O", "O", "O", "X"]	0	0	No winner

Given the sentence**" Time-to-market"** - We will stop looking at the other options because they have no pattern or easy to test case.

6.evaluate function correctness

we checked the code by examine the case that can be a winners :

(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6))

First tested for first player with X and after that for second player with sign O.

check	predicted	simulated	comment
[" ", " ", " ", " ", " ", " ", " ", " ", " "]	0	0	Special case
PLAYER 1			
["X", "X", "X", " ", " ", " ", " ", " ", " "]	-1	-1	Winner player 1
[" ", " ", " ", "X", "X", "X", " ", " ", " "]	-1	-1	Winner player 1
[" ", " ", " ", " ", " ", " ", "X", "X", "X"]	-1	-1	Winner player 1
["X", " ", " ", "X", " ", " ", "X", " ", " "]	-1	-1	Winner player 1
[" ", "X", " ", " ", "X", " ", " ", "X", " "]	-1	-1	Winner player 1
[" ", " ", "X", " ", " ", "X", " ", " ", "X"]	-1	-1	Winner player 1
["X", " ", " ", " ", " ", "X", " ", " ", " ", "X"]	-1	-1	Winner player 1
[" ", " ", "X", " ", "X", " ", "X", " ", " "]	-1	-1	Winner player 1
PLAYER 2			
["O", "O", "O", " ", " ", " ", " ", " ", " "]	+1	+1	Winner player 2
[" ", " ", " ", "O", "O", "O", " ", " ", " "]	+1	+1	Winner player 2
[" ", " ", " ", " ", " ", " ", "O", "O", "O"]	+1	+1	Winner player 2
["O", " ", " ", "O", " ", " ", "O", " ", " "]	+1	+1	Winner player 2
[" ", "O", " ", " ", "O", " ", " ", "O", " "]	+1	+1	Winner player 2
[" ", " ", "O", " ", " ", "O", " ", " ", "O"]	+1	+1	Winner player 2
["O", " ", " ", " ", " ", "O", " ", " ", "O"]	+1	+1	Winner player 2
[" ", " ", "O", " ", "O", " ", "O", " ", " "]	+1	+1	Winner player 2
No winner			
["0", "X","0", "O", "X", "X", "X", "O", "O"]	0	0	No winner
["X", "O","X", "O", "O", "X", "X", "X", "O"]	0	0	No winner
["O", "X","X", "X", "X", "O", "O", "O", "X"]	0	0	No winner
["O", "X", "O", "X", "X", "O", "O", "O", "X"]	0	0	No winner

Given the sentence**" Time-to-market"** - We will stop looking at the other options because they have no pattern or easy to test case.

7.avalispots function correctness

check	predicted	simulated	comment
Base case			
[" ", " ", " ", " ", " ", " ", " ", " ", " "]	9	9	-
["X", " ", " ", " ", " ", " ", " ", " ", " "]	8	8	-
["X", "X", " ", " ", " ", " ", " ", " ", " "]	7	7	-
["X", "X", "X", " ", " ", " ", " ", " ", " "]	6	6	-
["X", "X", "X", "X", " ", " ", " ", " ", " "]	5	5	-
["X", "X", "X", "O", "O", " ", " ", " ", " "]	4	4	-
["X", "X", "X", "O", "O", "X", " ", " ", " "]	3	3	-
["X", "X", "X", "O", "O", "O", "X", " ", " "]	2	2	-
["X", "X", "X", "O", "O", "O", "X", "X", " "]	1	1	-
["X", "X", "X", "O", "O", "O", "X", "X", "O"]	0	0	-
Random case			
["O", "X", " ", "X", "O", " ", "X", "O", " "]	3	3	-
["X", "X", "X", "O", " ", "X", " ", "X", "X"]	2	2	-
["O", "X", "O", " ", " ", " ", " ", " ", "O", "O"]	5	5	-
[" ", "X", "X", "O", " ", "O", " ", " ", " "]	5	5	-
["X", "O", "X", "X", "O", "O", " ", " ", " "]	5	5	-
["X", "X", "X", "O", " ", "O", "O", " ", " "]	3	3	-
[" ", "X", "X", "O", "O", "X", " ", " ", " "]	4	4	-
[" ", "X", "X", "O", "O", "O", "X", " ", " "]	3	3	-
[" ", " ", "X", "O", "O", "O", "X", "X", " "]	3	3	-
["X", " ", " ", "O", "O", "O", "X", "X", "O"]	2	2	-

**" Time-to-market"

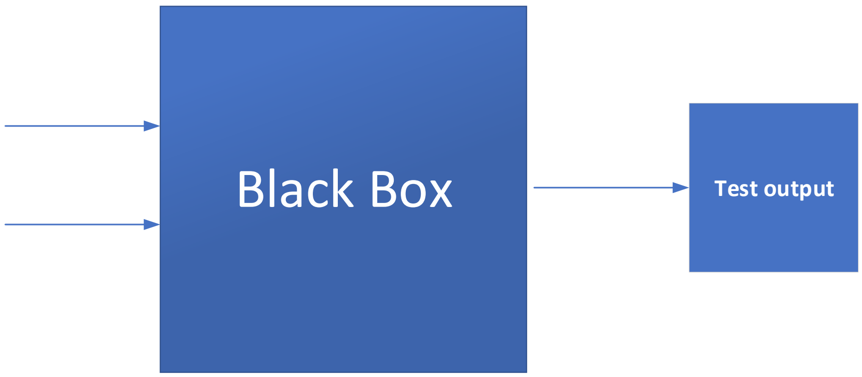
8.Minimax function correctness

First we looked at basic situations and then examined the possibility of the computer to win every position on the board

check	predicted	simulated	comment
Random case			
["X", " ", " ", " ", " ", " ", " ", " ", " "]	4	4	-
[" ", " ", " ", "X", " ", " ", " ", " ", " "]	0	0	-
["X", " ", " ", "X", "O", " ", " ", " ", " "]	6	6	-
["O", "X", "X", " ", "O", "X", " ", " ", " "]	8	8	-
[" ", " ", "X", " ", "O", "X", " ", "X", "O"]	0	0	-
["X", "X", "O", "X", "O", " ", " ", " ", " "]	6	6	-
["O", "X", " ", "X", "X", "O", " ", " ", " "]	7	7	-
["X", "O", "X", " ", "O", " ", " ", "X", " "]	3	3	-
["X", "O", "X", "O", "O", " ", "X", "X", " "]	5	5	-
["O", " ", "X", "O", "X", " ", " ", "X", " "]	6	6	-

**" Time-to-market"

“Black Box” looking



check	**player AI**	**Automatic player** **“Wort case player”**	**comment**
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output
[” ”, ” ”, ” ”,” ”, ” ”, ” ”,” ”, ” ”, ” ”]	Miniamx	Wort case player	Check the correctness of the output

We chose to test the game for an auto player.
The automatic player software is supposed to be the worst player.
We tested the correctness of the game through the atomic test and the computer game with itself.

Difficulties in testing and malfunctions

- I first found that the finish function returns a win even for a situation where there is no Input in the board. The problem fixed by adding an if statement
- Minimax is an AI algorithm assuming that the opponent is also playing optimally. Its was hard to create good situation to check.

Suggestions for improvement

1. Add GUI
2. Add histograms for the automated test and for a collection of user games