

Team Assignment 1: Tic-Tac-Toe in Jason

The goal of this assignment is to implement an agent in Jason that can play the game of Tic-Tac-Toe.

I'm assuming everyone is familiar with this game, but just in case, if you don't know it, see: <https://en.wikipedia.org/wiki/Tic-tac-toe>

This is a team assignment, so it should be done in teams of 3 or 4 students.

This assignment counts for 1/6th of your final mark for this course.

Each team should deliver their agent online (the .asl file) on the Campus Virtual. The deadline for this is Monday 31 March 2025, at 9:00 in the morning.

Furthermore, on Wednesday 2 April, during the class hours (15:00 – 17:00), I will conduct a short interview with each team to ensure that all team members contributed to the assignment and understood it. All team members must be present (except if there is a valid reason why you cannot be there).

After the delivery deadline I will run a small tournament among all the submitted agents.

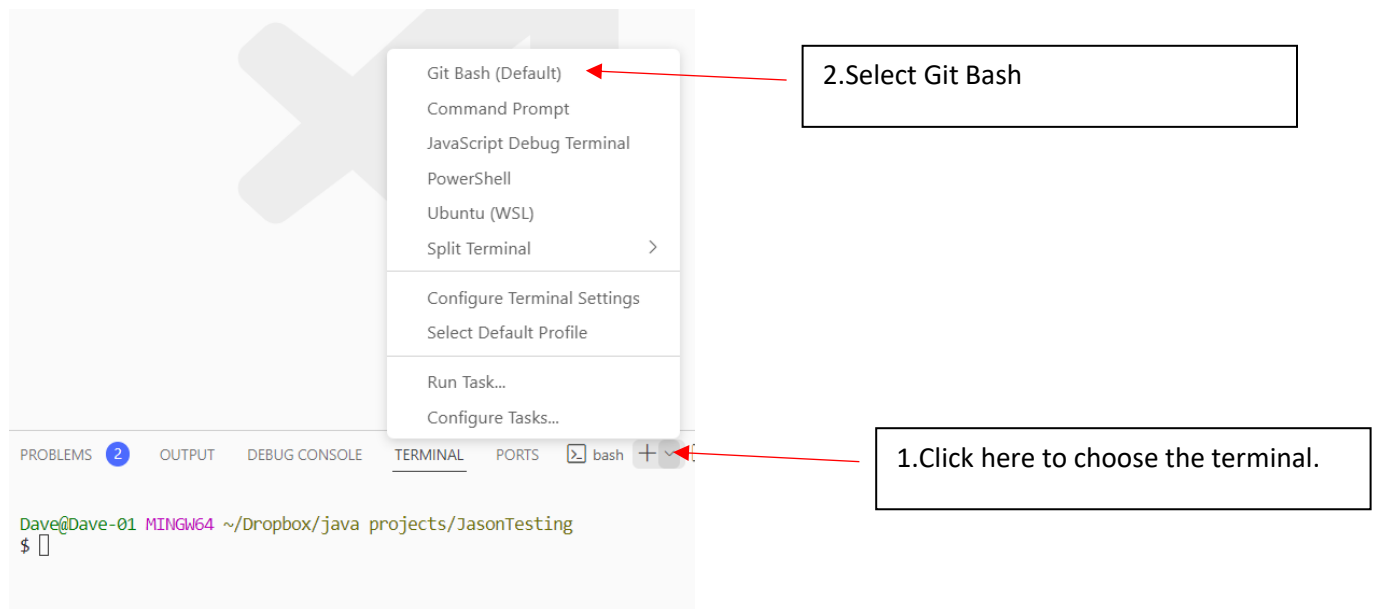
Step 1: install Jason.

To install Jason on your computer, you can follow the following instructions, which explain how to use Jason in combination with the VSCode IDE:

<https://jason-lang.github.io/doc/tutorials/vscode/readme.html>

Note however, that this tutorial forgets to explain one important step: After installing VSCode and Git Bash, when opening a terminal in VSCode, make sure you select the Git Bash terminal and not some other terminal (see image below).

Also, after adding updating your PATH variable, don't forget to restart your computer.



Step 2. Download and Run the Tic-Tac-Toe project:

Go to the Campus Virtual and download the file **tic_tac_toe_assignment.zip**.

Save it somewhere on your computer and unzip it.

Open the unzipped folder in VSCode:

Click on File → Open Folder, and navigate to the unzipped folder.

You can see that this project contains two mas2j files and two agents.

The file **tic_tac_toe_game.mas2j** will run a single game of Tic Tac Toe between two agents.

The file **tic_tac_toe_tournament.mas2j** will run an entire tournament between multiple agents.

To test it, open the GitBash terminal and type: **Jason tic_tac_toe_game.mas2j**

You should now see something like this:

```
MAS Console - tic_tac_toe_game

[SingleGameEnvironment] Action performed: myPlayer play(2,2)
[SingleGameEnvironment]
o _
x _
_ x

[SingleGameEnvironment] Action performed: randomPlayer play(0,2)
[SingleGameEnvironment]
o_o
x _
_ x

[SingleGameEnvironment] Action performed: myPlayer play(2,0)
[SingleGameEnvironment]
o_o
x _
x_x

[SingleGameEnvironment] Action performed: randomPlayer play(2,1)
[SingleGameEnvironment]
o_o
x _
xox

[SingleGameEnvironment] Action performed: myPlayer play(0,1)
[SingleGameEnvironment]
oxo
x

Clean Stop Pause Debug New agent Kill agent REPL agent Sources
```

You can close the window by clicking on the Stop button in the bottom.

Open the source codes of the two agents: **randomPlayer.asl** and **myPlayer.asl**.

The randomPlayer is an agent that just plays random moves. At this point, myPlayer is just an identical copy of randomPlayer. However, the idea is that you are going to adapt the code of this agent to make it better.

Step 3. The Assignment

The assignment is to adapt myPlayer to improve it.

First, change the name of myPlayer to something more unique, and update the mas2j files accordingly.

To understand how the current implementation of myPlayer works, just read the comments in its source file.

For technical reasons your agent must satisfy the following conditions:

- At startup, the agent should perform the 'sayHello' action.
- At the end of each game, it should perform the 'confirmEnd' action.

Note that these conditions are already implemented in the following plans, so just make sure you don't remove them:

```
+started <- sayHello.  
+end <- confirmEnd.
```

Other than that, there are no specific requirements that your agent has to satisfy, except that it should of course play at least somewhat better than the randomPlayer.

While the goal is to make your agent play as well as possible, it should be noted that to determine your grade *it is much more important to how elegantly you have programmed your agent.*

That is:

- Make use of variables and rules to make your code more compact and flexible.
- Make sure that your plans don't include any unnecessary actions.
- Divide your code into multiple plans that are each easy to understand and that have a clear logical function. This will make it much easier to debug your code. Don't put all your code in one single giant plan!
- Make use of achievement goals if that can simplify your plans (but don't use them if it only makes things more complicated).

For example, the following code:

```
isCoordinate(0).  
isCoordinate(1).  
isCoordinate(2).  
isCell(X,Y) :- isCoordinate(X) & isCoordinate(Y).
```

is much more elegant than:

```
isCell(0,0).  
isCell(0,1).  
isCell(0,2).  
isCell(1,0).  
isCell(1,1).  
...
```

In other words, *the winners of the tournament will not necessarily get the highest grades.*

Also, add comments to every rule and plan to make it easier for us to understand your logic. If your code is incomprehensible it may result in a reduction of your grade.

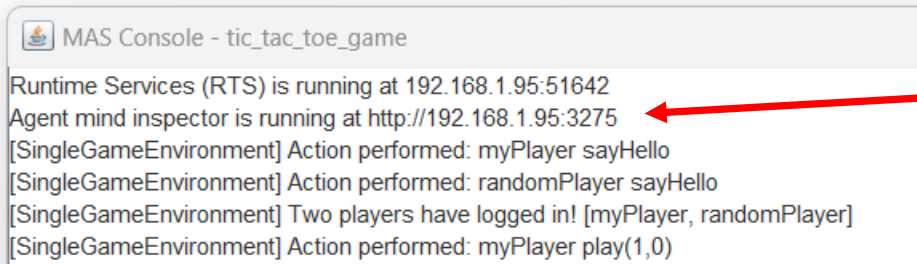
Do not modify the Java files. The environment should be treated as fixed. The goal of this assignment is to implement an agent, given the actions and observations that the environment provides. When delivering the assignment, you should only submit your agent's **.asl** file.

Tips

For quick tests of your agent you can use the **tic_tac_toe_game.mas2j** file, which allows you to run a single game against one opponent.

For more extensive testing you can use the **tic_tac_toe_tournament.mas2j** file. It runs a tournament in which every agent plays a number of games against every other agent. At the end of the tournament, it displays how many points each agent scored (2 points for every victory and 1 point for every draw). This can be useful to assess how well your agent plays over a larger number of games, or it can be used to compare multiple different implementations of your agent by letting them all play against each other and see which one is the best. Please see the comments inside that file to modify it correctly.

For debugging purposes, it can be useful to inspect the belief base of the agent. You can do this by opening a browser and navigating to the IP address indicated at the top of the MAS console:



```
MAS Console - tic_tac_toe_game
Runtime Services (RTS) is running at 192.168.1.95:51642
Agent mind inspector is running at http://192.168.1.95:3275
[SingleGameEnvironment] Action performed: myPlayer sayHello
[SingleGameEnvironment] Action performed: randomPlayer sayHello
[SingleGameEnvironment] Two players have logged in! [myPlayer, randomPlayer]
[SingleGameEnvironment] Action performed: myPlayer play(1,0)
```

You can view your agent's belief base here.

Some ideas to get started:

- Can you ensure that, whenever your agent has a winning move, it will indeed play that move?
- Can you ensure that, whenever your opponent has a winning move in the next round, you play it first to prevent your opponent from winning?