TCP1201 Objected-Oriented Programming and Data Structures
Assignment Part 2

Trimester 2, Session 2022/2023
Faculty of Computing and Informatics
Multimedia University

Part 2 Submission Deadline: 18 June 2023 (Sunday), 11:59pm

General Information

1. Make sure the submitted program is compile-able. Zero mark will be given for late submission or uncompilable program.

A. This 2-part assignment contributes **40%** of the total course mark.
B. Part 1 contributes 10% while Part 2 contributes 30%.
C. Part 2 is an extension of Part 1.
D. Part 2 presentation is a live presentation. All members must present in the presentation.
E. ZERO mark for late submission.
F. STRICTLY NO COPYING from other students in other groups or from any other sources (Internet, books, etc.). ZERO mark to students who plagiarize AND to the students who share their code. For this assignment, plagiarism means the following:
    1. Turning in a work that, from the examiner's point of view, you do not sufficiently understand.
    2. Turning in someone else's work (whether partly or fully) as your own.
    3. Any means of cheating.

Grouping

1. Keep the same members as in Part 1 unless your group faces problems (discuss with your lab lecturer).
2. Each group member may be given different marks based on their contributions.

Go Boom Game

A. Watch How To Play Go Boom YouTube to understand the game.
B. Notice that the **start of the game** in this assignment is **different** from the YouTube video above.
C. Your task is to develop an object-oriented Java program for the Go Boom game.
D. Part 1 features (console output):
    1) All cards should be faced up to facilitate checking.
    2) Start a new game with randomized 52 cards.
    3) The first card in the deck is the first lead card and is placed at the center.
    4) The first lead card determines the first player:
        - A, 5, 9, K for Player1
        - 2, 6, 10 for Player2
        - 3, 7, J for Player3
        - 4, 8, Q for Player4
    5) Deal 7 cards to each of the 4 players.
    6) All players must follow the suit or rank of the lead card.
    7) The highest-rank card with the same suit as the lead card wins the trick.
    8) The winner of a trick leads the next card.
E. Part 2 features (console + GUI)
    1) If a player cannot follow suit or rank, the player must draw from the deck until a card can be played.
    2) When the remaining deck is exhausted and the player cannot play, the player does not play in the trick.
    3) Finish a round of game correctly. Display the score of each player. (No need to implement stopping the game at 100 points because there won't be sufficient time for that during presentation.)
    4) Can exit and save the game (use file or database).
    5) Can resume the game. The state of the game is restored when resuming a game (use file or database).
    6) Reset the game. All scores become zero. Round and trick number restart from 1.

7) Support GUI playing mode (cards should be faced up or down as in the real game). The GUI can be in JavaFX, Swing, Spring, or Android.
8) Keep the console output to facilitate checking. The data in console output and the GUI must tally.

F. Output

1) Design your GUI to be user friendly.
2) For Part 2 features, design the GUI as the main control, console produces corresponding output based on what happens in the GUI.
3) If your group is unable to produce GUI output, you may develop your program in console, but your group will lose the GUI marks. Check the marking rubric below.

G. Commands to the GUI Program

In GUI, the following commands should be invoked via buttons or other appropriate controls:

1. Start a new game.
2. Resume previous game.
3. Exit the game.
4. Draw cards from deck until a playable card is obtained. If the deck is empty, skip to the next player.
5. Click a card to play.
6. Reset the game.

Part 2 Submission Format

Submit the works to the GitHub under your MMU GitHub Education account. Make the repo **private** so that non-members cannot access your code. Add your lab lecturer as a **collaborator**.

1. Submit the following items to your group's GitHub repo. One group makes one submission. Use the same repo as Part 1.
    a. **PART2** folder – contains your **source code and any data files** (*.java, etc.)
    b. **PART2.md** – Fill in the required info.
2. Paste your group's GitHub link into the Google Classroom for Assignment Part 2.

Part 1 Marking Rubric

Item	Mark
1 Part 1 Features (8 marks) 1 mark for a fully working feature. 0.5 mark for a partially working feature. 1) All cards should be faced up to facilitate checking. 2) Start a new game with randomized 52 cards. 3) The first card in the deck is the first lead card and is placed at the center. 4) The first lead card determines the first player: • A, 5, 9, K for Player1 • 2, 6, 10 for Player2 • 3, 7, J for Player3 • 4, 8, Q for Player4 5) Deal 7 cards to each of the 4 players. 6) All players must follow the suit or rank of the lead card. 7) The highest-rank card with the same suit as the lead card wins the trick. 8) The winner of a trick leads the next card.	/8
2 Video Presentation (2 marks) 2 marks for presenting all 8 features excellently. 1 mark for presenting some features adequately.	/2
Part 1 Total (10 marks) **Zero mark for no presentation, late submission, or plagiarism.**	/10

Part 2 Marking Rubric

	Item	Mark
3	**Part 2 Features (7 marks)** 1 mark for a fully working feature. 0.5 mark for a partially working feature. 1) If a player cannot follow suit or rank, the player must draw from the deck until a card can be played. 2) When the remaining deck is exhausted and the player cannot play, the player does not play in the trick. 3) Finish a round of game correctly. Display the score of each player. (No need to implement stopping the game at 100 points because there won't be sufficient time for that during presentation.) 4) Can exit and save the game (use file or database). 5) Can resume the game. The state of the game is restored when resuming a game (use file or database). 6) Reset the game. All scores become zero. Round and trick number restart from 1. ~~7) Support GUI playing mode (cards should be faced up or down as in the real game). The GUI can be in JavaFX, Swing, Spring, or Android.~~ (Mark separately below) 8) Keep the console output to facilitate checking. The data in console output and the GUI must tally.	/7
4	**GUI Quality (3 marks]** Use card images, choice of UI controls, color and size, etc. 3 – Working GUI with card images and good design. 2 – Working GUI without card images. **1** – Partial GUI or very buggy **0** – No GUI	/3
5	**User Friendliness (5 marks]** Intuitive input and output, proper error handling. 5 – Beyond expectation 4 – Above expectation 3 – Meet expectation 2 – Below expectation **1** – Poor	/5
6	**Class Design (5 marks)** Use of inheritance, composition, and interface, etc. in the program. 5 – Beyond expectation 4 – Above expectation (good class design including inheritance and composition) 3 – Meet expectation (has both inheritance and composition) 2 – Below expectation (without inheritance or composition) **1** – Poor (1 class only)	/5
7	**Use Appropriate Data Structures (5 marks)** 1) Array or List [1m] 2) Set [2m] **3)** Map [2m]	/5
8	**Presentation and Q&A (5 marks)** Demo, and explanation on class design and data structures used. 5 – Beyond expectation 4 – Above expectation 3 – Meet expectation 2 – Below expectation **1** – Poor	/5

5 Bonus (2 marks) Android or Spring GUI	/2
Part 2 Total (30 marks) **Zero mark for no presentation, late submission, or plagiarism.**	/30
Total (Part 1 + Part 2)	/40