CS102 – Algorithms and Programming II
Homework 2 - Simplified Okey Game
Spring 2025

This is group homework. You will complete this homework with your project group and submit one solution. Make sure you collaborate and divide the work equally as much as possible. Include a text file “Members.txt” inside your submission that includes your group member names and their IDs. For the group submission, it is enough for one member to upload it on Moodle.

You will also write a “Reflections.txt” where you reflect on the homework using at least 100 words: Was it easy to work on a partial implementation? Were there any difficulties working as a team? You will submit this file under the individual submission page, all members should complete reflections individually.

For this homework, you will complete the partial implementation of a simplified Okey game variant. For the complete set of rules of the original game, you can check the following page:

https://en.wikipedia.org/wiki/Okey

Our version of the game will be a simplified console application. The following rules are altered to make the game simpler:

- There are no jokers, so we do not reveal a joker at the beginning of the game, and no tile can be a substitute for a different one. There are also no false joker tiles.
- There are 4 copies of each tile (color-number combination) instead of 2, but the range of tiles is decreased to [1,7]. In total, there are 4 colors, 7 numbers, and 4 copies making up 112 tiles.
- Consecutive numbers cannot be chained. Normally, you can have a chain of the same color (i.e., 2-3-4-5); this is not allowed in our variant. Chains can be made only using the different colors of the same number.
- Any player who has three chains of length 4 wins the game. If there are no tiles on the ground to draw and there is no winner, the game ends with a tie.
- There can be only one chain per value in a winning hand; for example, you cannot have two chains of 5’s. Each chain of length 4 should use a different number.
- The starting player is predetermined. The player at index 0 starts first.

The game will be single-player, played against 3 computer opponents. The human player starts first. The application will display the player's current tiles and then will display the options to get the user’s choice. The high-level logic of the game is already included in the given partial implementation. You should complete any code marked with TODO based on the explanations in the comments. You can add new classes or methods if you think necessary. You can modify the existing code or fix their mistakes if there are any. Don’t forget that the partial implementation is there to guide and help you, but if it is not useful, you may shape it according to your design.

In the Okey game, players take turns to complete their tiles into a winning hand. At each turn, the player may pick a tile from the stack or the tile discarded by the previous player. The first starting player does not pick a tile.

At the beginning, the stack of tiles is shuffled, and each player is given 14 tiles except for the first player, who gets 15 tiles. A tile should be discarded without drawing a new one in the first player's first turn. Then, in any of the turns, each player has two options to draw a tile from: the tile discarded by the previous player or a tile from the tile stack. The discarded tile is visible to all players to strategize; the tiles in the stack are not visible unless a player draws it, and in that case, it is only visible to the player who draws it. The player whose turn now picks a tile to make their hand 15 tiles and then discards one of the least useful tiles.

A winning hand should have three chains of length 4 in our version. For example, the following hands are winning hands (the valid chains of length 4 are underlined):

Ex1: 1-2-2-2-2-2-3-5-5-5-5-6-6-6-6
Ex2: 2-2-2-2-2-2-2-3-3-3-3-5-5-5-5

Note that the player whose turn is now draws one extra tile to have 15 tiles in hand, and these extra tiles do not disturb the chain. In a chain of a specific number, there can only be one instance of each color. For example, in the case of "**2-2-2-2-2-2-2**", duplicate **2**'s does not contribute to the chain, but also does not disturb the others to make a chain of 4. For example, **2-2-2-2** will not make a chain. Additionally, the duplicate tiles cannot contribute to two different chains in a winning hand. For example, if we have the following hand:

Ex: 2-2-2-2-2-2-2-2-2-2-2-2-2-2

We just have one chain of length 4, the first instance of each 2 contributes to the chain, and the duplicates are ignored. It would be a good strategy to discard duplicates when possible.

The existing classes of the partial implementation are as follows:

- ApplicationMain.java: Includes the main method that initializes and processes the game. Until the game finishes, the main loop receives input on the player's turn.
- OkeyGame.java: Includes the high-level logic of the game. Contains 4 players, and the tiles stack. You will complete the methods such as shuffling the tiles, handling the computer's turn, and checking the win condition inside this class.
- Player.java: Includes the logic for one player. It contains the player's tiles. Methods such as adding or removing a tile from that player's hand, displaying the player's tiles, and finding the longest chain are included in this class.
- Tile.java: Represents a tile with a specific color and value. Methods to compare and print a tile are included in this class.

The requirements for all the methods are included in the partial implementation.

Note that you should keep the tiles sorted in ascending order to check the win condition easily. You should make the add tile method of the player to insert new tiles in order so that each player's hand is always sorted without using a sorting algorithm.

For example, suppose our hand is:

1-2-2-3-3-3-4-4-4-4-6-6-6-6

If we draw **5**, this should be inserted into the correct position to keep the order. This requires you to loop over the existing tiles to find the correct position, then shift the remaining tiles to the right by one:

1-2-2-3-3-3-4-4-4-4-5-6-6-6-6

The computer players should decide whether to draw from the stack or the tile discarded by the previous player following an algorithm. Here, you should check if drawing the discarded tile would benefit that player (i.e., it would increase the chain length for one of the numbers); if so, the computer should draw the discarded tile; otherwise, the computers should draw from the tile stack. Based on the current tile chains, the computer should also decide on the tile to discard. If there is a duplicate tile, discarding it first can be the best choice. Then, discarding the single tiles and the ones contributing to the shortest chains would be meaningful. You can implement different strategies for determining the tile computer player discards. The computer cannot look at the current hands of the other players but can keep track of the tiles they pick up from the discard stack. Try to come up with interesting algorithms.

Your program will also include an open-hand setting to see the computer players' hands for debugging. If the game ends with one of the players reaching a winning hand, you will display that player as the winner. If the game ends with no tiles in the tile stack, display each player's hand at the end.

You should also include an option for having 4 computer players: In this case, the game does not require any input from the user and the computer players will take turns printing their hands and actions until the game ends. The user should be able to choose this option or the regular game at the beginning of the application.

Since coloring is not available in the default console, we will use the following letters to indicate the colors:

```
Black Tiles:  (K1-K2-K3-K4-K5-K6-K7) * 4 = 28 Tiles
Red Tiles:    (R1-R2-R3-R4-R5-R6-R7) * 4 = 28 Tiles
Blue Tiles:   (B1-B2-B3-B4-B5-B6-B7) * 4 = 28 Tiles
Yellow Tiles: (Y1-Y2-Y3-Y4-Y5-Y6-Y7) * 4 = 28 Tiles
------------------------------------------------------------
                                          112 Tiles in total
```