Radiolab - Prisoner's Dilemma

Listen to first 10 minutes in class.

Two members of a criminal gang are arrested and imprisoned. Each prisoner is in solitary confinement with no means of communicating with the other. The prosecutors lack sufficient evidence to convict the pair on the principal charge, but they have enough to convict both on a lesser charge. Simultaneously, the prosecutors offer each prisoner a bargain. Each prisoner is given the opportunity either to betray the other by testifying that the other committed the crime, or to cooperate with the other by remaining silent. The offer is:

> If A and B each betray the other, each of them serves two years in prison

> If A betrays B but B remains silent, A will be set free and B will serve three years in prison (and vice versa)

> If A and B both remain silent, both of them will serve only one year in prison (on the lesser charge).

What are some possible strategies to play? (playing each team 200 times in a round-robin)

You and your team will create a class to compete in the Tournament.

All the team's classes will compete in a round-robin tournament and play every other team 200 times.

Here is the scoring YOU vs. OTHER TEAM

```
public static final double COOPERATE_COOPERATE = 3;
public static final double COOPERATE_DEFECT = 0;
public static final double DEFECT_COOPERATE = 5;
public static final double DEFECT_DEFECT = 1;
```

Your class will implement an interface called Entry.

```
public interface Entry {

    /* Returns this entry's move for the given round.
     * @param round the integer round number
     * @return the Move made for this round by this entry*/
    public Move getMove(int round);

    /* Sets the moves made for the given round by this entry and the opponent.
     *
     * @param round the integer round number
     * @param yourMove the Move this entry  made for this round
     * @param opponentMove the Move the opponent made for this round*/
    public void setResult(int round, Move yourMove, Move opponentMove);
}
```

getMove(...) returns your Move -> either Move.DEFECT or Move.COOPERATE

setResult(int r, Move you, Move opp) -> called after your program and opponents program have made a move and it tells you what each of your moves were for that round. Your program can keep track of all the moves, some of the moves, ... and set instance variable(s) appropriately to have a strategy in your getMove method.

```
public enum Move {
   DEFECT,
   COOPERATE
}
```

The Director program plays two programs against each other (and eventually all teams agains all other teams in a round robin).

It first calls getMove(int round) from each program to get the move of each.

It then calls setResult(int round, Move yourMove, Move opponentMove) to tell each program what just happened.

This repeats 200 times with the Director keeping track of each program's score.

Get into teams of 2 - 4 people and discuss strategy for your entry in tournament.

Once you have a strategy for your team, write a class that implements the Entry interface. If your team is stuck on how to implement your strategy, see me!

You will get the following classes:

Move.java, Entry.java, ScoreConstant.java

```
public enum Move {
    DEFECT,
    COOPERATE
}
```

```
public class ScoreConstant {

    public static final double COOPERATE_COOPERATE = 3;
    public static final double COOPERATE_DEFECT = 0;
    public static final double DEFECT_COOPERATE = 5;
    public static final double DEFECT_DEFECT = 1;
}
```