ECON381 Fall 2024

Homework Assignment 2

In the popular board game **Okey**, users are presented with a board and 14 keys (representing gaming cards).

Figure 1: A typical Okey board

You can easily Google for the rules if you have never played the game before. It is simple in rules but complex in game play as there are multiple users involved.

An integral part of the game is to determine if you are done. There are two ways to achieve this.

1. All 14 keys should be part of blocks of size 3 or more. A block is either a number of consecutive numbers of the same color (see 2,3,4,5,6,7 above) or same number with different colors (see the three different color 11's above).
2. You can also have 7 pairs (same number, same color).

One of the keys is considered as a Joker (Okey) and can replace any required key. There is also a replacement key (Sahte Okey) that replaces the Joker's ordinary function. For this homework assignment we will disregard these two rules and assume a simpler version of the game.

There is a simple Java class that shows you how to represent a key in Java in the Annex.

Your assignment involves the following questions in a report.

1. During game play you will add and remove keys to the board. What kind of operations would that mean? Please elaborate.
2. To determine if a user is done, what kinds of checks would you need to do? Please elaborate.
3. Given the OkeyKey class available and given your discussion for the above two topics, would you rather hold the 14 keys in the Okey board in a single fixed size Java array? Or would you have multiple arrays or linked lists to hold the blocks? Please elaborate.

For this assignment you do not need to do Java coding, but should have an understanding of arrays and linked lists.

Submit your work as a PDF file via Github.

ECON381 Fall 2024

Annex

Consider the following Java class for the representation of keys.

```
public class OkeyKey {

    // Enum to represent the four colors in Okey
    public enum Color {
        RED, BLACK, BLUE, YELLOW
    }

    // Attributes of an Okey key
    private final int number;
    private final Color color;

    // Constructor
    public OkeyKey(int number, Color color) {
        if (number < 1 || number > 13) {
            throw new IllegalArgumentException("Number must be between 1 and 13.");
        }
        if (color == null) {
            throw new IllegalArgumentException("Color cannot be null.");
        }

        this.number = number;
        this.color = color;
    }

    // Getters
    public int getNumber() {
        return number;
    }

    public Color getColor() {
        return color;
    }

    // toString method for better representation
    @Override
    public String toString() {
        return "OkeyKey{" +
                "number=" + number +
                ", color=" + color +
                '}';
    }

    // Equals and hashCode for comparison and use in collections
    @Override
```

```
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        OkeyKey okeyKey = (OkeyKey) o;

        if (number != okeyKey.number) return false;
        return color == okeyKey.color;
    }

    @Override
    public int hashCode() {
        int result = number;
        result = 31 * result + color.hashCode();
        return result;
    }
}
```