TDT4280 Assignment 1: Game Theory

Practical Information

- Delivery:
  - A pdf, ps or txt file with a solution for task 1 and a description of your system and your score table for task 2.
  - A zip-file named YourLastName(s).zip with all the .java and .class files for this exercise.
- You can deliver in pairs.
- Please use English in your report and code comments.

Task 1

Camomilla and Daisy are competing sellers. Their price competition can be described by the following game where the game's payoffs represent the sellers' daily profits in thousands of dollars.

		Daisy							
		$1		$2		$3		$4	
Camomilla	$1	2.5	2.5	5	0	5	0	5	0
	$2	0	5	4	4	8	0	8	0
	$3	0	5	0	8	4.5	4.5	9	0
	$4	0	5	0	8	0	9	4	4

1. Is Camomilla's strategy $4 strongly dominated (yes/no)? If so, indicate which strategy(s) strongly dominate(s) $4.
2. Is Camomill's strategy $4 weakly dominated (yes/no)? If so, indicate which strategy(s) weakly dominate(s) it.
3. Are any of Camomilla's strategies undominated (yes/no)? If so, which one(s)?
4. Under Iterated Elimination of dominated Strategies, does the game have a solution (yes/no)? What is the solution? Write down the sequence of eliminations.

Task 2

This assignment is similar to Axelrod's tournament of iterated prisoner's dilemma. Axelrod arranged this tournament while he was studying how cooperation can be evolved. Axelrod's main field is political science and he used the "evolution of cooperation" in political science but also in many other disciplines, among others, for understanding how cancer cells evolve cancer tumors (you may want to see the references at the end of this text).

In this task, you will compare different playing strategies to see their ability to promote cooperation when they play an unknown (to the players) number of rounds (N).

You will arrange a tournament where each player plays with 5 other players. Therefore, there will be 6 players each of which with a different playing strategy. The strategies are:

1. The agent plays always cooperatively
2. The agent plays always defect

3. TIT-for-TAT (see the text book)
4. TIT-for-every-other-TAT
5. A mixture of two of the strategies above (using any definition of mixture you want)
6. Your own strategy. If you can, try to find a strategy that beats all of the above strategies.

Note that it is not necessary to beat the 5 given strategies to get a ”pass” for the assignment.

Experiment

After you implement the strategies, you will conduct an experiment where the cooperative performance of players will be compared. Each agent will play, one by one, with all the other five agents a predefined number of rounds (*N)* - hence, iteratively. The players do not know in advance how many rounds they will play against the opponent. At the end of each round, the payoff is assigned to each agent according to the payoff matrix from the textbook:

	i defects	i cooperates
j defects	2 2	0 5
j cooperates	5 0	3 3

After playing all *N* rounds with one opponent, the match score (*M-SCORE*) is computed as the average payoff over *N* rounds:

M-SCORE = sum payoff / *N*

After playing with all other agents, the final score (*F-SCORE*) of an agent is computed as the average of match scores over all opponents:

F-SCORE = sum *M-SCORE* / 5

You will repeat this experiment 3 times with number of rounds N = 10, 20 and 30.

Implementation

You will implement the tournament and the 6 strategies in Java. Strategies should be in classes that implement the provided Agent interface:

Agent.java

```java
import java.util.List;

/***
 * An interface to be implemented by all the players
 */
public interface Agent {
    /***
     * Possible choices in the prisoner's dilemma
     */
    public enum Action {COOPERATE, DEFECT}

    /***
     * Choose an action based on the opponent's previous actions
     * @param opponentPreviousActions opponent's actions from the previous rounds,
     *                                empty list if it is the first round
     * @return action of the player in the current round
     */
    Action dilemma(List<Action> opponentPreviousActions);
}
```

A strategy can be implemented in the overridden *dilemma* method. Tournament class should call *dilemma* method for each round and remember the actions returned by the two participating agents. These actions will be provided to the opponent in the next round through *opponentPreviousActions* parameter of *dilemma* method. Tournament class should also assign payoffs at the end of each round, compute match score (*M-SCORE)* at the end of each match and print the final score (*F-SCORE)* at the end of the experiment. The rest of the implementation is up to you.

Delivery

1. The results of the experiment: You will document the result of your experiment on a table where the final score of each agent (*F-SCORE)* will be given for each number of rounds (*N)*.
2. Your strategy: Describe your strategy, its logic and how you think it will encourage cooperation. Provide a pseudocode for your strategy. PS! It is allowed that you get ideas from the web, but you should give reference. Otherwise, if we find out about it ourselves you will not "pass" the assignment.
3. The code: We might test your strategies against each other –if we do so, the winner gets a Kvikk Lunsj chocolate. It is therefore important that you implement the Agent interface we have provided. Your agent should be named after your strategy (make up a nickname for it).

References

1) Launching "The Evolution of Cooperation". Robert Axelrod. Journal of Theoretical Biology, Volume 299, Pages 21-24, 2012. http://www.sciencedirect.com/science/article/pii/S00225
2) Evolution of cooperation among tumor cells. Robert Axelrod, David E. Axelrod and Kenneth J. Pienta. 2013. http://www.pnas.org/content/103/36/13474.full.pdf+html