The second practice on Game Theory

(Innopolis University, Fall Semester 2022, BS-III)

Deadline

27.11.2022 23:59 PM

Outputs

A PDF-report in IEEE format[1] called *NameSurnameReport.pdf* (for example, *IvanIvanovReport.pdf*), SINGLE *Java* file with your code called *NameSurnameCode.java* (for example, *IvanIvanovCode.java*) and your SINGLE source code file used for your testing and testing cases called *NameSurnameTesting.java* (for example, *IvanIvanovTesting.java*)

Requirements

- Program must complile under JDK 17, use of lower or higher JDK will automatically lead to 0 even for correctly written codes
- Code should follow Java Code Conventions[2], it should be readable and should contain JavaDoc
- Report should contain title, student's name, surname, group, description of your method, other relevant stuff
- Do NOT use additional libraries and frameworks (*e.g.* Spring, Lombok, etc.)
- NO extension of a deadline. Works sent after deadline will NOT be evaluated
- NO plagiarism. Plagiarised works will not be evaluated. MOSS will be used for plagiarism detection

Weight

25 points

Grading Criteria

- 30% for the code correctness
- 20% for the rank in tournament
- 10% for the code readability (following Java Code Conventions) and JavaDoc
- 40% for the well-structured and informative report

[1] https://www.ieee.org/conferences/publishing/templates.html
[2] https://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Problem Description

Snowball fight is a physical game primarily played during winter when there is sufficient snowfall. Usually, players are dividing into teams and attack opponents or their fields. In this simulation, we will create a model of the snowball game environment in order to understand the nature of this game.

The environment where players play the snowball game is three territorial regions which we will helpfully entitled as *A*, *B*, and *C* fields. Fields *A* and *B* contain one player on each side. The field *C* does NOT contain any players and should be considered as a hot field where snowballs automatically melt and disappear. The players cannot change their positions and should shot snowballs from the field they are located on.

You may assume that for both *A* and *B* fields the initial number of snowballs is *N*=100. Each *A* and *B* fields have a single Snowball Generating Machine (SGM). Because of those machines each imaginary minute the number of snowballs *N* increases by 1 for both fields.

Both players are using special Snowball Cannons (SC) able to shoot snowballs. Assume that shooting does not melt or destroy snowballs. The players can use Cannons at most once per minute to opponent's field and at most once per minute to hot field (totally at most twice, assume it as sequential shots, where shot to opponent's field happens firstly). Each shot can contain 1 snowball or more. Not shooting is also allowed and requires 0 to be returned by a proper method. If no shooting happened to opponent's and hot fields, then it is assumed that during this minute shooting didn't happen. If SC was used once or twice per minute, then it is assumed as a presence of shooting during this minute. However, the shooting history affects the maximum number of snowballs shoot for the next minutes. The maximum number of snowballs shoot by cannon per minute (together for both shots) is defined by equation

$$f(x) = \left\lfloor \frac{15 \times e^{x}}{15 + e^{x}} \right\rfloor$$

where x is the number of minutes passed after the previous shot (presence of shooting). This equation is not affected by the number of snowballs shoot during the last shots. Cannon is also limited by the number of snowballs present in the field.

Assume that initially both SC are having history of shooting during the previous minute. After 60 minutes (also called rounds), the number of snowballs left in your field has to be minimized for maximizing your payoff.

Task

You will create an agent for the Snowball player, and test your agent against others. Your code should be written in *Java*, and you need to create your agent (public class with public default constructor) implementing the interface as seen below. The interface has:

1. a void method *reset()* which will be called in order to 'reset' the agent before the match with each player containing 60 rounds (imaginary minutes)
2. an integer method *shootToOpponentField()* returning the number of snowballs your agent will shoot to the opponent's field given the last opponent's shot to your field (0 – if this is the first round), and the number of minutes passed after your previous shot (0 – if this is the first round). The *shootToOpponentField()* method returns at least 0 snowballs

3. an integer method *shootToHotField()* returning the number of snowballs your agent will shoot to the hot field given the last opponent's shot to your field (0 – if this is the first round), and the number of minutes passed after your previous shot (0 – if this is the first round). The *shootToHotField()* method returns at least 0 snowballs
4. *getEmail()* method returning String with your Innopolis email
5. an integer default method *maxSnowballsPerMinute()* returning the maximum number of snowballs which can be shot by a player per minute based on the number of minutes passed after the previous shot

You will have to test your agent(s) via a tournament method, then report on your findings.

Finally, you will present a "best agent" which will be part of the class competition.

Submission

You are required to submit:

1. Any Source Code used for your testing and testing cases

2. Your "best agent" code

3. A PDF-report of 2-3 pages in IEEE format which gives a description of your method and talks about the design process and testing used in order to make your player agent.

You are provided with an interface code, which you need to implement by your player

```
package gametheory.snowball;

public interface Player {

    void reset();

    int shootToOpponentField(int opponentLastShotToYourField, int
snowballNumber, int minutesPassedAfterYourShot;

    int shootToHotField(int opponentLastShotToYourField, int snowballNumber,
int minutesPassedAfterYourShot);

    String getEmail();

    default int maxSnowballsPerMinute(int minutesPassedAfterYourShot) {
        double exp = Math.exp(minutesPassedAfterYourShot);
        return (int) (15 * exp / (15 + exp));
    }

}
```

The code of this interface should be excluded from your submission because all of your player codes will be located in the same directory. The multiple versions of the Player interface will lead to compilation error. Moreover, you are NOT allowed to make any changes in the given code, the package also should remain the same. You should assume that your code should be in the package called *gametheory.snowball.students2022*. Please, ensure that you follow all of the given instructions because it can significantly affect your grade.

Your code will be tested by your TA, which main Tournament class will be located in *gametheory.snowball* package. In your agent code, your main class should NOT be present. You should assume that violations of the method parameters' values and returning method values will be checked.

Note: You are expected to create your own code, and write your own report for your agent. You are, however, free to discuss your agent with classmates, plan agents with work together in the tournament, etc. You are recommended to use any VCS or cloud for storing your code. A broken computer should not be the reason for losing your code. To avoid 0 for plagiarism do NOT use public VCS repositories, only private ones