TCP1201 Objected-Oriented Programming and Data Structures Assignment

Trimester 2, Session 2019/2020
Faculty of Computing and Informatics
Multimedia University

Deadline: 9 Feb 2020 (Sunday), 11:59pm

Outline

1. This assignment contributes **20%** of total subject mark.
2. The assignment consists of **one** question only.
3. Max **two** students form a group to make one submission.
4. **Interview** will be conducted after the due date.
5. Submit the assignment **in time** even though it is incomplete. Make sure the program is compile-able by Java 8. Zero (0) mark will be given for late submission. Request for deadline extension won't be entertained as the lecturer requires sufficient time to assess and release the mark before the final exam. Exact interview date and time will be scheduled after the deadline.

Simple FreeCell

1. This game is a simplified version of FreeCell. You may play a few FreeCell online to have some ideas.
2. This game requires only one deck of 52 unique cards. There are four suits namely diamonds (D), clubs (C), hearts (H), and spades (S). as shown below (from top to bottom):

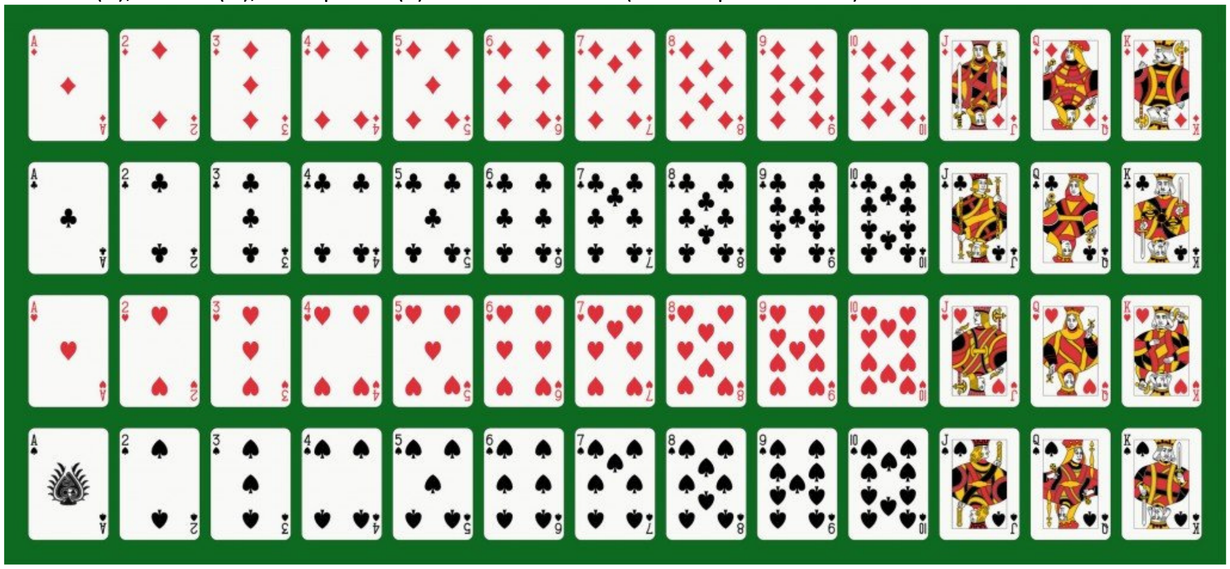
Photo Credit: Mannaggia / Fotolia

Each card has a point value based on its face. For example, card with face 2 has 2 points. For A, J, Q, and K, its point values are listed in the following table.

Face	A	10 (X)	J	Q	K
Point	1	10	11	12	13

3. The game has four (4) Piles and nine (9) Columns.
4. At the beginning of a game, the Piles are empty while Columns 1-8 are placed with random cards. A sample Command Prompt output is shown below.

```
Pile   c: []
Pile   d: []
Pile   h: []
Pile   s: []
```

```
Column 1: [c9, h5, dQ, d6, c7, s8, hQ]
Column 2: [h6, sJ, d2, hX, d5, cA, hA]
Column 3: [d9, dA, h7, d8, s3, sX, h2]
Column 4: [h8, sA, sK, hK, c6, s4, h9]
Column 5: [s5, c2, s7, s2, cQ, c5]
Column 6: [dK, cJ, cX, hJ, c4, c8]
Column 7: [dX, s6, c3, s9, d7, dJ]
Column 8: [d3, cK, sQ, h4, h3, d4]
Column 9: []
```

5. The objective of the game is to move all the cards to the piles, based on a set of move rules.
6. The game is solved when the Columns are empty while the Piles have cards placed in ascending order following the suits and faces.

```
Pile   c: [cA, c1, c2, c3, c4, c5, c6, c7, c8, c9, cX, cJ, cQ, cK]
Pile   d: [dA, d1, d2, d3, d4, d5, d6, d7, d8, d9, dX, dJ, dQ, dK]
Pile   h: [hA, h1, h2, h3, h4, h5, h6, h7, h8, h9, hX, hJ, hQ, hK]
Pile   s: [sA, s1, s2, s3, s4, s5, s6, s7, s8, s9, sX, sJ, sQ, sK]
Column 1: []
Column 2: []
Column 3: []
Column 4: []
Column 5: []
Column 6: []
Column 7: []
Column 8: []
Column 9: []
```

Move Rules

1. Card(s) shall be moved from the end (right-most) of a Column to the end of another Column or the end of another Pile.
2. No card shall be moved from Piles.
3. When moving a card to a Pile, the card must be placed at the end of the Pile of the matching suit and the moved card must be one point bigger than the last (right-most) card at the Pile.

Example	Result
`Pile   c: []`	`Pile   c: []`
`Pile   d: []`	`Pile   d: []`
`Pile   h: [hA]`	`Pile   h: [hA, h2]`
`Pile   s: []`	`Pile   s: []`
`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`	`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`
`Column 2: [h6, sJ, d2, hX, d5, cA]`	`Column 2: [h6, sJ, d2, hX, d5, cA]`
`Column 3: [d9, dA, h7, d8, s3, sX, h2]`	`Column 3: [d9, dA, h7, d8, s3, sX]`
`Column 4: [h8, sA, sK, hK, c6, s4, h9]`	`Column 4: [h8, sA, sK, hK, c6, s4, h9]`
`Column 5: [s5, c2, s7, s2, cQ, c5]`	`Column 5: [s5, c2, s7, s2, cQ, c5]`
`Column 6: [dK, cJ, cX, hJ, c4, c8]`	`Column 6: [dK, cJ, cX, hJ, c4, c8]`
`Column 7: [dX, s6, c3, s9, d7, dJ]`	`Column 7: [dX, s6, c3, s9, d7, dJ]`
`Column 8: [d3, cK, sQ, h4, h3, d4]`	`Column 8: [d3, cK, sQ, h4, h3, d4]`
`Column 9: []`	`Column 9: []`
`Command > 3 h2 h`	`Command >`

4. When moving a card to another Column, the suit and color are always ignored. The moved card must be placed to a Column where its last card is one point bigger than the moved card.

Example	Result
`Pile   c: [cA]`	`Pile   c: [cA]`
`Pile   d: []`	`Pile   d: []`
`Pile   h: [hA, h2]`	`Pile   h: [hA, h2]`
`Pile   s: []`	`Pile   s: []`

`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`	`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`
`Column 2: [h6, sJ, d2, hX, d5]`	`Column 2: [h6, sJ, d2, hX, d5]`
`Column 3: [d9, dA, h7, d8, s3, sX]`	`Column 3: [d9, dA, h7, d8, s3, sX]`
`Column 4: [h8, sA, sK, hK, c6, s4, h9]`	`Column 4: [h8, sA, sK, hK, c6, s4, h9]`
`Column 5: [s5, c2, s7, s2, cQ, c5]`	`Column 5: [s5, c2, s7, s2, cQ, c5, d4]`
`Column 6: [dK, cJ, cX, hJ, c4, c8]`	`Column 6: [dK, cJ, cX, hJ, c4, c8]`
`Column 7: [dX, s6, c3, s9, d7, dJ]`	`Column 7: [dX, s6, c3, s9, d7, dJ]`
`Column 8: [d3, cK, sQ, h4, h3, d4]`	`Column 8: [d3, cK, sQ, h4, h3]`
`Column 9: []`	`Column 9: []`
`Command > 8 d4 5`	`Command >`

5. If you have a group of consecutive right-most cards whereby the faces are in order, you may move all of them to another Column in just one command.

Example	Result
`Pile   c: [cA]`	`Pile   c: [cA]`
`Pile   d: []`	`Pile   d: []`
`Pile   h: [hA, h2]`	`Pile   h: [hA, h2]`
`Pile   s: []`	`Pile   s: []`
`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`	`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`
`Column 2: [h6, sJ, d2, hX, d5]`	`Column 2: [h6, sJ, d2, hX, d5, h4, h3]`
`Column 3: [d9, dA, h7, d8, s3, sX]`	`Column 3: [d9, dA, h7, d8, s3, sX]`
`Column 4: [h8, sA, sK, hK, c6, s4, h9]`	`Column 4: [h8, sA, sK, hK, c6, s4, h9]`
`Column 5: [s5, c2, s7, s2, cQ, c5, d4]`	`Column 5: [s5, c2, s7, s2, cQ, c5, d4]`
`Column 6: [dK, cJ, cX, hJ, c4, c8]`	`Column 6: [dK, cJ, cX, hJ, c4, c8]`
`Column 7: [dX, s6, c3, s9, d7, dJ]`	`Column 7: [dX, s6, c3, s9, d7, dJ]`
`Column 8: [d3, cK, sQ, h4, h3]`	`Column 8: [d3, cK, sQ]`
`Column 9: []`	`Column 9: []`
`Command > 8 h4 2`	`Command >`

6. If you have a group of consecutive right-most cards whereby the suits and faces are in order, you may move all of them to the Pile in just one command.

Example	Result
`Pile   c: [cA]`	`Pile   c: [cA]`
`Pile   d: []`	`Pile   d: []`
`Pile   h: [hA, h2]`	`Pile   h: [hA, h2, h3, h4]`
`Pile   s: []`	`Pile   s: []`
`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`	`Column 1: [c9, h5, dQ, d6, c7, s8, hQ]`
`Column 2: [h6, sJ, d2, hX, d5, h4, h3]`	`Column 2: [h6, sJ, d2, hX, d5]`
`Column 3: [d9, dA, h7, d8, s3, sX]`	`Column 3: [d9, dA, h7, d8, s3, sX]`
`Column 4: [h8, sA, sK, hK, c6, s4, h9]`	`Column 4: [h8, sA, sK, hK, c6, s4, h9]`
`Column 5: [s5, c2, s7, s2, cQ, c5, d4]`	`Column 5: [s5, c2, s7, s2, cQ, c5, d4]`
`Column 6: [dK, cJ, cX, hJ, c4, c8]`	`Column 6: [dK, cJ, cX, hJ, c4, c8]`
`Column 7: [dX, s6, c3, s9, d7, dJ]`	`Column 7: [dX, s6, c3, s9, d7, dJ]`
`Column 8: [d3, cK, sQ]`	`Column 8: [d3, cK, sQ]`
`Column 9: []`	`Column 9: []`
`Command > 2 h4 h`	`Command >`

Commands to Program

The game shall support the following three (3) commands:

1. r – Restart a new game.
2. x – Exit the game.
3. *source card destination* – Move *card* from *source* Column to *destination* Column or Pile.

Submission Format

1. A zip file named "*TT0X_Student1Name_ Student2Name*.zip" where *TT0X* is your section and *StudentName* is your name.

2. The zip should contain your **source code files** (*.java) and **Javadoc documentation**.
3. Upload the zip to MMLS before the deadline.

Mark Sheet

Item	Student 1	Student 2
1 Late submission or Plagiarism Zero (0) mark for the assignment.		
2 Interview & Presentation Zero (0) mark for assignment if fail to present for interview.		
Mark will be granted for all the following items only if it works and the student can explain the code.		
3 Program Execution [12m] Zero (0) mark for the assignment if code is unable to compile or run.		
3.1 Start and exit a game [2m] (a) Can exit the game by issuing command 'x' [1m] (b) Can restart the game by issuing command 'r' (cards must be reshuffled randomly) [1m]	/ 2	/ 2
3.2 Move cards from a column to another column [3m] (a) Move many cards (at a time) following rules [3m] (b) Move a card following rules [2m] (c) Move a card without following rules (illegal) [1m]	/ 3	/ 3
3.3 Move cards from a column to a pile [3m] (a) Move many cards (at a time) following rules [3m] (b) Move a card following rules [2m] (c) Move a card without following rules (illegal) [1m]	/ 3	/ 3
3.4 Error handling on user input [3m] (a) Report invalid command [0.5m] (b) Report invalid column [0.5m] (c) Report invalid Check card [0.5m] (d) Report invalid Check pile [0.5m] (e) Report invalid Check move [1m]	/ 3	/ 3
3.5 Game displays faces A, X, J, Q, King in uppercase but player can play by entering card name in lowercase. [1m]	/ 1	/ 1
Program Execution Subtotal	**/ 12**	**/ 12**

4 Program Design [8m]		
4.1 Use adequate data structures (3m) (a) Array, ArrayList, LinkedList, and/or Vector [1m] (b) Stack and/or Queue [1m] (c) Map and/or Set [1m]	/ 3	/ 3
4.2 Create a custom OrderedStack class to represent a pile [2m] (a) The pop method shall check whether the new card pushed into the OrderedStack follows the rules. [1m] (b) Throw an exception if the new card does not follow the rule. [1m]	/ 2	/ 2
4.3 Style (indentation and naming) [1m] (a) Good [1m] (b) Average [0.5m] (c) Poor [0m]	/ 1	/ 1
4.4 Java Documentation using Javadoc [2m] (a) Adequate javadoc documentation for public classes, public fields, public methods, parameters, and return types. [2m] (b) Missing javadoc documentation for public classes, public constants, public methods, public method parameters, and/or public method return types. [1m]	/ 2	/ 2
Program Design Subtotal	**/ 8**	**/ 8**
5 Bonus [2 m] (a) GUI (JavaFX or Android), or (b) Auto-solve - Can fully solve a game automatically by issuing command 's'. Must show every move step by step.	**/ 2**	**/ 2**
Total	**/ 20**	**/ 20**