Assignment 1
War Card Game

CSIS 3475

War card game overview

- War (US) or Battle (UK) is a card game typically played by two players. It uses a standard 52 card deck in decreasing order: A K Q J 10 9 8 7 6 5 4 3 2.
- The objective of the game is to win all cards.
- The deck is divided evenly among the players, giving each a **cards to be played** stack. In unison, each player reveals the top card of their deck—this is a "battle"—and the player with the higher card takes both of the cards played and moves them to their **won** card stack. Aces are high, and suits are ignored.
- If the two cards played are of equal value, then there is a "war". Both players place the next three card of their pile face down, and then another card face-up on the **war** stack. The owner of the higher face-up card wins the war and adds all ten cards on the table to their **won** deck. If the face-up cards are again equal then the battle repeats with another set of face-down/up cards. This repeats until one player's face-up card is higher than their opponent's.
- When there are no more cards in the **cards to be played** stack, the **won** stack is shuffled and then used as the **cards to be played** stack. Basically, the **won** stack cards are moved to the **cards to be played** stack after shuffling.
- If a player runs out of cards during a battle or war, that player immediately loses and the game ends.
- In this variant of the game, note that each player has three stacks.
- For more information, see https://en.wikipedia.org/wiki/War_(card_game) and https://www.pagat.com/war/war.html
- A nice graphical simulation is at https://cardgames.io/war, but you may want to be careful with this site.

Assignment

- Download **Assignment 1.zip** and import it into an Eclipse workspace using the standard instructions.
- Create the game using ArrayStack.
  - Complete the project **War Game Using Stack**
- Create the game using Deque.
  - Complete the project **War Game Using Queue**
- Submit the completed projects using the standard submission instructions
  - Export the projects to a zip archive named **Assignment 1 YourName.zip** where YourName must be your first initial and last name.
  - You MUST use the submission instructions exactly or you will lose marks.
  - You MUST name the archive correctly or you will lose marks.
  - For example, for Michael Hrybyk
    - Assignment 1 MHrybyk.zip

War Game Using Stack

- This project has all of the files needed to complete the program.
  - WarCardGame.java
    - main program and logic
    - creates main deck, shuffles it, and deals the cards initially to two players
    - loop logic for battles and wars
    - determines when game is over (loser declared)
  - ArrayStack.java – used to build card decks
    - implements StackInterface.java
  - Deck.java
    - extends ArrayStack to form a card deck
    - method to create a standard deck
    - shuffle method
  - Player.java
    - contains the three decks (cardsToBePlayed, wonCards, and war)
    - has methods to add and remove cards
    - implements PlayerInterface.java
  - Card.java
    - a single card with a card type, suit, number, and rank
    - CardType.java – enum for a number or face card
    - Suit.java – enum for card suits

War Game Using Deque

- This project has all of the files needed to complete the program.
  - WarCardGame.java
    - main program and logic
    - creates main deck, shuffles it, and deals the cards initially to two players
    - loop logic for battles and wars
    - determines when game is over (loser declared)
  - LinkedDeque.java – used to build card decks
    - implements DequeInterface.java
  - Deck.java
    - extends LinkedDeque to form a card deck
    - method to create a standard deck
    - shuffle method
  - Player.java
    - contains the three decks (cardsToBePlayed, wonCards, and war)
    - has methods to add and remove cards
    - implements PlayerInterface.java
  - Card.java
    - a single card with a card type, suit, number, and rank
    - CardType.java – enum for a number or face card
    - Suit.java – enum for card suits

War Game Using Stack Tasks

- Complete each task below in the project. Make sure you write code that tests each task thoroughly as you finish it.
- Task 1
  - Complete the ArrayStack implementation.
  - Similar to textbook, but no extended capacity or check integrity needed.
  - pop/peek returns null if empty rather than throwing an exception.
  - need to code the size() method.
- Task 2
  - In Deck.java, complete the createStandardCardDeck() method. This creates the full 52 card deck.
  - Study the shuffle method and understand how it works.
- Task 3
  - Implement the PlayerInterface methods in Player.java which uses three Decks.
  - You will need to use ArrayStack methods to complete this.
  - Do not change the toString() method in any class.
- Task 4
  - Using Player and Deck classes, implement the game in WarCardGame.java according to the rules laid out in the overview.
  - Test your code thoroughly. Output should correspond in format exactly to that contained in SampleOutput.txt
    - although due to random nature of the game and shuffling, each game played will have a different number of rounds and cards won/played.

War Game Using Deque Tasks

- Complete each task below in the project. Make sure you write code that tests each task thoroughly as you finish it.
- Task 1
  - Complete the LinkedDeque implementation.
  - Similar to textbook, but no extended capacity or check integrity needed.
  - getFront/Back and removeFront/Back methods return null if empty rather than throwing an exception.
  - need to code the size() method.
- Task 2
  - In Deck.java, complete the createStandardCardDeck() method. This creates the full 52 card deck.
  - Study the shuffle method and understand how it works.
- Task 3
  - Implement the PlayerInterface methods in Player.java which uses three Decks.
  - You will need to use LinkedDeque methods to complete this.
  - Do not change the toString() method in any class.
- Task 4
  - Using Player and Deck classes, implement the game in WarCardGame.java according to the rules laid out in the overview.
  - Test your code thoroughly. Output should correspond in format exactly to that contained in SampleOutput.txt
    - although due to random nature of the game and shuffling, each game played will have a different number of rounds and cards won/played.

Grading

Item	Marks
Project properly named and submitted	.2
All code properly formatted and commented	.4
Stack Project Tasks 1-2	1
Stack Project Task 3-4	1.2
Deque Project Task 1-2	1
Deque Project Task 3-4	1.2
Total	5