In this assignment, your task is to write a program that excels at a multi-round version of the game of Rock-Paper-Scissors-Lizard-Spock (RPSLS) against opponents playing non-equilibrium strategies. This is a generalization of the standard Rock-Paper-Scissors game, with two additional actions, which ensures a lower probability of tied outcomes. You can learn about the rules of this game (and play against others) at the following page:

`https://rpsls.net`

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 RPSLS, the optimal strategy for both players is the mixed strategy $(1/5, 1/5, 1/5, 1/5, 1/5)$, i.e., to choose rock, paper, scissors, lizard and Spock each with probability 1/5. 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 $(1/5, 1/5, 1/5, 1/5, 1/5)$ mixed strategy is provably the best response to an opponent also playing $(1/5, 1/5, 1/5, 1/5, 1/5)$. However, what if we're up against someone playing the pure strategy $(1, 0, 0, 0, 0)$ (i.e., always throw rock)? You could still employ your Nash equilibrium strategy — you would win 40% of your games (when you happened to throw paper or Spock), tie 20%, and lose the other 40% (when you happened to throw scissors or lizard). But there's a better counter-strategy to employ here, namely $(0, 0, 1, 0, 0)$ (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 RPSLS guarantees that you will never lose more than 40% 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 RPSLS 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 RPSLS bot in a round-robin tournament against bots designed by your classmates. Bots that employ novel and well-motivated strategies, but that do not fare well in the tournament, will also receive full-credit on this portion to encourage intellectual risk-taking.

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 RPSLS 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 CSC 370 Jupyter instance.

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!).

Deadlines

Note that there are two deadlines for this assignment:

- **11:55pm, Monday, April 4**: Optionally enter up to three of your bots for a "warm-up" class-wide tournament. Upload your bot entries via Moodle. The results will be made available to everyone to guide future development and/or to include in your paper.
- **11:55pm, Monday, April 11**: This is the deadline for submitting your final tournament entry and the associated write-up (LaTeXsource and the compiled PDF). **You may only submit one bot for the final tournament!**

Recommended Timeline

Here's a recommendation for how to budget your time over the next couple of weeks as you work on this assignment.

- **Mar. 24–27:** Read the problem description, look over the provided code base, implement some simple bots and run some preliminary experiments. Try designing your own approaches for modeling opponents. How do these bots do? Write your *Introduction* section.
- **Mar. 28–Apr. 1:** Do background research and see what strategies have been studied for this game in the past. Also consider algorithmic approaches that have been used in other iterated games (for example, iterated Prisoner's Dilemma). Implement one or more of these and run some more experiments. Write your *Background* section.
- **Apr. 2–3:** Finalize your entry (or entries) to the warm-up tournament. Write your *Experiments* section.
- **Apr. 4–8:** Continue running experiments as necessary, using the results from the warm-up tournament to guide further development. Write your *Results* and *Conclusions* sections.
- **Apr. 9–11:** Wrap-up any pending experiments, write the the abstract, revise and proofread the entire paper and submit your final draft, as well as your entry to the final tournament.