LaSalle College
Felix Soumpholphakdy

March 21, 2019
Advanced Data Structure: 420-J13-AS

# Project Season 4

## Overview

For your final project, you will design and implement artificial intelligence (AI) algorithms to play simple games. These algorithms will be tested and scored against dummy algorithms and also submissions from other students. Your AI will be simple at first, but at every iteration it must improve and adapt to new requirements and harder opponents.

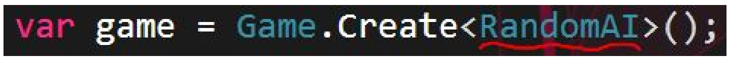
The final season aims at designing a strong AI that is able to beat creative opponents. Students will learn about Markov chains as a predictive strategy. Students that wish to further challenge themselves can attempt to beat other students at the leaderboard to obtain bonus points.

## Grade Distribution

| | |
|---|---:|
| Game 1 (RPSLS) | 10% |
| Game 2 (to be announced) | 10% |
| Report | 10% |
| **Total** | **30%** |

## Season 4 Submission Due March 29 For Bonus

## Instructions

1. Clone the project at https://github.com/felixsoum/420J13AS-RPSLS
2. Find the file that is assigned to you. It will be located in the path: `\RPSLS\AI\S{section number}\{4 letters of permanent code}.cs`
3. Fix any problems from the previous week.
4. Override and implement the *Observe()* method which allows your AI to observe your opponent's previous move.
5. Update your *Play()* method to take these observations into account to beat the AI challenges.
6. Setup your AI by going to Program.cs and substituting *RandomAI* at line 15 by your class, then running the program (Ctrl + F5).

```
var game = Game.Create<RandomAI>();
```
7. Submit only your class file to Lea.

## Rules

The most important rule is to not plagiarize. It is recommended and good to work with classmates to discuss strategies and practice against each other, but copy-pasting someone else's work is plagiarism. As computer scientists, we have designed very good tools for detecting this in code, even if the variables are renamed or statements are re-ordered. You might even learn some algorithms for that in this course. Basically, do not do that or you will strongly penalized.

Additionally, you cannot depend on other student AI classes (do not try to invoke another AI's methods). You will also be penalized if you do that.

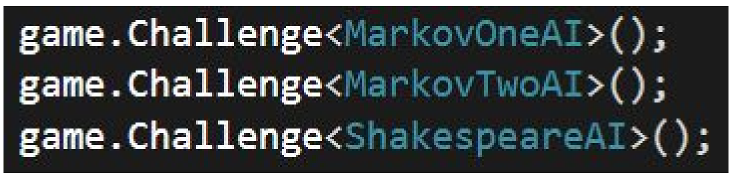
The following rules are guidelines for the tournament. If you do not follow them, you will be disqualified, but you will not be penalized for the project grades.

- New rules will only be added after a tournament has finished and unwanted behaviors have been detected
- Playing an illegal move will result in losing the round
- If you need a new *Random* object, you are only allowed to use *Game.SeededRandom*, which is seeded because the game results must be reproducible (deterministic).
- Each of your AI's turns must take less than 10 milliseconds to execute. One turn consists of one call to *Observe()* then *Play()*. If your AI takes too long, then the system will tell you in the console. This time limit is subject to change.
- You are not allowed to use any classes from the *System.Reflection* namespace, because you do not need them and if you do then I am suspicious of your code.
- Trying to instantiate any AI yourself will throw an unauthorized exception.

## Challenges

Each challenge puts your AI against a specific dummy AI. One challenge is composed of 20 battles, and each battle is composed of 100 rounds of RPSLS. To pass the challenge, your AI must defeat the dummy in all battles, but not necessarily all rounds. A battle is won by the AI that has won the most rounds in it. Between each battle, the random seed will change. The initial seed will also be changed before evaluating your AIs. This makes it impossible for any AI to pass a challenge due to luck or hard coding all moves.

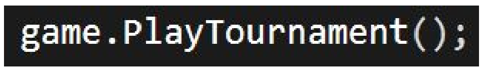
You can toggle challenges simply by uncommenting or commenting the code (Program.cs line 17-19). You can also run multiple challenges sequentially.



```
game.Challenge<MarkovOneAI>();
game.Challenge<MarkovTwoAI>();
game.Challenge<ShakespeareAI>();
```

## Leaderboard

A tournament is implemented to encourage students to outperform themselves. The best students of each section will have a chance to informally present their AI to the class to receive bonus points. Feel free to try the tournament yourself by running the *game.PlayTournament()* statement in Program.cs line 21. Note that if you want the exact same result as the real tournament, then make sure to disable (comment) the challenges, because they affect the outcome through the use of the SeededRandom object. The seed will be taken from the weekly lotto numbers.



```
game.PlayTournament();
```

## Season 4 Grading Detail

**[1 point]** For passing the challenge against *MarkovOneAI*. For every move *x*, there exists a move *y* that has a high probability (60%) of being played right after *x*. For example, after playing a Rock the AI could favor playing Paper most of the times.

**[1 point]** For passing the challenge against *MarkovTwoAI*. For every move *x* and *y*, there exists a move *z* that has a high probability (60%) of being played right after *x* then *y*. For example, after playing a Rock, then Paper, the AI could favor playing Scissors most of the times. But it’s possible that after playing Spock, then Paper, the AI could instead favor playing Lizard.

**[1 point]** For passing the challenge against *ShakespeareAI*. This AI converts Shakespeare’s SonnetXVII into a sequence of moves. This sequence of moves is available through the static method ShakespeareAI.CreateSequence(). Then when instantiated, the AI will pick a random index and play moves sequentially starting at that index. Note that Shakespeare has a bad memory, so there is a 50% chance that instead of playing the correct move, he will be a play a random move instead.