School of Electronic Engineering and Computer Science

ECS769P – Advanced Object Oriented Programming

Card Games – Developing a pack of generic playing cards for games using object-oriented approach

20% Project Assignment
Release Date: **Thursday, 13th February 2020**
Demonstration: **Monday, 6th April 2020 (4-6pm)**
Submission: **Thursday, 9th April 2020**

Coursework Submission details: electronic submission on QM+.
Feedback: verbal feedback given during demonstration and written feedback given via QM+.
Marks and written feedback returned: approximately 2-3 weeks after submission.

1. Specification overview

The aim of this assessment is to gain experience in the design and implementation of C++ programs using an object-oriented approach. Your task is to model a pack of generic playing cards, and implement human-computer card games. You need to implement the "Guessing" game and the "Black Jack" game as specified below. To simplify the requirement, we define:

Card

A conventional playing card has a face and a suit. For example:
Ace of Diamonds
Five of Spades
Ten of Hearts
King of Clubs

Deck

A deck contains the set of **52** unique playing cards.
Diamonds: Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King.
Hearts: Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King.
Spades: Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King.
Clubs: Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King.

Shuffle cards

Put the 52 playing cards in a random order.

Deal a card

Get a card from the top of the deck.

2. A card guessing game

This game is similar to the lab 1 exercise 3, but this time the player guesses the card.

Playing this game (human vs computer)

- When the game is loaded, it should firstly **shuffle** the deck.
- The bank (computer) deals one card (invisible to the Player).
- The player guesses this card by typing the face and suit of the card, for example "Two of Hearts".
- The bank should then show a message that indicates whether the face value is too small or too big and whether the suit is correct. (Note: Ace->King, smallest to biggest).
- When the card has been guessed correctly, print the number of valid guesses that the player took. It then should ask the player whether to continue.
- If the player chooses to continue, the bank should deal the next card from the deck.
- User may input "quit" to quit the game at any time.

3. Card game: Black-Jack

The full description and rule of this game can be found at https://en.wikipedia.org/wiki/Pontoon_(card_game) , however you are only required to implement a simplified version of the game, please see the objective and rules below.

This game is also known as "Pontoon" or "Twenty-One". In this game, each card has a value. Ace-1, Two-2, Three-3, Four-4, Five-5, Six-6, Seven-7, Eight-8, Nine-9, Ten-10, Jack-10, Queen-10, King-10. (In the real game Ace can have value of 1 or 11, **you do not need to consider this.**) The aim of the game is to collect a hand of cards (up to 5) that sum to a value as close as possible to, but not exceeding 21. A hand that has a value greater than 21 is said to be 'bust' and is worth nothing.

Objective and rules

The card game is played between a computer and a human. The bank (computer) deals a hand of two cards (face down) for itself and a hand of two cards for the player (human). The player can then inspect his own cards and choose either to 'stick' or 'twist' according to the value in hand. If the player chooses 'twist', the bank deals the player another card. This continues until the player chooses 'stick', has 5 cards in hand, or is 'bust'. If the player chooses 'stick', it is the bank's turn to play. The bank deals itself cards until it has a hand with a value greater than or equal to that of the player, has 5 cards in hand or goes 'bust'.

To win the game, player should collect a hand of cards that sum to a value as close as possible to, but not exceeding 21. The player wins if there is less than 5 cards in hand, total value less than or equals to 21, and the total value in hand is greater than the bank. If both the player and the bank have the same value, the one has more cards in hand (maximum 5 cards) wins. If they have the same value and same number of cards, the player wins.

To aid in understanding the specifications, a sample interaction can be found in the Appendix.

Playing this game (human vs computer)

- When the game is loaded, it should firstly **shuffle** the deck.
- The bank (computer) deals two cards for itself (invisible to the Player), and two cards for the player (human).
- The player can view his own two cards. Two options are given to the player: 'stick' or 'twist'.
- If the player chooses 'twist', the bank deals the player another card. This continues until the player chooses 'stick', has 5 cards in hand, or is 'bust'.
- If the player chooses 'stick', it is the bank's turn to play. The bank deals itself cards until it has a hand with a value greater than that of the player or has 5 cards in hand, or goes 'bust'.
- When a round of game is over, the player can choose to continue a new round or quit.
- When player chooses to continue, the above steps will be repeated.
- When player chooses to quit, statistics should be shown on the screen and the program is terminated.

Example of statistics:

```
Rounds: 5
Won: 3
Lost: 2
--
Round 1: won.
Round 2: lost.
Round 3: won.
Round 4: lost.
Round 5: won.
```

4. Requirements

You are required to model and implement the pack of playing cards and two card games as described above using C++ programs. In implementing your solution, you should bear in mind that the aim of this assessment is to gain experience in object-oriented programming, so your program should apply the object-oriented principles such as encapsulation, abstraction, inheritance and polymorphism. You should use the appropriate data types, data structures and custom templates.

You are also required to write a short report describing the design and implementation and the idea behind the solutions you have used with the focus on reusability and generality. For example: the algorithm used for shuffle the cards; how the cards/deck classes are reused and how they can be reused for other card games; what the data structure you used for representing the cards/deck and why; how polymorphism is implemented; what custom template used and why, etc.

The short report should have no more than 4 pages.

5. Submission Details

It is an electronic submission on QM+. Make a ZIP file, including all the required files:

1. All the source files (header files and source code files)
2. A readme.txt describing how to configure/run your programs, the programming environment you used to create and build the program and the C++ compiler and version you used.
3. The report in PDF format.

6. Demonstration

You must attend the lab session on **6th April 4-6pm** to demonstrate your programs. You may use the Lab PC or bring your own laptop.

7. Marks breakdown (approximate)

50% of the marks are achieved if the program works correctly as described in the Spec.
30% of the marks are achieved by the quality of the solutions. This includes: efficient algorithms that make good usage of memory and time, capture exceptions, error handling, good OO design, appropriate use of inheritance and polymorphism etc.
20% of the marks are achieved by the quality of the report.

Please use the messageboard on QM+ for enquires and discussions.

8. Appendix

Sample interaction of the Black Jack game (red text is user input)

your hand is :

King of Spades
Jack of Diamonds
value = 20

[s]tick or [t]wist? **s**

banks hand is :

six of Spades
King of Clubs
value = 16

the bank draws a card...

the banks hand is :

six of Spades
King of Clubs

Ace of Clubs
value = 17

the bank draws a card...

the banks hand is :

six of Spades
King of Clubs
Ace of Clubs
Ace of Diamonds
value = 18

the bank draws a card...

the banks hand is :

six of Spades
King of Clubs
Ace of Clubs
Ace of Diamonds
four of Spades
BUST!!!

Well done, you won!!!

do you want to play again ([y]es/[n]o)? **y**

your hand is :

four of Spades
King of Clubs
value = 14

[s]tick or [t]wist? **t**

your hand is :

four of Spades
King of Clubs
King of Spades
BUST!!!

Bad luck, the bank won.

do you want to play again ([y]es/[n]o)? **y**

your hand is :

eight of Diamonds
five of Clubs
value = 13

[s]tick or [t]wist? **t**

your hand is :

eight of Diamonds
five of Clubs
six of Diamonds
value = 19

[s]tick or [t]wist? **s**

banks hand is :

eight of Hearts
five of Diamonds
value = 13

the bank draws a card...

the banks hand is :

eight of Hearts
five of Diamonds
ten of Hearts
BUST!!!

Well done, you won!!!

do you want to play again ([y]es/[n]o)? **n**

Rounds: 3
Won: 2
Lost: 1
--
Round 1: won.
Round 2: lost.
Round 3: won.

Thanks, bye!