B-DOT-501

Card Games

Client/server solution for playing card games

Card Games

binary name: cardGames.sln
group size: 2
repository name: DOT_cardGames_$ACADEMICYEAR
repository rights: ramassage-tek
language: C#, .Net framework
compilation: via MSbuild or any VS2017 compatible builder

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

The goal of this project is to discover the C# language through its dedicated **.Net** framework. You must implement a client/server solution in order to play cards (any game you fancy!).
Create a complete solution composed of a **server** program and **client** programs:

- the server accepts connections from several clients,
- a client connected to the server takes on the role of a card player.

We highly recommand you to use **Visual Studio 2017** as an IDE.
It may look somewhat puzzling at first sight but going through the step of learning every aspect of it may prove beneficial in the long term.
Once you're familiar with it, you can go one step further by adding plugins such as Resharper (the license is free for students), which will provide you some good coding advice and helpful snippets in time of needs.

SPECIFICATIONS

You must build a client / server solution where the client represent a player and the server the game master. The game you have to implement is the "Coinche", a version of the french game "Belote" (check this site out to see every rule of this game).

If the rules of this card game seems too complex, it is tolerated to implement a card game with less rules.
The idea being : you can chose any card game you want (Poker, Belote, The president game etc...) and you will be evaluated on the complexity of the game you chose

.

UNIT TESTS

Unit testing, and testing in general, is a critical part in the process of developing a project. You will be evaluated on the relevance of the tests you made and the coverage they provides.

There is a whole part dedicated to Unit Testing in C# on the MSDN platform, don't forget to check it out!

CONCEPTION

The goal here is not to go through the process of rewriting your own version of something already done, nor to write many lines of code.
We rather demands you to behave as a Software architect and to integrate existing library in your solution.

Therefore, you must understand how those libraries function and do the operational code in order to complete the project.

Before starting the project, take the time to analyze and understand how each component works. In theory, you should study the different possible solutions and choose the right component, depending on the desired needs (**State of the Art**), and even make a quick demonstration program that proves the correct functioning of a component or algorithm (**POC or Proof of Concept**).

Your active participation during the preparatory activities would be a big help.

DOCUMENTATION

In every OOP project, architecture is the key to an healthy project and documentation the key to its maintainability.
It is fundamental to go through the process of thinking your project, bit by bit, with a progressive and efficient approach.

You have to provide a full documentation to clearly justify the different choices you made during the process.

Adding class and sequence diagrams greatly improve the relevance of your documentation.

Network communication

There are numerous libraries that allow you to do the encapsulation of the network communication for your projects sturdier and faster to create.

You are allowed to use the System.Net provided by .Net. Just remember that the stability of your application may be your top priority!

The goal of this section is to create and maintain the connection between a client and a server.
Also think about multi-client management and asynchronous functions so that the program doesn't jam up.

You can use the networkcomms library or any other type of library (such as DotNetty).

Protocol

To have your server communicate with its clients, you need to implement your own RFC.

The key for a successful protocol is to work on it during the conception of your project.
Define your own structure of packets then serialize it before sending it.

It is mandatory to send serialized data.

Bonus

Bonus are always a good things in project. Here you can have fun and let go off your imagination.

Just to help you, here is a few sample of what we can see as a bonus:

- an AI which plays your card games.
- a system of chat / lobby where people can talk and be matched with each others.
- a tournament set where we wait for some people before starting the game.

You also may have the idea to build a graphical interface to give some life to your game. It's a great bonus but keep in mind the work it involves and don't rush the mandatory part of the project to get to it.

You will have plenty of occasion in the next activities to develop your UI skills with WPF / UWP.
Still, if you really want to make a graphical interface, you can do it as a bonus.