Info233 - Obligatory assignment 3

Delivery deadline: Friday 21.4.2016 14:00

For this assignment you are allowed and encouraged to work in groups of two. Report your group to your Lab Assistant to be given a username and password to be able to access the mysql database located on wildboy.uib.no

The assignment must be delivered at mitt.uib.no. The content itself must be an eclipse project containing your source code, which has been archived in a zip file. Name the zip file with your student username (ex: abc123.zip).

Very Short Description

Write your own arena to allow robot sword-fighting and implement your own swordsman.

The Story

Robot sword-fighting is becoming a more and more popular sport, since there is no chance of anyone getting hurt and it is fun to watch.

This sport takes place in an arena designed particularly for this purpose (see Figure 1).

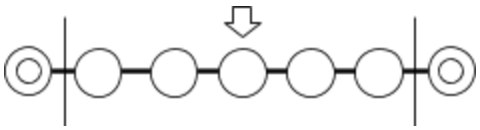

Figure 1

In the beginning of each match, the two opposing robots meet inside the middle circle and each robot begins with a certain amount of energy (a non-negative integer).
The game itself proceeds in rounds.

1. At the beginning of each round, each robot decides how much energy it wants to use in that round to do his sword fighting (this has to be a non-negative integer).
2. After both robots have decided on the energy spent, they will engage in sword-fighting, with the robot who spent more energy winning the fight.
3. If a robot won, he will advance forward to the next circle, while the robot who lost will retreat, meaning that both robots will have moved to the next circle (see Figure 2)
4. If both robots chose to spend the same amount of energy, the round ends in a tie and none of them moves.

5. If a robot runs out of energy, he will still able to retreat if he lost a round (i.e. the other robot used more energy than 0)

The game is over after the robots have moved into the “double” circles at either end of the arena, or if both robots ran out of their energy.

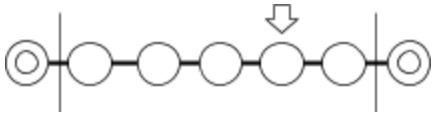

Figure 2: The robot attacking from left to right has won the first fight, the fighting has moved to the next circle (as shown by the arrow)

After the game is over, both players will be awarded points depending on the position where the game has ended (see Figure 3)

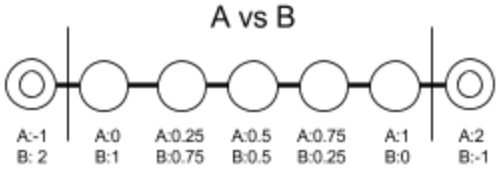

Figure 3: Awarded points based on the ending position of the game. Player A attacks from left to right (player B from right to left)

Your job is to implement a simulator of this game with a capability of maintaining a list of player rankings.

Technical Specification

Implement a class called GameMaster that will facilitate the game. Use the singleton design pattern to ensure a single instance of GameMaster.

GameMaster
+ GameMaster() + getGameMaster(): GameMaster + setPlayers(Player player1, Player player2) + startGame() + listenToPlayerMove(Player player, int move) + evaluateTurn() + updateRanking()

getGameMaster() will return a instance of GameMaster

setPlayers(Player player1, Player player2) will assign the players that are going to play against each other.

startGame() sends a message to each of the players to come up with their next move. This is done by running *player.* for each player.

listenToPlayerMove(Player player, int move) each player uses this method to communicate how much energy he wants to use in the current turn. Treat all invalid inputs (values other than the energy currently available to the player) as equal to 0. If both players made a call to this method during the current round, run *evaluateTurn()*

evaluateTurn() use the information submitted via *.listenToPlayerMove* to identify who won and update the players on the state of the game by either running *player.makeNextMove* (if the game has not yet ended), or *player.gameOver* (in case the game has come to an end).
If the game came to an end, also run *.updateRanking()*

updateRanking() update the player rankings in the ranking table. This table is to be stored in a remote (mySQL) database. Use the table named “ranking”, with columns “player” (VARCHAR128) and “score” (FLOAT). You will be given the credentials required to connect to your group’s database from your seminar leader.

Player
+Player(String name) +registerGameMaster(GameMaster gameMaster) +makeNextMove(int currentPosition, int yourEnergy, int opponentEnergy) +gameOver(float earnedPoints)

registerGameMaster(GameMaster gameMaster) register the gameMaster

makeNextMove(int currentPosition, int yourEnergy, int opponentEnergy) Figure out how much energy the player wants to spend based on the current state of the game. Call *gameMaster.listenToPlayerMove* to inform the gameMaster about the players choice.

gameOver(float earnedPoints) Informs the player that the game has come to an and and how many points he earned in this game.

Your Task

This assignment will be evaluated according to the three following tasks:

1. Implement the abovementioned classes to be able to simulate the game. Package your code as **no.uib.info233.v2017.username.oblig3**
2. Implement two different robots (strategies) how to play the game. That is two subclasses of player performing different operations on *.makeNextMove*.

3. Write a test that checks if the player responds with valid energy values (between 0 and his current energy).
4. Analyze the current design of the Technical Specification. Try to identify its weaknesses,inefficiencies, or other flaws in the design and testability.