420-B31

***Programming III (420-B31-HR)***

***Assignment 3 - The Stack and Queue ADTs***

***War Game***

Date assigned: Monday, November 5, 2018

Date due – Test cases and Class diagram: Monday, November 5, 2018 in lab

Date due – code and Junit tests: Monday, November 26, 2018

***Objectives:***

After completing this assignment, the student should be able to:

1. Create and use a Stack object.
2. Create and use a Queue object.
3. Create thorough test cases in jUnit.

***Problem Specifications:***

In this assignment, you are going to create a card game and jUnit test cases to test the card game. The specifications for the game is as follows.

# War

**Players**: 2

**Objective**: To force the other player to run out of cards.

**Setup**: Shuffle a deck of cards and deal out the entire deck – 26 cards to each player. Each player's cards are kept in a face-down pile.

**Play**: In each round, each player turns up the top card of his pile. The player with the higher-ranking cards takes both cards and adds them to the bottom of his pile. A tie is resolved by a "war": each player plays three cards from his pile face-down and another face-up. The high card among these two last wins all ten cards. Subsequent ties are handled similarly, with the eventual winner of the round taking an even larger pile of cards.

**Game End**: If a player is unable to play a card (even during a war), he loses.

# Design

You will create a **Card** class and a **Deck** class to be used by the game. The **toString()** method will return the rank and suit of the card.

The **Card** class will represent a single playing card and have a rank and a suit.

The **Deck** class will represent a card deck of 52 cards. The constructor will create an unshuffled **list** of all 52 cards. The **shuffle()** method with rearrange the cards into a randomly sorted **queue**. The **deal()** method will remove the first card from the queue and return it to the client class. The **size()** method will return the number of cards that have not yet been dealt. For testing purposes, you will also need the ability to not shuffle the cards, and instead create a test deck in the order you require to test your scenario.

The **War** class will contain a **Deck** object, two Strings representing the names of the two players, two queues representing the two hands of the two players and a stack representing the kitty accumulated when there is a tie. The **start()** method will deal out the two hands. The **play()** method will "play a round" according to the rules established above.

The **WarGame** class will play **War**.

The java files for the various classes of stacks and queues which we have studied are on Moodle. You may use whichever of these implementations of stacks and queues that you feel is most appropriate.

# Partial Sample runs:

## WarGame

Welcome to the game of war.

The object of the game is to force the other player to run out of cards.

All the cards are dealt at the beginning of the game.

Each play both players lay the top card of their pile face up. The player with the highest rank card, puts both cards on the bottom of his pile.

If both cards have the same rank, each player plays three cards face down and plays another round.

The winner of the tie-breaking round gets all the played cards (the cards in the tie, the six face down and the two in the tie-breaking play.)

Both hands have been dealt.

Player 1 has 26 cards to start.

Player 2 has 26 cards to start.

Player 1 plays 7C. Player 2 plays 10D.

Player 2 won this hand.

Player 1 has 25 cards left.

Player 2 has 27 cards left.

Hit any key to continue or Q to quit.

Player 1 plays JD. Player 2 plays 10S.

Player 1 won this hand.

Player 1 has 26 cards left.

Player 2 has 26 cards left.

Hit any key to continue or Q to quit.

Player 1 plays QC. Player 2 plays KD.

Player 2 won this hand.

Player 1 has 25 cards left.

Player 2 has 27 cards left.

Hit any key to continue or Q to quit.

Player 1 plays 4S. Player 2 plays KC.

Player 2 won this hand.

Player 1 has 24 cards left.

Player 2 has 28 cards left.

Hit any key to continue or Q to quit.

Player 1 plays 2H. Player 2 plays QD.

Player 2 won this hand.

Player 1 has 23 cards left.

Player 2 has 29 cards left.

Hit any key to continue or Q to quit.

Player 1 plays KS. Player 2 plays 9D.

Player 1 won this hand.

Player 1 has 24 cards left.

Player 2 has 28 cards left.

Hit any key to continue or Q to quit.

Player 1 plays JS. Player 2 plays JC.

It's a tie! Each player lays 3 cards face down

Kitty has 8 cards.

Player 1 has 20 cards left.

Player 2 has 24 cards left.

Hit any key to continue or Q to quit.

Player 1 plays 3C. Player 2 plays 10H.

Player 2 won this hand.

Player 1 has 19 cards left.

Player 2 has 33 cards left.

Hit any key to continue or Q to quit.

Q

Thank you for playing!

***To be handed in:***

1. A Word document containing the list of test case scenarios for the program should be uploaded to Moodle. A simple numbered list of test cases is acceptable, where each test describes what scenario you will test. You will also need to think about and describe in your test case document how you are going to test these scenarios. (Provide some sample Decks that will allow you to test the scenarios.) You may bring your test cases to the teacher prior to the assignment due date to get feedback on the tests.
2. Complete junit test cases to test the scenarios developed, using test decks.
3. The code for the assignment. The ***username*\_B31\_A03\_Stacks\_Queues** folder containing the java files for the project, the junit tests and the test case plan should be uploaded to **Moodle**.

***Marking Scheme:***

|  |  |  |
| --- | --- | --- |
|  | **Mark** | **Out of** |
| Class Diagram |  | 5 |
| Test case scenarios |  | 10 |
| Junit Test Cases |  | 25 |
| Card class |  | 5 |
| Deck class |  | 15 |
| War class |  | 40 |
| WarGame class |  | 10 |
| War program execution |  | 10 |
| Organization |  | 5 |
| **Total** |  | **125** |

***Organization Marks:***

Marks will be given for organization. This includes:

* naming files and folders according to the department standards
* giving meaningful names to variables, classes, objects and methods
* formatting and indenting Java classes using the Eclipse format tool
* submitting the assignment in correctly on **Moodle**
* including all required files in the submitted assignment folder