CS 110 Java CardGame Part I (no GUI)
Card, Deck and **Hand (for a game of 31)**

NOTES:

- Collaboration policy for this program: This is individual work, but students may consult each other and TAs for this semester for discussing the approach.
  You must NAME significant collaborators at the head of the files.

- See the rules for the game. Anywhere in the document where there is a choice in the way the game is played, you decide what the rule will be. (Eg. whether an Ace is worth 1 or 11, or just 1, or just 11….)

- Arrange files in one folder called **CardDeckHand**
  Files to turn in: ONE zip file-- **CardDeckHand.zip** to include:

  Card.java **TestCard.java**
  Deck.java **TestDeck.java**
  Hand.java **TestHand.java**

Directions:
Build each ADT implementation and test independently before integrating. You will be required to use the java library "ArrayList" as indicated below, and in addition, you may (and should) use any other appropriate "collection" library. You can see all the operations by looking it up in the Sun documentation online. Googling "java ArrayList" should get you to a good link.
These client test programs output to the console.

****

1. Card.java + TestCard.java

Card.java implements the ADT:
ADT: The "**what**" needs to be accomplished
values: a suit and a rank, you may want to add the .jpg file name as well
operations: (discusssed in class, these are required, but you may need more)

constructor setting suit and rank
setSuit
setRank
setPictureName
getSuit
getRank
getPictureName

toString() // to show 2h 3h…10h, Jh, Qh, Kh, Ah etc.
equals: true if both the suit and the rank match

TestCard.java
Write a test program that creates at least 3 different cards, and try out each
of the functions.

2. Deck.java + TestDeck.java

Deck ADT: The "**what**" needs to be accomplished
values: a collection of 52 different cards of a standard deck with sequential property and the concept of the "top" card (or bottom if you see that in the rules!).
operations: (these are the required -- you may add more)
constructor (with all the 52 cards in it)
getTopCard (remove and return the one ("top") card on the deck)
shuffle (One easy way to shuffle is to take each card in the ArrayList and swap it with another card by randomly choosing another index. You may use a built-in shuffler if YOU can figure out how.)

toString() (2h 3h, etc. all on one line that may wrap around if too long)

Deck.java implements the ADT Deck:
Many low level array operations are available within the ArrayList class library. The easiest way to implement our deck class is to make a private instance of ArrayList within the deck class and use the ArrayList functions to accomplish the deck functions. That is, for each deck function, the code inside will probably use the ArrayList functions on the private instance to accomplish the task required.

The client has no idea that there is an ArrayList "underneath" the deck and should not need to know. In fact, the ArrayList operations are NOT available to the client, only the Deck operations. This is the abstraction that helps to make a client programmer's job easier.

TestDeck.java

```
// this is the syntax for creating a new standard deck of 52 cards.
Deck  deck = new Deck( );
```

The TestDeck.java file must create an instance of a deck, and show the brand new deck on the screen. Then the deck should be shuffled, and displayed again. The screen should show both the new deck and the

shuffled deck.

Then show that one card may be removed from the deck. Repeat this function call and get the top 5 cards from the deck and show them on the screen. Then shuffle the deck again (the remainder of the deck), and show the deck on the screen, then deal the next 5 cards from the deck and show them.

3. **Hand.java** + **TestHand.java**

(For our specific game – note that hands for other games may be different)

ADT: The "**what**" needs to be accomplished
values: a collection of cards from a standard deck.

operations: (-- you may add more):
construct (empty hand)
add (add one card to the hand)
isEmpty
clear (remove all cards)
toString() (prepares a string of output showing card info)

"**How**" to accomplish in concrete code:
Hand.java

TestHand.java:
Create a deck of cards, shuffle them then show the shuffled deck on the screen. Create 1 empty hand:

Hand hand = new Hand(); // this is the client code for creating an empty hand.

"Deal" by getting a card off the top of the deck and adding it to the hand. Do this 10 times. Then show the hand on the screen. Make sure the cards in the hand are indeed the "top" ten cards off the deck. Now display the deck (rest of it). The top ten cards should have been removed and the remaining cards should be showing.

Test each of the functions for the hand and show that they work.