Homework 3

In this assignment, your task is to write a program that excels at the game of Rock-Paper-Scissors (RPS) against opponents playing non-equilibrium strategies. If you are unfamiliar with the game, a good place to start is the Wikipedia page (which among other things, has an extensive section on international RPS tournaments):

`http://en.wikipedia.org/wiki/Rock-paper-scissors`

As discussed in class, a Nash equilibrium for a two-player game is a pair of strategies (π_1, π_2) for the two players such that no player can improve their payoff by unilaterally changing their own strategy. In Rock-Paper-Scissors, the optimal strategy for both players is the mixed strategy $(1/3, 1/3, 1/3)$, i.e., to choose rock, paper, and scissors each with probability 1/3 on each round. However, there is an important caveat: *a Nash equilibrium strategy is best for you only if your opponent is also employing their Nash equilibrium strategy.* In many cases, however, your opponent may not be using their Nash equilibrium strategy, in which case you may be able to do much better than simply playing your own Nash equilibrium strategy. For example, the $(1/3, 1/3, 1/3)$ mixed strategy is provably the best response to an opponent also playing $(1/3, 1/3, 1/3)$. However, what if we're up against someone playing the pure strategy $(1, 0, 0)$ (i.e., always throw rock)? You could still employ your Nash equilibrium strategy — you would win 33% of your games (when you happened to throw paper), tie 33%, and lose the other 33% (when you happened to throw scissors). But there's a better counter-strategy to employ here, namely $(0, 0, 1)$ (i.e., always throw paper) that guarantees a 100% win rate.

A Nash equilibrium strategy focuses on minimizing exploitability — indeed, the Nash equilibrium strategy for RPS guarantees that you will never lose more than 33% of the time, no matter the opponent's strategy. But in many games, this needs to be balanced against effectively exploiting your opponents' non-optimal behaviors. But why might your opponents act non-optimally? They might be trying to probe *you* for weaknesses and trying to forecast your actions! Or, as is the case with human players, they may not be very good at acting truly randomly. Your goal on this assignment is to devise strategies for RPS that attempt to model their opponent, so as to exploit them, while remaining unexploitable themselves. A portion of your grade on this project (5%) will be based on the performance of your RPS bot in a round-robin tournament against bots designed by your classmates.

All of your bot entries must be written in the Java programming language!

Getting Started

Start by downloading the code archive on Moodle. The `Tournament` class runs an RPS tournament between two named bots that lasts a specified number of matches. All the tournament parameters can be specified on the command line. Every bot that you design must implement the `RoShamBot` interface. For example bot implementations, refer to `ApeBot`, `NashBot`, `MixedBot`, and `SolidAsARockBot`. To run a tournament between two bots, say `MixedBot` and `ApeBot`, first compile the two corresponding classes. Then, run the `Tournament` class as follows:

```
java Tournament MixedBot ApeBot 500
```

You should feel free to modify the `Tournament` class in any way that eases the experimentation and data collection process. There are two main requirements for the bots that you submit as your tournament entries:

- They *must* implement the `RoShamBot` interface.
- They must operate within the stipulated time and space constraints — a 10000-round match between your entry and `NashBot` should complete within 60 seconds using no more than 512MB of heap space on a standard campus Mac computer.

Timeline

Note that there are two deadlines for this assignment:

- **5pm, Monday, February 23**: This will be a "warm-up" event. Every team should submit between 1 and 3 bots to participate in a mock tournament. The results will be made available to everyone to guide future development.
- **11:55pm, Friday, February 27**: This is the deadline for submitting your final tournament entry and the associated write-up. **You may only submit one bot for the final tournament!**

The Write-Up

Remember the overarching writing rule for this course: *you need to be sufficiently precise with your writing and include enough details that a competent reader could reproduce your results.* Here are some specific things to address in your write-up, in no particular order. This is *not* meant to be an exhaustive list.

- Describe the algorithm behind the entry to the class tournament. What approach did you use to model your opponent?
- Were there other promising algorithms that you developed as well? How did you choose which algorithm would be your final entry to the class tournament? You should justify your choices with data.
- As always, cite your sources if you sought inspiration from the literature (*not* Wikipedia!).

Deliverables

As stated earlier, you should submit the bot that is your tournament entry by the deadline via Moodle. In addition, you should also submit a write-up composed in LaTeX, formatted using the AAAI template. The write-up should be no more than six pages in length, including references. Electronic submissions of your write-up (via Moodle) are acceptable.