

PRACTICING INTERFACES

Our goal is to practice interfaces and also to think a little bit about Artificial Intelligence! We want to be able to come up with different strategies for playing Rock-Paper-Scissors and then run them against each other to see which strategy is the best.

1. Your TA will go over how to play the game Rock-Paper-Scissors using a couple of volunteers from class.
2. Check out the project titled "131Fall18Lab24".
3. Look at the class called Move. It's a bit odd. Note that the constructor is private! There are exactly THREE Move objects in the world at all times:

Move.ROCK, Move.Paper, Move.Scissors

4. Look at the interface called RPSStrategy, which has just two method prototypes:

```
public Move getFirstMove();  
public Move getNextMove(Move opponentPreviousMove);
```

The first method will decide which of the three moves to use in the very beginning. (It will return either Move.ROCK, Move.PAPER, or Move.SCISSORS).

The idea for the second method is that you pass to it the previous move of the opponent, and it will somehow compute what move to make next (either Move.ROCK, Move.PAPER, or Move.SCISSORS)

5. Look over the (stupid) strategies that are provided:

RockAlwaysStrategy
RandomMovesStrategy
MimicOpponentsPreviousMoveStrategy

6. Take a look at the PlayGame class. Note that there is a main method that will instantiate two strategies and have them play against each other a whole bunch of times. Run it!

7. Now comes the fun.... write a few strategy classes of your own.

Call the classes whatever you want, just make sure they implement the RPSStrategy interface.

Edit the main method in the PlayGame class to see how your strategies do against each other, or against one of the (stupid) strategies that I have provided!

HINT: You may want to use arrays as instance variables to store all of the moves that have been made historically by yourself and/or your opponent. The arrays would start off size 0, and then each time the method is called an entry would be added to the arrays. Recall that "resizing" an array amounts to making a NEW array that is one unit larger, copying all of the existing data, etc.