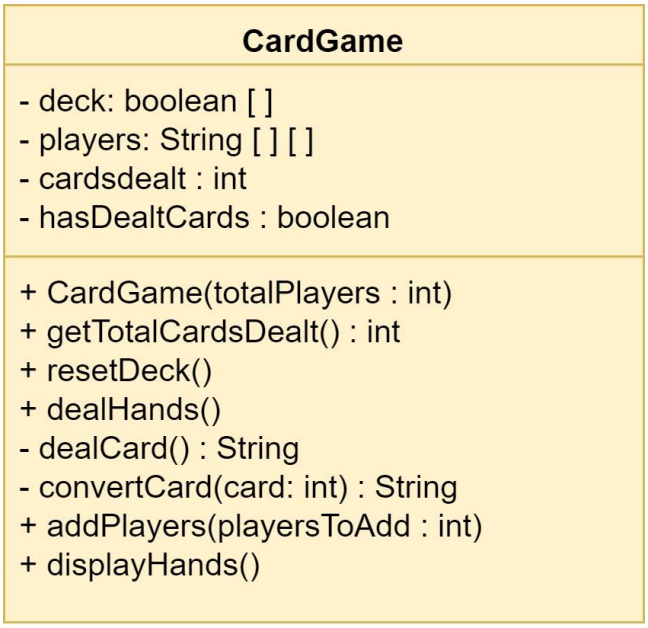
CompSci 251

Assignment 3

Due Oct. 1st, 2018

Program Specification

You will take your assignment 1 and turn it in to an object-oriented class following the UML diagram below. A few things have been added or removed, but most of the class stays the same. You are allowed to reuse your code from assignment 1, but keep in mind the keyword "static" should not be seen anywhere in the program (this is an object now, which means it maintains its own state). The only two variables allowed to be static are the ones I provided for assignment 1 that helped with card conversions. You are responsible for creating a driver for this program to test your code. I have provided two base files for you, but they are stripped of all information. A scanner object will NOT be needed for your CardGame class or the driver.



Implementation

- Driver
  - You are to write your own driver to test your code. You should instantiate a few CardGame objects and pass in the number of players for each game. Then use the public methods to test your code and see if you get correct output.

- public CardGame(totalPlayers: int)
  - Initialize your instance variables. A deck holds 52 cards. The total number of players allowed must be between 2 and 8. If a number passed in is not in range, the default size is 2. cardsDealt is initialized to 0. hasDealtCards is false when the deck has just been shuffled, and true when the cards get dealt. This ensures a deck is only allowed to deal cards after it is shuffled.

- getTotalCardsDealt()
    - returns the total number of cards dealt

- resetDeck()
    - Collect all cards, set cardsDealt to 0, and reset hasDealtCards to false indicating the cards can be dealt again.

- dealHands()
    - Deal out cards to each player in the game. You will need to call dealCard. The logic will be similar to assignment 1 with a new feature added. Now you will also need to check hasDealtCards. If the deck has already dealt cards and this method is called, you must display an error message to the console saying the cards have been dealt already and not deal out any more cards.

      Example of what driver could look like.

```
CardGame game1 = new CardGame(3);      //create cardgame with 3 players
game1.resetDeck();                     //make it so all cards available
game1.dealHands();                     //deal out hands to each player
game1.dealHands();              //can not do this, will print error message
game1.resetDeck();                     //"collect" all cards
game1.dealHands();                     //can now re-deal again
```

- dealCard()
    - Deal one card from the deck. The logic will be similar to assignment 1. Take note of the access permissions of this method.

- convertCard(card: int)
    - Return a string representation of a card. The logic will be similar to assignment 1. Take note of the access permissions of this method.

- displayHands()
    - Display all players hands. The logic will be similar to assignment 1.

More continued below

- addPlayers(playersToAdd: int)
    - A new method you must add. Add the specified number of players to the deck. Make sure the number passed in does not exceed the max number of players allowed and it does not reduce the total number of current players. If either of these two tests fail, do not copy anything and display an error message to the screen. You will need to create a new 2D String array, copy all contents from the current array to the new array in the same order, then deal out the cards to the new players. Don't forget to reassign your instance variable. This method should not be affected by the hasDealtCards boolean variable and if the cards have already been shown that is ok (grad students will handle this case). An example is below, note how player 1,2 and 3's hands do not change when 3 more players are added.

      ex.

```
Game One Players:
Player 1 hand:      7S     9C     2D     4S     3H
Player 2 hand:      8C     4H     JS     4C     JD
Player 3 hand:      KD     AS     7D     AD     8H
Game One Players has added 3 players.
Player 1 hand:      7S     9C     2D     4S     3H
Player 2 hand:      8C     4H     JS     4C     JD
Player 3 hand:      KD     AS     7D     AD     8H
Player 4 hand:      7H     5S     9H     7C     4D
Player 5 hand:      3S     8D     2C     10D    AH
Player 6 hand:      2S     JC     KS     AC     2H
```

Graduate Students

- boolean addPlayers(playersToAdd: int)
    - You will be responsible for adding a little more functionality to this method. Note the return type of the method will change for you from void to boolean, so it must return true/false. You are to add another instance variable *boolean hasShownHands* to your list of instance variables declared in the uml diagram. If the players have already shown their hands, the new players are not added to the game and you output a message saying so. You are not allowed to print out the message in this method though. It must be printed in your driver! The only way players can now be added is if the current players hands haven't been shown yet.

      ex.

```
Game One Players:
Player 1 hand:      7S     9C     2D     4S     3H
Player 2 hand:      8C     4H     JS     4C     JD
Player 3 hand:      KD     AS     7D     AD     8H

Attempting to add more players.
Cannot add more players, hands have been shown.  Please reshuffle.
```