CS1020 Sit-in Lab 02 B - Number Card Game

Semester 2 AY2013/2014

Background

A card game maker has created a new card game that is played between 2 or more players. Hesitant to release an untested game, he wants to run a computer simulation of the game before deciding whether to start production or not. You are the programmer hired to help him write the computer simulation.

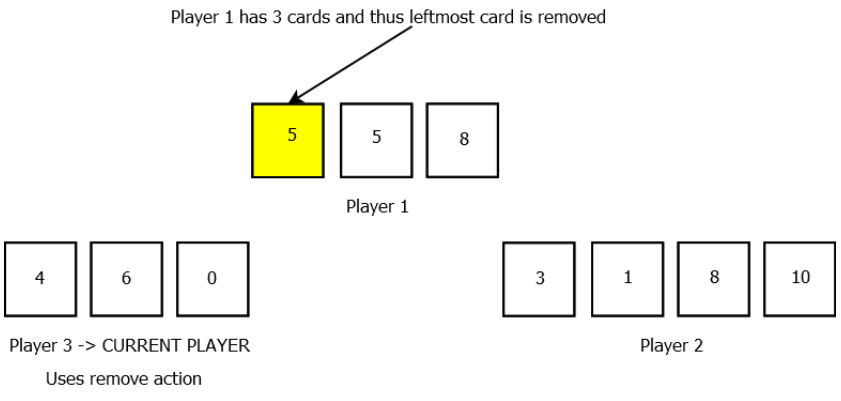
Use Java ArrayList to help you in this task. **Not using ArrayList will result in marks being halved. Using any other type of arrays instead of ArrayList will also result in marks being halved!**

Problem Description

The card game is played among N players. Each card is represented by an integer which is also the card's score. The scores range from 0 to 10. At the beginning of the game, each player is dealt a hand of 5 cards which is held from left to right and in such a way the card score cannot be seen by the other players. The players then sit in a circle clockwise from player 1 to player N and the game proceeds in rounds. In each round the players will perform one action starting from the 1st player and proceeding to the Nth player. The player can perform one of the following 2 actions during each round.

1. **Remove a card** from another player. The rules of card removing are as follows:
   (a) Can only remove card from player **clockwise next** to the current player.
   (b) Card is removed only if that other player has > 1 card. If this condition is not fulfilled no card will be removed.
   (c) Only leftmost card is removed.

An example of removing card action is shown below:



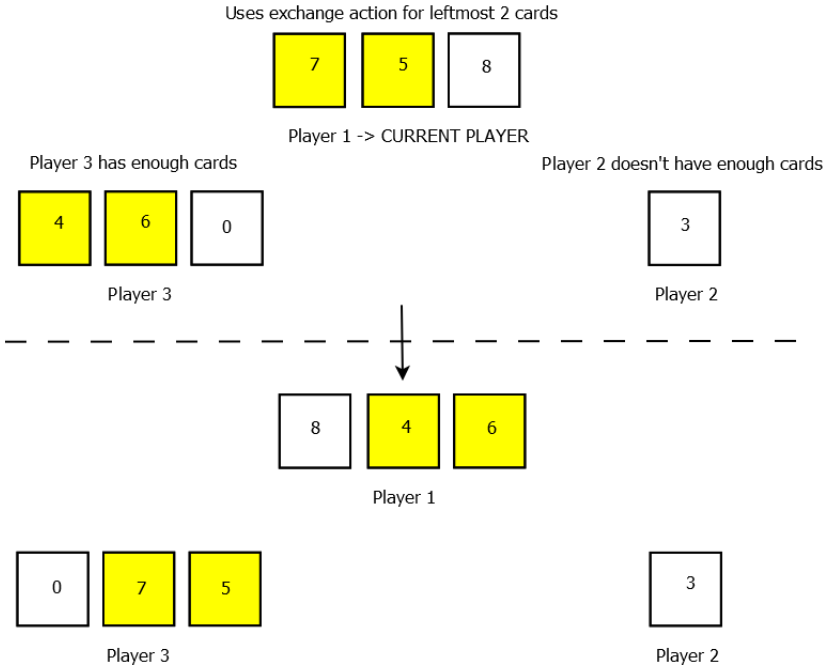
2. **Exchange cards** with another player. The rules for exchanging cards are as follows:

   (a) Pick leftmost K (K $>$ 0) number of cards from current player's hand.

   (b) Find the 1st player (call him P) in a **clockwise fashion** from the current player that has number of cards $\geq$ K. If P is not found, no action will be taken.

   (c) Take the K picked cards from current player's hand and place them in the same order to the rightmost end of P's hand. Next, take the leftmost K cards from P's hand and place them in the same order to the rightmost end of the current player's hand.

An example of exchanging cards action is shown below:



After a decided upon number of rounds, one card will be randomly picked from each player's hand. The player's final score is then the score of that card. The winner is the one with the highest final score. If there are more than 1 player having the same highest final score, there is no winner.

An example of a game played with 3 players and lasting 1 round is shown below:

```
Start of Game
player 1 - 1,1,2,3,5 <- initial cards for player 1
player 2 - 2,7,6,8,6 <- initial cards for player 2
player 3 - 9,4,9,9,0 <- initial cards for player 3

Round 1
player 1 -> Remove action

player 1 - 1,1,2,3,5
player 2 - 7,6,8,6  <- 2 is removed
player 3 - 9,4,9,9,0

player 2 -> pick 2 cards from leftmost for exchanging.

player 1 - 1,1,2,3,5
player 2 - 8,6,9,4  <- 9,4 are added
player 3 - 9,9,0,7,6 <- 7,6 are added

player 3 -> pick 3 cards from leftmost for exchanging.

player 1 - 3,5,9,9,0 <- 9,9,0 are added
player 2 - 8,6,9,4
player 3 - 7,6,1,1,2 <- 1,1,2 are added

End of Game (* indicates the card picked for each player)
player 1 - 3,5,*9,9,0
player 2 - 8,6,9,*4
player 3 - 7,*6,1,1,2

The winner is player 1 with a score of 9
```

Input

The 1st line in the input is N ($N \geq 2$) which gives the number of players. *Use descriptive variable name instead of N in your program. This applies for the other descriptors here.* The 2nd line is M ($M \geq 1$), the number of rounds the game is played. The next N lines consists of 5 integer values each. These integer values are separated by a single whitespace and represents the initial 5 cards dealt to each player from player 1 to player N respectively. The next N lines after that are the actions taken by each player from player 1 to the player N respectively for the first round. This is repeated for another M-1 times for the rest of the rounds. The format of the actions are as follows:

- Remove card action: R
- Exchange card action: E <leftmost K cards>

The final N lines are the card positions of the cards "randomly" picked for each player at the end of the game. Card position is expressed from leftmost to rightmost starting from 1. An example input is given below representing the game play above.

```
3
1
1 1 2 3 5
2 7 6 8 6
9 4 9 9 0
R
E 2
E 3
3
4
2
```

Output

Print the cards in each player's hand from player 1 to player N each on a separate line. For each player print his cards from left to right (separated by a comma, WITHOUT any space). Finally print the winning player number followed by the final score of the player separated by a comma, WITHOUT any space. If there is no winner, print **nowinner**. For the example input, the output is as follows:

```
3,5,9,9,0
8,6,9,4
7,6,1,1,2
1,9
```

Skeleton Program

Your program is to be named **NumberCardGame.java** (do not change this name). A skeleton program is provided, but you can change it any way or add new class(es) and method(s) as you deem fit. **Write all your code in NumberCardGame.java, do not create new files!** The skeleton program is as follows:

```
import java.util.*;

/* Service class representing a player's hand of cards */
class PlayerHand {
        private ArrayList<Integer> cardList;

}

/* Client class to simulate the number card game */
public class NumberCardGame {

        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
        }
}
```

Testing your program

The following **sample** input and output files are in your plab account:

```
NumberCardGameTC1.in, NumberCardGameTC2.in, NumberCardGameTC3.in
NumberCardGameTC1.out, NumberCardGameTC2.out, NumberCardGameTC3.out
```

NumberCardGameTC1.in, NumberCardGameTC2.in and NumberCardGameTC3.in are the input test cases, while NumberCardGameTC1.out, NumberCardGameTC2.out and NumberCardGameTC3.out are the expected output for the respective test cases.

After you have compiled your program, to test it with say NumberCardGameTC1.in, you type:

```
java NumberCardGame < NumberCardGameTC1.in
```

Grading Scheme

1. Program correctness = 70 marks, Design = 20 marks, Programming = 10 marks
2. No marks awarded if the program does not compile.
3. Marks will be deducted if **student particulars and program description** are not filled up in the top portion of the source code.
4. There are 10 test cases. Each test case is worth 7 marks.
5. Marks will be halved if ArrayList is not used or any other types of arrays is used.
6. Things to look out for under design
   (a) correct usage of programming constructs (eg use correct type for variables)
   (b) Not overly complicated logic
   (c) No redundant logic
7. Things to look out for under programming style
   (a) Meaningful comments (including pre- and post-condition description if necessary)
   (b) Proper indentation
   (c) Meaningful identifiers