CS102 – Algorithms and Programming II
Homework 2 - Simplified Okey Game
Fall 2023

This is a group homework. You are going to work on this homework together 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 this submission, it is enough for one member of the group to upload it on Moodle.

There will also be an individual submission for your reflections about this homework. You will submit “Reflections.txt” on Moodle where you write at least 100 words of reflections about working on this homework: Was it easy to work on a partial implementation? Were there any difficulties working as a team? What was your role in the team work? Each person will submit their reflections under the corresponding assignment page individually.

For this homework, you are going to complete the partial implementation of a simplified Okey game. For the complete set of rules of the 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 no false joker tiles, so instead of a total of 106 tiles in the game, we only have 104 tiles. That is 4 colors * 2 sets * 13 numbers.
- No scoring. The finishing player is declared the winner without calculating any score.
- No circular continuity of the numbers. Normally, you can have 11-12-13-1 as a chain; to simplify the game, this is no longer allowed. You can have 11-12-13, but it cannot continue with a 1.
- 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 current tiles of the player 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 in case it is not useful, you may shape it according to your design.

In the Okey game, players take turns trying to complete their tiles into a winning hand. At each turn, the player may pick a tile from the stack or pick the tile discarded by the previous player. The first starting player does not pick a tile. Each player has 14 tiles. 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 a collection of sets and runs. We call these chains in the implementation for simplification:

- A set contains tiles of the same number with different colors.
- A run contains consecutive numbers sharing the same color.

In our case, both are called chains. The winning hand has either 2 chains of 4 tiles and 2 chains of 3 tiles or 1 chain of 5 tiles and 3 chains of 3 tiles. In the original game, there are more winning hands, but in our simplified version, these two are sufficient.

The existing classes 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 player's turn.
- OkeyGame.java: Includes the high-level logic of the game. Contains the 4 players and tiles stack. You will complete the methods such as shuffling the tiles, handling the computer's turn and checking the win conditions 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 tiles of the player and finding the longest chain that can be formed by a tile is included in this class.
- Tile.java: Represents a tile with specific color and value. Methods to compare and print a tile is included in this class.

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

Note that for checking the win condition, tiles can be organized differently. For the longest chain that can be achieved for a tile, the tiles of the player should be either sorted by color first or value first. Suppose the following color first order and the chains achieved per tile:

1Y	2Y	3Y	8Y	4B	5B	7B	9B	13R	1K	2K	3K	4K	5K
3	3	3	1	2	2	1	1	1	5	5	5	5	5
3			1	2		1	1	1	5				

Note the color letters are Yellow, Blue, Red, and blacK.

Since 1Y-2Y-3Y forms a chain of 3, we would say the chain achieved for 1Y with the color first ordering is 3.

Let's also consider the value first ordering for the same tiles:

1Y	1K	2Y	2K	3Y	3K	4B	4K	5B	5K	7B	8Y	9B	13R
2	2	2	2	2	2	2	2	2	2	1	1	1	1
2		2		2		2		2		1	1	1	1

In this case, as 1Y-1K forms a chain of 2, we would say the chain achieved by 1Y with value first ordering is 2. So, color first ordering is more advantageous for 1Y and the longest chain achieved is 3 in this case. While determining a tile to discard for the computer players, you can look at the longest chains achieved by each tile and discard the one of the minimums. As for checking the winning condition, the longest chains are again useful. The longest chain configurations for the winning conditions are given in the partial implementation. Again, the rules are more relaxed to keep the game simpler and it will be enough to check the winning conditions using the criteria given in the comments of the code.