Ira A. Fulton Schools of Engineering

school of computing, informatics, & decision systems engineering

CSE 110 – Assignment #7

Maximum points: 20 pts

Topics

- Array (Chapter 6)
  - Array object as an instance variable
  - Array operations

Your programming assignments require **individual work** and effort to be of any benefit. Every student must work independently on his or her assignments. This means that every student must ensure that neither a soft copy nor a hard copy of their work gets into the hands of another student. Sharing your assignments with others in any way is **NOT** permitted. Violations of the University Academic Integrity policy will not be ignored. The university academic integrity policy is found at http://www.asu.edu/studentlife/judicial/integrity.html

Use the following Guidelines:

- Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
- Keep identifiers to a reasonably short length.
- User upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.

Important Note:

All submitted assignments must begin with the descriptive comment block. To avoid losing trivial points, make sure this comment header is included in every assignment you submit, and that it is updated accordingly from assignment to assignment. **(If not, -1 Pt)**

```
//*******************************************************
// Name: your name
// Title: title of the source file
// Description: Write the description in your words.
// Time spent:  how long it took you to complete the assignment
// Date: the date you programmed
//*******************************************************
```

Part# 1: CodingBat.com Exercise: (5 pts)

Go to the following sites and do the exercises. If you solve the question, post the code as a comment in the part #2 code.

a) swapEnds: https://codingbat.com/prob/p118044
b) commonEnd: https://codingbat.com/prob/p191991
c) countEvens: https://codingbat.com/prob/p162010
d) sum13: https://codingbat.com/prob/p127384
e) fix34: https://codingbat.com/prob/p159339

Note:

The answers to the five questions (a through e) above should be typed in the block of comments in the **CardList.java** file such as;

```
/*** Part#1 Asnswer
   a) public int[] swapEnds(int[] nums) {
      ...
      }

   b) …
   …
*/
```

Part#2: Programming (15 pts)

Your assignment is to write and submit two java files. One is a class definition named **CardList** (CardList.java), and another is the test driver program, **Assignment7.java**.

1) The CardList is a class to keep the trump cards, and it has the two instance variables: **`cards`** and **`history`**.

CardList
- cards : char[] - history : String
+ CardList(int) + flip() : void + shift() : void + shuffle() : void + change(int size) : void - shuffleIndex() : int[] + getHistory () : String

The **`cards`** is an array object storing characters among A234567890JQK. The 'A' is for Ace, '2' is for two, '3' is three, ... '0' is for ten, 'J' is for Jack, 'Q' is for queen, and 'K' is for king like trump cards. The **`history`** is the record of updates.

The class must include the constructors and methods below: (If your class does not contain any of the following methods, points will be deducted). The public and private access modifiers are represented as "+" and "-" respectively.

1. `CardList(int numberOfCard)`: **(2 pts)**
   The constructor is to create an char-type array with the length of first input parameter. Each element in the array is initialized as a random character such as `cards[0]` = '3', `cards[1]` = 'Q', and `cards[2]= 'A'`.

2. `flip():void` **(2 pts)**
   This method is to flip the order of elements in `cards`. For example, the `cards` is flipped from {'3', '4', '5', '6', '7', '8'} to {'8', '7', '6', '5', '4', '3'} after this method is called.
   *) When it is called, update the **`history`** by adding the new list of cards and name of operation (look at the getHistory () below).

3. `shift():void` **(2 pts)**
   This method is to shift the position of each element in `cards` one by one. For example, the `cards` {'5', '7', '8', '6', '4'} is updated to {'7', '8', '6', '4', '5'} by `shift()`. The element out of range is moved to the end of array such as

`cards[0]` →`cards[-1]` (this is out of range) → `cards[cards.length-1]`.
*) When it is called, update the **`history`** by adding the new list of cards and name of operation (look at the getHistory () below).

4. `shuffle():void` **(2 pts or 4 pts)**
   (2 pts without `shffleIndex()`)
   This method is to shuffle all elements in `cards` randomly. *) When it is called, update the **`history`** by adding the new list of cards and name of operation (look at the getHistory () below).
   (4 pts with `shffleIndex()`)
   The following can be realized with the completed version of shuffleIndex() below. Any elements **MUST NOT** be in the same position as before. For example, it is OK to shuffle the `cards` from {'0', 'A', '2', '3', '4'} to {'4', '2', 'A', '0', '3'}. However, {'4', 'A', '2', '0', '3'} is not accepted because 'A' and '2' are not changed from the 2nd and 3rd positions. Before coding, look at and use the **`shuffleIndex()`** below.
   When more than half of elements have a same card, the output may looks invalid.
   Tips: Calculate the shuffuled index list such as
   `int[] pos = shuffuleIndex();`
   and update each card `cards[i]` using the value of `card[ pos[i] ];`
   For example, the pos[0] must be some number except 0, so we can use the pos[0] as the index of swapping target.

5. `change(int num)`: **(2 pts)**
   This method is to change some of cards by replacing random values among A234567890JQK. The number of cards to change is defined by the input `num`. Always the change starts from the first element `cards[0]` to `cards[num-1]`
   The `change(3)` changes three cards such as from {'Q', 'J', '4', '5', 'A'} to {'1', '5', '3', '5', 'A'}, because the high-lightened three items are changed. In the method, check the validity (positive and less than the length of cards) of `num` first, and if it is not valid, do nothing.
   *) When it is called, update the **`history`** by adding the new list of cards and name of operaution (look at the getHistory () below).

6. `shuffleIndex ():int[]` **(To get 4-pts in the shuffle())**
   This method is to return an array of shuffled **integers**, which will be used to shuffle all cards. **The size is the same as the cards.length.** Any element in the array **MUST NOT** be the same as the position index. Suppose that the input size is 5. The output should be initialized as {0, 1, 2, 3, 4}, and shuffle elements repeatedly and check the validity before returning it. For example, {1, 3, 4, 2, 0} is valid, because the output[0] = 1, output[1] = 3, ... output[4] = 0. All values are different from the index (position value).

However {2, 1, 3, 4, 0} is not accepted because the 1 is in the position [1] is the array. By using while-loop, swap random two elements until all become different from the index. Use this private method in `shuffle()`.

7. `getHisotry(): String` **(2 pts)**
   Return the `history`, which has all history of operations line by line ("\n" : line change). It is OK to use `Arrays.toString(cards)`.

```
[Q, 6, 7, 4, 8, A, 0]
[A, 3, 5, 4, 8, A, 0]: Change
[0, A, 8, 4, 5, 3, A]: Flip
[A, 8, 4, 5, 3, A, 0]: Shift
[8, 4, 3, A, 0, A, 0]: Shuffle
```

2) Write the test driver called **Assignment7.java**. **(1 pts)** Assignment7.java contains the main method to create a `CardList` instance object and test the methods in the class. The program will ask a user to enter one of the commands. Based on the user's choice, the program needs to perform corresponding operation. This will be done by calling (invoking) one of the methods you defined in the class. It will terminate when the user enters 'q'. The tester program provided in the last assignments may help. In addition to the main method, the Assignment7 class has a static method, `printMenu()`, to print out the following commands

```
Command Options
-----------------------------------
a: Create new cards
b: flip the cards
c: shift  the cards
d: shuffle the cards
e: change the cards
?: Display the menu again
q: Quit this program
```

Here is the description for each option:

a: ask & read the size of array, and create new object and display the elements
b: flip and display the history of operations
c: shift and display the history of operations
d: shuffle and display the history of operations
e: ask & read the size of cards to change, and display the history of operations.
?: display the menu
q: quits the program

Output Example

- **Input in red is not shown when you submit online**
- **This program uses many random values. So the output here should be different from your output.**

```
----------------
INPUT 1
----------------
a
5
b
c
d
e
3
?
q

----------------
YOUR OUTPUT 1
----------------
*** Start of Program ***

Command Options
-----------------------------------
a: Create new cards
b: flip the cards
c: shift the card
d: shuffle the cards
e: change the cards
?: Display the menu again
q: Quit this program


Please enter a command or type ? a
    a [Create new cards]
    [Input the size of cards]: 5
[8, Q, 2, 6, 5]

Please enter a command or type ? b
    b [flip the cards]
[8, Q, 2, 6, 5]
[5, 6, 2, Q, 8]: Flip

Please enter a command or type ? c
    c [shift the cards]
[8, Q, 2, 6, 5]
[5, 6, 2, Q, 8]: Flip
[6, 2, Q, 8, 5]: Shift
Please enter a command or type ? d
```

```
    c [shuffle the cards]
[8, Q, 2, 6, 5]
[5, 6, 2, Q, 8]: Flip
[6, 2, Q, 8, 5]: Shift
[8, 6, 2, 5, Q]: Shuffle

Please enter a command or type ? e
    e [Change the cards]
    [Input the number of cards to change]:3
[8, Q, 2, 6, 5]
[5, 6, 2, Q, 8]: Flip
[6, 2, Q, 8, 5]: Shift
[8, 6, 2, 5, Q]: Shuffle
[8, 4, 7, 5, Q]: Change

Please enter a command or type ? ?
Command Options
-----------------------------------
a: Create new cards
b: flip the cards
c: shift the card
d: shuffle the cards
e: change the cards
?: Display the menu again
q: Quit this program


Please enter a command or type ? q
*** End of Program ***
```

Use only the Java statements that have been covered in class to date. **DO NOT** use **ArrayList** or any other items out of the Chapter1-6 and 8. If in doubt, ask. If you use them, then you lose the points of task. Don't copy any code developed by others. Don't give your code to others. Don't use any algorithm, which you cannot understand. Your assignment file is checked by the MOSS (by Stanford Univ.), which is a program to detect cheatings.

/**

Submit your homework by following the instructions below:

**/

- Go to the course web site, and then click on the GradeScope on CANVAS.
- Submit your **Assignment7.java and CardList.java** files on-line.
- The **CardList.java** should have the following, in order:
    - In comments, the answers to questions presented in Part#1.
    - The working Java code requested in Part #2.
- The Assignment7.java and CardList.java files must have the header comments described in "Important Note" in the top page.
- The **Assignment7.java** and **CardList.java** files must compile and run as you submit it. You can confirm this by viewing your submission results.

Note: You may resubmit as many times as you like until the deadline, but we will only mark your last submission. **NO LATE ASSIGNMENTS WILL BE ACCEPTED.**