Programming Assignment #6

Overview:

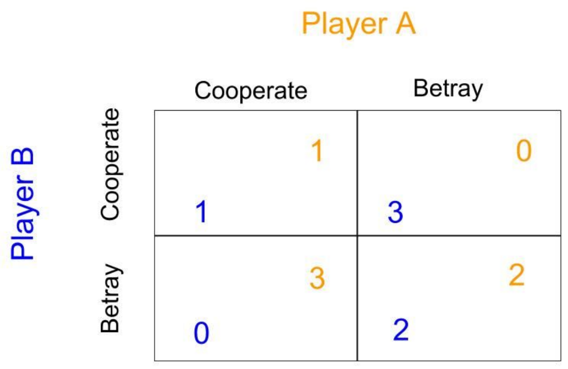
In programming assignment #6 you will be implementing bots for two different games: *the prisoner's dilemma* and *the congestion game*. Your bots will make decisions about how to play on the basis of what they've seen in the past. Unlike other programming assignments where you implemented the same program as your classmates, in this assignment you will have the freedom to have your bot make decisions in whichever way you like. The goal will be for your bot to win one of the two games in *The Grand Tournament.* Dr. Holler will be giving out prizes to winners as well as special awards.

Game Details

Game 1: Prisoner's Dilemma. The prisoner's dilemma is a classic game in Game Theory. It goes like this:

Two bank robbers, Kate (A) and Jack (B), have been arrested and are being interrogated in separate rooms. The authorities have no other witnesses, and can only prove the case against them if they can convince at least one of the robbers to betray his accomplice and testify against them. Each bank robber is faced with the choice to cooperate with his accomplice and remain silent or to defect and testify for the prosecution. If they both cooperate and remain silent, then the authorities will only be able to convict them on a lesser charge of loitering, which will mean one year in jail each. If one testifies and the other does not, then the one who testifies will go free and the other will get 3 years in jail. However, if both testify against the other, each will get two years in jail for being partly responsible for the robbery.

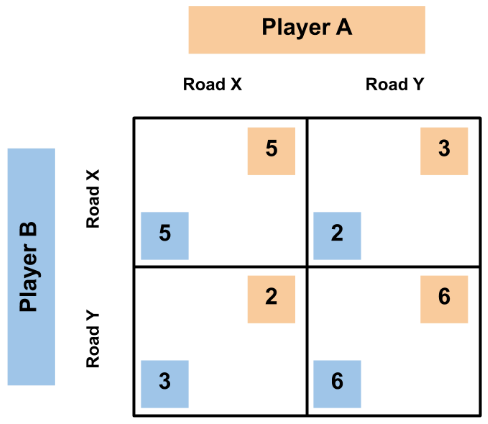
The prisoner's dilemma can be summarized in the following diagram, which is called a *bimatrix*



Game 2: Congestion game. There are actually a whole bunch of "congestion games" in game theory. You will be playing a particularly simple one with only two players and two roads. Congestion games are closely related to the topic of Dr. Holler's thesis. The game goes like this:

*There are two cars (A and B). They are in the same starting location and need to drive to the same destination point. They have two choices of roads to take: road X and road Y. If only one car takes road X, it will take 2 minutes to reach the destination. If only one car takes road Y, it will take 3 minutes to reach the destination. However, if **both** cars take road X, they will each take 5 minutes to reach the destination due to increased traffic. Similarly, if both cars take road Y, they will each take 6 minutes to reach the destination due to increased traffic.*

The congestion game can be summarized with the following bimatrix



Note that for both the prisoner's dilemma and the congestion game, the goal is to get the **lowest score.**

Details of the *The Grand Tournament*:

We will hold a tournament where your bots will compete against each other. The tournament will consist of a prisoner's dilemma bracket and a congestion game bracket. For each bracket, you will submit one competitor, named *PDBotLastName* for the prisoner's dilemma and *CongBotLastName* for the congestion game. In both brackets, you will play a match against every other student in the class. A match consists of 300 rounds of play in which you repeatedly play against the same opponent. Your bot may be designed in a way that adapts over time based on what it has seen the opponent do in the past. It may also include behavior that uses randomness. At the end of the tournament, the score from all of your matches will be added up (separately for the prisoner's dilemma bracket and congestion game bracket), and the students with the lowest scores for each bracket will be crowned victorious.

Implementation details:

To receive full credit for this programming assignment, you must do the following:

1. Create **at least four bots for each game**. These bots can be as simple or complex as you like.
2. Test your bots against each other using the *PDTournament* and *CongTournament* classes that I provide. Write a summary (½ to 1 page) reflecting on anything you noticed or learned while having your bots compete against each other. Name your summary *reflections.txt*, you can create and write this document directly in Eclipse. More details on how to use the *PDTournament* and *CongTournament*
3. Submit **only two** of your bots to be used in the tournament. You must give those bots special names: *PDBotLastName* for the prisoner's dilemma and *CongBotLastName* for the congestion game. For example, Dr. Holler's prisoner's dilemma bot would be called *PDBotHoller*.

All of your bots will implement two methods:

Method # 1

```
public boolean play(boolean a1, boolean b1, boolean a2, boolean b2,
boolean a3, boolean b3, double frac, int round,  int totalRound).
```

boolean a1: your action in the last round
boolean b1: your opponent's action in the last round
boolean a2: your action 2 rounds ago
boolean b2: your opponent's action 2 rounds ago
boolean a3: your action 3 rounds ago
boolean b3: your opponent's action 3 rounds ago
double frac: the fraction of the time your opponent has cooperated in the past in the case of the Prisoner's dilemma, and the fraction of the time your opponent has taken road X in the case of the congestion game.
int round: the current round, indexed starting at zero
int totalRound: the total number of rounds to be played in the match

In the Prisoner's dilemma, **true** corresponds to **cooperate**. In the Congestion game, **true** corresponds to **road X**.

Notice that in the first round of play, there are no previous actions. In this case, all of the boolean actions will evaluate to **true** by default**.** In the second round, only the actions *a1* and *b1* will correspond to actual previous actions. In this case, the other actions will still evaluate to true by default. From the 4th round and onwards, the booleans *ai* and *bi* will correspond to actual actions.

Method #2

public String toString()

This method simply returns a string that identifies your bot with a name. Make your names descriptive. You can see a sample of this method in the **PDBot** class provided.

Dr. Holler's code

I'm providing you with 4 classes that you need to include in your project. They are detailed below.

PDBot. This is a very simple Bot that can play the prisoner's dilemma. Any bot you create must "extend" this bot to be compatible with my code. Don't worry about what extending really means. All you need to know is that, when you create your bot, your code must look like this:

```
      public class PDBotLastName extends PDBot {
}
```

CongBot. Just like PDBot but is used for congestion game bots.

PDTournament. This is the class I will be using to run our tournament, but you can also use it for testing your bots against each other. If you have two bots *bot1* and *bot2* that you would like to compete in a match that lasts for 100 rounds, you can construct a PDTournament object as follows:

```
      PDTournament myPDTournament = new PDTournament(100, bot1,
bot2);
```

Then, when you want to run the match, you can call the line

```
      myPDTournament.runTournament();
```

The resulting scores will be printed in the console.

CongTournament. Works in the same way as PDTournament, but for congestion game bots.

Comment: This was an in class project (though students were allowed to work on it outside of class as well if they desired, and some poured a lot of time into it) that we did in AP CSA. It turned out to be one of the highlights of my year, and I think most if not all students really enjoyed it. The “tournament” structure was really fun, and the strategy aspect gave students a natural motivation to look at each other’s code. I was intentionally nonspecific in suggested strategy, and in the end it paid off by being able to showcase a number of diverse student-designed strategies. Also, having it in a tournament structure required me to write the code that supported having bots battle. This gave me an opportunity to highlight some programming techniques. We capped off this project with an awards day, where I included awards for the tournament winners as well as special/goofy awards like “nicest driver” and “meanest prisoner”.

With all that being said, there were a number of aspects of this project that weren’t very smooth. Some students had difficulty importing my code and using it to test their bots. Some students collaborated a lot with each other, but others focused only on their code and didn’t get as much out of the experience. I’d like to think about what structures I can put into this project, while keeping it open ended, to increase student collaboration.