OOP Programming Assignment

Term: May 2021

1. General Instructions

The general guideline for this assignment is as follows:

1. Evidence of plagiarism or collusion will be taken seriously, and University regulations will be applied fully to such cases, in addition to ZERO marks being awarded to all parties involved.
2. The total mark for the assignment is 100 and contributes 20% to the total grade.
3. This is a **group assignment of 2 persons**.
4. The deadline for the submission of reports is on **Week 12 – 27 August (Friday) 11.59 pm**.
5. Each student must submit ONE (1) **complete source code (src) with UML diagram (pdf)** and ONE (1) **recording video (6 mins max and size does not exceed 100MB)** to explain each classes function and the output of your application. Please **compress all the files** and rename with your name, student id and program.

   Example: *GroupXX_CS_Assignment*
6. Your file must be submitted to the following platform before the **due time/date**.
   Send your answer script to Email Account:

   i. CN students please send your assignment to:
   UCCD2044FACN@gmail.com

   ii. CS students please send your assignment to:
   UCCD2044FACS@gmail.com

   iii. CT students please send your assignment to:
   UCCD2044FACT@gmail.com

   iv. IA students please send your assignment to:
   UCCD2044FAIA@gmail.com

   v. IB students please send your assignment to:
   UCCD2044FAIB@gmail.com

   Note: Use your "1UTAR" email account to submit your assignment. For the title of your email, please use the file name of your compress file name_ Assignment. That is,

   GroupXX_CS_Assignment

7. **Late submission** after the due time/date may incur a late penalty as shown below:
   i. 0 hour < lateness ⩽ 0.5 hour: 50%-mark deduction
   ii. 0.5 hour < lateness ⩽ 1 hours: 100%-mark deduction

2. Background

In this assignment, your team is asked to develop a **Card** game using **Java** and **Object-Oriented Programming** technique. This game is played using standard 52-card deck (13 ranks of 4 suits). The followings describe the game rules of the proposed card game.

Rules and Instructions for Game Play

The goal is to be the player that has the highest score point. The highest score points to be a winner.

- **THE DEAL**

The game is played by 2~4 players. Each player gets **7 cards (rounds)**.

- **THE PLAY**

Initially **7** deck of shuffled cards are distributed to all players (7 rounds). In each round, a player is given a chance to select a single card from his own set of **cards (7)**. The player with maximum card number wins the round and gets a point. At last, player with the greatest number of points wins the game.

Example: Players 1 and 2 will give 7 cards randomly. Player 1 select card and get [♣A], Player 2 select card and get [♥2]. Player 1 won and get 1 point because [♣A] higher than [♥2].

- **HOW TO KEEP SCORE**

The game ends after player already choose/ finish select all **7 card (rounds)** or decide to **Stop Game**. Player with the most point win the game.

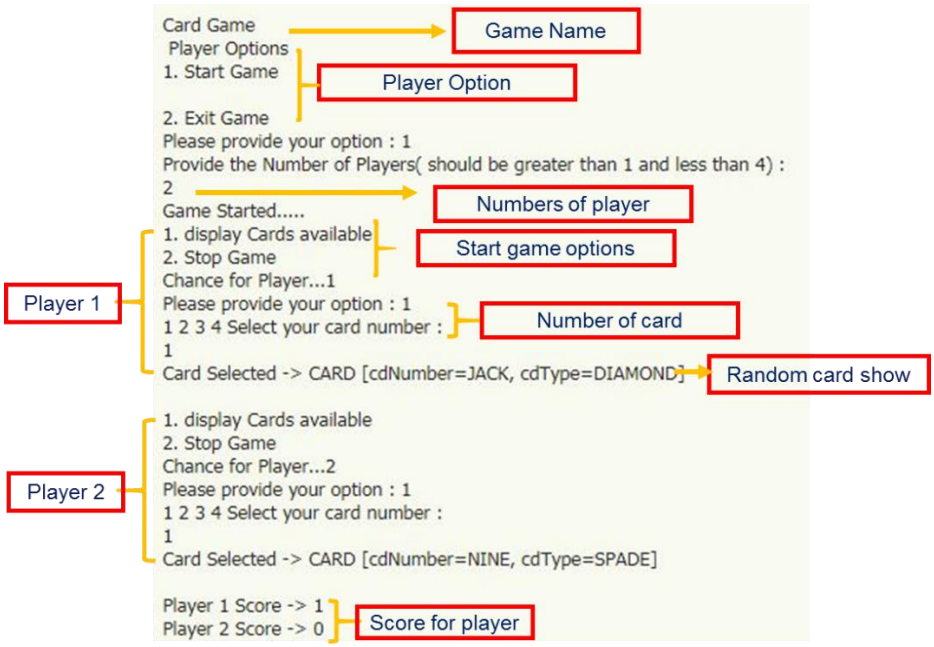
3. Requirements

Your assignment is to develop a program to simulate the card game described in the previous section.

3.1 Program Functions (45%)

Your program must fulfill the following functional specifications:

- Allow ***2-4 human*** players to play the game. No computer player (AI) is required.
- Need to have Player Option: ***Start Game and Exit Game***.
- Shuffle the deck, distribute **7 cards** to each player and start a game.
- ***Display cards*** (numbers) available and option to ***Stop Game*** in the middle of game for player.
- ***Select the numbers*** and ***show card selected*** by player.
- Show the ***score for both player***.
- Example Output:



- Go back to ***Player Option*** after ***finish or Stop Game***, enable to choose to ***start or exit***.

3.2 Object-Oriented Design (20%)

You must use Java and object-oriented programming techniques in your implementation. Your design should follow good object-oriented design principles such as:

- Single responsibility – a class should have responsibility for a single functionality of a program and that responsibility should be encapsulated by the class.
- Open/closed principle – classes should be open for extension but closed for modification.
- Efficient and no redundancy – keep it simple.

The followings are the basic classes that **MUST** exist in your program. The responsibilities of each of the classes, as well as some of their data fields and methods are suggested. You are free to select the data types and method signatures for the classes, and to add additional data/methods if necessary.

Card:

Card is the class for representing a card.

- This class should define generic card properties such as suit (*spade*, *heart*, *club*, *diamond*) and rank (*A*, *2~10*, *J*, *Q*, *K*).
- A card object should be **immutable**; meaning its content (suit and rank) cannot be changed once it is created.
- Should support **sorting** and **compare** of an array of cards using Java generic sorting algorithm.

Player:

Player class is used to represent a player in the game that has the following requirements:

- Has a name. (Player 1 etc.)
- Compare and returns the total score.

Card Game:

The **Card Game** class is the game engine which controls the flow of the game, imposes the basic game rules, displays messages and gets inputs from the players. It should at least have the following attributes and methods:

- Map **Cards** to **Player** and decide number of **Cards** per player. (**import.java.util.Map** class) https://www.geeksforgeeks.org/map-interface-java-examples/
- An array/list of **Player** objects. Number of players can be **2~4**.
- An array/list of **Card** objects. The deck should be shuffled at the start of game and distribute the cards to player.
- ***Play game, display scores, display cards for player, display winner, create multiple player and get next player***

Game Demo (Main):

A main function class that starts the game.

** Game (Optional): (https://www.w3schools.com/java/java_interface.asp)

Implement **interface** use ***implements*** keyword. Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance.

3.3 Extra Effort (10%)

- User Interface (GUI) – Enhance the application by providing easy to use and nice user interface.

3.4 Presentation: Video (25%) – 6 mins Max (3 mins code & 3 mins functions description)

- Title of the assignment, student names and student IDs.
- ***Program (code) description***, including what is the code about and how it works.
- ***Function description***, demo how each function working, such as card for each player, winning and etc.
- ***UML Class diagrams*** showing the program structure.

Both students need to be in the presentation video