FIT1048 Foundations of C++

Assignment Two: Basic Object-Oriented Design and Implementation

Due Date: Friday 15th September 2017

Weighting: 15% of your final mark for the unit

Submission Instructions: A zip file containing your Visual Studio project must be compiled and uploaded to the Moodle site. Your code MUST be submitted as a Visual Studio project to facilitate ease of assessment and feedback.

Task Details:

This assignment consists of one main programming task. The purpose of this assignment is to get you comfortable with designing and implementing basic multi-class C++ programs. The task is detailed later in this assignment specification, as are their specific marks allocation.

Successful completion of the fundamentals of the task as described may obtain you up to a maximum of 80% of the total assignment marks. The last 20% of the mark will be allocated to additional functionality that you can design. This provides an opportunity for you to implement a feature of your choice and ensures that each student submission is suitably different. The additional functionality should demonstrate advanced or more complex application of principles covered to date. It need not be large amounts of work but should demonstrate a willingness to explore new and advanced concepts. You MUST detail what you have done in an accompanying "readme" file.

The assignment must be created and submitted as a Visual Studio 2015 project. You may complete the exercises in your preferred IDE, however you should create a Visual Studio project in order to submit. This project must then be zipped up into one zip file for submission. The zip file **MUST** be named "FIT1048-AA2-*YourAuthcateID*.zip". This zip file must be submitted via the Moodle assignment submission page. Note… to reduce the file size of your zip, you can delete the "ipch" folder from your project before zipping without affecting your program.

Explicit assessment criteria are provided, however please note you will be assessed on the following broad criteria:

- Meeting functional requirements as described in the description
- Demonstrating a solid understanding of object-oriented design and C++ coding, including good practice
- Following the unit Programming Style Guide
- Creating solutions that are as efficient and extensible as possible

NOTE! Your submitted program MUST compile and run. Any submission that do not compile will be awarded zero marks. This means you should continually compile and test your code as you do it, ensuring it compiles at every step of the way.

If you have any questions or concerns please contact Matt as soon as possible.

Assignment Task: Go Fish!

Overview:

You are to implement a computer -based variant of the card game Go Fish. This is a two-player card game played with a standard deck of cards in which the aim of the game is to capture sets of 4 cards of the same value (for example 2, 3, Queen, etc). You can play an online version here: https://cardgames.io/gofish/. In your version, you will play against one computer opponent.

Basic Play:

To begin, 7 cards are dealt to each player from a standard deck of 52 cards. The game plays as follows:

- The first player asks the second player for a card of a particular value. They must ask for a card of the same value as one of the cards in their hand.
- If the second player has a card of that value, they will remove it from their hand and give it to them. If they do not have one, they will tell the player to "Go Fish!". This means the first player must take a new card from the deck.
- If the first player was successful in asking for a card, they can ask for another card. This continues until they are told to "Go Fish", at which point the second player will then start the process of asking for a card.
- If a player gets 4 cards of the same value, they reveal the set to the other player, score 1 point, and remove those cards from their hand.
- If at some point a player has no cards in their hand, they take 7 new cards from the deck (or as many as left if there are less than 7).
- The game continues until all sets of cards are formed. The player with the most points wins.

Computer Player:

Obviously, you as a player can make strategic choices as to which card you ask the other player for. Your computer opponent does not need to be "intelligent"… it just needs to at least randomly ask for a card that it has at least one of in its hand. You can implement any other AI you like as long as it conforms to that basic game rule. More sophisticated AI can be considered extra functionality.

Extra Functionality:

The marking criteria indicates that you should make some individual additions to this in order to achieve the final 20% of the mark. This is up to you, however could include such features as (but is certainly not limited to):

- More sophisticated AI
- Adding more computer players
- Being able to save a game
- Making different sets worth more points

It is up to you!

Program Design:

In the final assignment you will be able to design your program in any way you like. For this assignment however, you MUST implement your program using the following classes (as a minimum, you may include more if appropriate):

- Player class: Holds details of the player including a collection of cards
- Card class: Consists of the value and suit of the card

Assignment 2 Assessment Criteria

Go Fish! (up to 50 marks total)

Name:

Marker:

Grade:

	HD	D	C	P	N
Does the program compile and run? **ZERO marks will be awarded for a non-compiling program.**	Yes/No				
Class Design (8)					
Card Class (4)					
☐ Required Data Members & Member Functions					
☐ Contains only aspects that relate to a "Card", i.e. no data members or member functions that are not directly related to a Card					
Player Class (4)					
☐ Required Data Members & Member Functions					
☐ Contains only aspects that relate to a "Player", i.e. no data members or member functions that are not directly related to a Player					
Functionality (25)					
☐ Game set up, including creating a card deck, dealing shuffled cards, etc. (6)					
☐ Appropriate game dialog and player interaction (3)					
☐ Implementation of successful card draw (3)					
☐ Implementation of "go fish" (3)					
☐ Computer AI working correctly (4)					
☐ Appropriate display to screen (4)					
☐ End game condition (2)					

Overall Quality of Solution and Code (7)					
☐ Does the program perform the functionality in an efficient and extensible manner? (2)					
☐ Has a well-designed OO program been implemented? (3)					
☐ Has an appropriate Programming Style Guide been followed, including documentation? (2)					
Extra Functionality (10)					
☐ Does the program addition demonstrate advanced application of the week's concepts? (5)					
☐ Does the program addition demonstrate functional creativity? (5)					

Comments: