Dynamite!

Rules for Dynamite!

Dynamite! is an explosive variation of Rock, Paper, Scissors, with the added extras of dynamite and water bombs. Dynamite beats rock, paper and scissors. Water bombs beat dynamite, but lose to everything else.

The dynamite is limited to 100 sticks, but you have as many water bombs as you like. To play you must write a bot to decipher your opponents strategy and decide your next move. Each round is worth one point and your bots will play until one player reaches 1000 points. In the case of a tie, no player gets a point and the point for that round rolls over to the following round. A maximum of 2500 rounds are played, at which point the game is considered a draw.

Bots can be written in JavaScript and uploaded as a single .js file, in Java packaged as a single .jar file, or in C# as a .dll library file. See below for further instructions.

Writing a Bot (JavaScript)

A bot must have the function makeMove which reads in the current gameState and decides the next move to be made. The gameState holds all of the previous moves made by each player. Your bot is always player 1. The following code snippet shows the form of the gameState; it is an array containing each player's move for all of the previous rounds.

```
{rounds: [
```

```
{rounds: [
    {
        p1 : "R",
        p2 : "D"
    },
    {
        p1 : "W",
        p2 : "S"
    },
    ...]
}
```

Click here to download a model javascript bot to help get you started. (/rockBot.js)

The bot must be compatible with Node 8.10

Writing a Bot (Java)

A bot must extend the Bot interface (provided as a part of the download below) and override the makeMove method. This method takes a GameState object and returns a Move object. The following code snippet shows an example of how you would return the last move the opposing bot has made. (Note that this would fail in the first round!)

```
public class TestBot implements Bot {
    public Move makeMove(Gamestate gamestat
e) {
        return gamestate.getRounds()[gamest
ate.getRounds().length - 1].getP2();
    }
}
```

Click here to download the java bot interface. (/javarunner-1.0-bot.jar)

Package a jar which contains a single class implementing the Bot interface. (Other supporting classes are fine)

The bot must compatible with Java 8

Other JVM languages

You may use any language that compiles to a suitable JAR - e.g. Scala or Kotlin. For convenience, the standard libraries for **Scala 2.12.6** and **Kotlin 1.2.51** are available to bots - meaning you only need your own classes in the jar.

Writing a Bot (C#)

A bot must implement the IBot interface (provided below) and override the MakeMove method. This method takes a GameState object and returns a Move object. The following code snippet shows an example of a simple bot that always returns paper.

```
using BotInterface.Bot;
using BotInterface.Game;

namespace ExampleBot
{
    public class ExampleBot : IBot
    {
        public Move MakeMove(Gamestate game
state)
        {
            return Move.P;
            }
    }
}
```

Click here to download the c# bot interface. (/BotInterface.dll)

Upload a .dll file which contains a single class implementing the IBot interface. (Other supporting classes are fine)

The bot must be compatible with .NET Core 2.1

Writing a Bot (Python)

The module must contain a class which implements a make_move method. This method takes a GameState object and returns a string corresponding to the move. The following code snippet shows an example of a simple bot that always returns paper.

```
class PaperBot:
    def __init__(self):
        pass

    def make_move(self, gamestate):
        return 'P'
```

Upload a .py file which contains a single class implementing a make_move method. (Other supporting classes are fine)

The gamestate object has the same shape as the JSON above, with Python lists and dictionaries.

Click here to download a model Python bot to help get you started. (/paperbot.py)

The bot must be compatible with Python 3.6

Using the JavaScript CLI

To test your JavaScript bots locally you can use the Dynamite! command line interface. To play a match, run the dynamite-cli.js file and give the relative file paths of the two bots that you would like to play.

```
node dynamite-cli.js  myBot1.js myBot2.js
```

Click here to download the CLI. (/dynamite-cli.js)