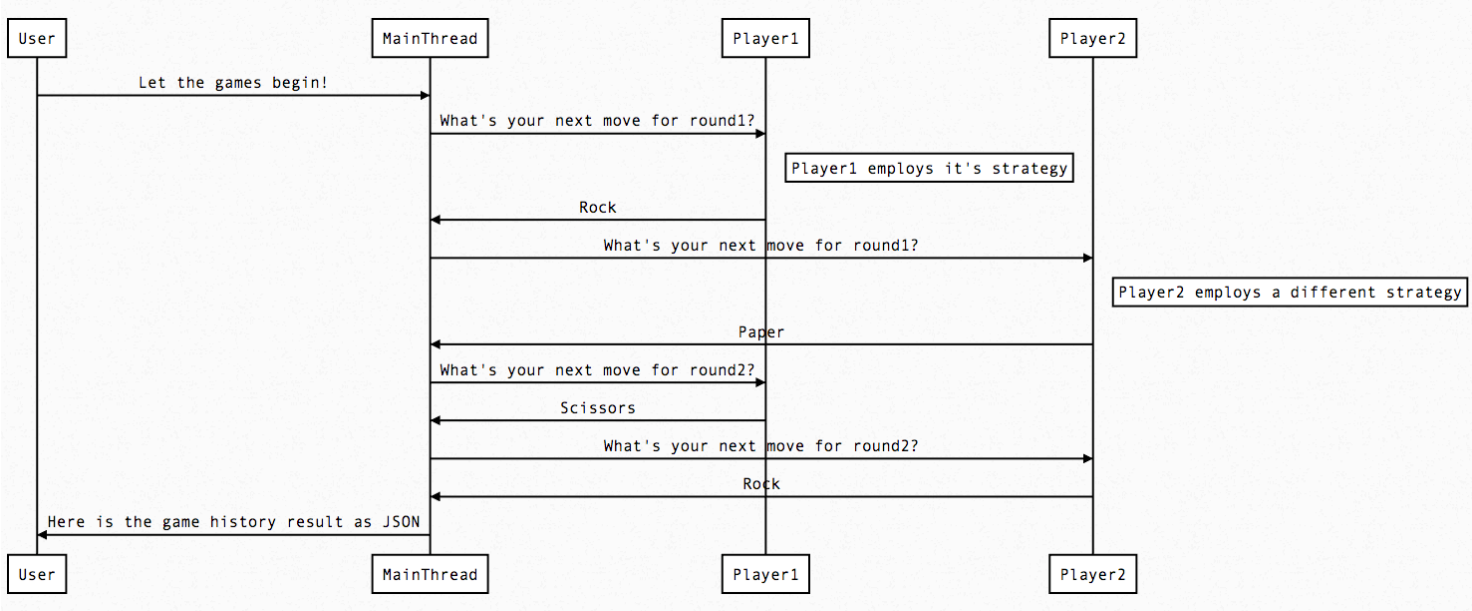
Write a multithreaded program that represents two players playing Rock, Paper, Scissors.

- Each player is modeled as an independently executing thread
- There is a main thread which controls the game and gets the next move from each player thread
- The main thread produces a JSON file that represents the game history

The final output should be a JSON file that contains the complete game history of all game rounds since the start of the program:

```
[
    {
        "Round": 1,
        "Winner": "Player2",
        "Inputs": {
            "Player1": "rock",
            "Player2": "paper"
        }
    },
    {
        "Round": 2,
        "Winner": null,
        "Inputs": {
            "Player1": "scissors",
            "Player2": "scissors"
        }
    },
    ...
    {
        "Round": 100,
        "Winner": "Player1",
        "Inputs": {
            "Player1": "scissors",
            "Player2": "paper"
        }
    },
]
```

Interaction



interaction

Components

The program must have the following:

1. A thread for Player1 with the following properties
    1. It has the ability to receive a request from another thread (in this case, the main thread) to return it's next move.
    2. The input it receives should include the full history of the previous rounds with Player2, which should look like the above example
    3. It can employ any strategy to generate it's move, including choosing a random move. It's up to you decide what strategy to use.
2. A thread for Player2 with the following properties
    1. It's identical to Player2, except it employs a different strategy for generating it's move. Player2's strategy should be based on the history of Player1's previous moves. For example, randomly choosing one of Player1's moves and using that as it's next move. Or, using Player1's last move as it's next move.
3. A main thread that does the following:
    1. Start the Player1 and Player2 threads
    2. Repeat the following steps 100 times:
        1. Ask Player1 and Player2 for their next move each, passing in the full history of previous rounds (initially empty)
        2. Determine the winner for this round (if any)
        3. Store the results in memory, which will be passed to the players as inputs on subsequent rounds

3. Emit the results in a file called `result.json` and output a message to the console with the full path to the result file.
4. Stop the Player1 and Player2 threads

Testing

Write a test that exercises the code and validates that the JSON output is valid and contains 100 game rounds

Objectives

The purpose of this challenge:

- Demonstrate that you understand the concepts of multi-threaded programming
- Can write clean, well factored, well named, easy to understand code
- Can understand the requirements and produce a result that satisfies them

What's *not* in scope:

- Performance optimization
- Catching every single corner case

Rules

- You may use any built-in or 3rd party libraries
- You may not ask specific questions on Stack Overflow or to anyone you know