Name _____ Period _____

Card Dealer

Your Tasks (Mark these off as you go)

- ☐ Explain the purpose of a library as it applies to computer science
- ☐ Review the Card library
- ☐ Review the DeckOfCards library
- ☐ Write the dealCards function
- ☐ Write the suffleCards function
- ☐ Add more functionality to the CardDealer
- ☐ Receive credit for this lab guide

☐ Explain the purpose of a library as it applies to computer science

A **software library** contains already-developed procedures that you can use in creating your own programs. You don't have to write new procedures, for example, to display images or plot data in your program; someone's already written those procedures and put them in a library for the public to use.

What are some examples of libraries that we have already utilized in this class?

☐ Review the Card library

The Card library that you will interact with in this lab is small library that will enable you to create a playing card from a standard 52 card deck. Creating a Card requires two parameters: the suit of the card and its value. Once the card is created, there are additional functions that will allow you to interact with your cards.

Below illustrates how to use the Card library to creates the ace of spades,

```
var card1 = new Card("spades", 0);
```

To better understand how this work, let's peek into card library. It is not necessary to understand all the code in this library – and that is the beauty of a library – it is only important that you know how to interact with it.

Card.js

```
/**
 * Creates a card which represents a card from a standard 52
 * card deck
 * @author Pluska
 */
class Card {

    /**
     * creates a random card by creating a
     * random suit and a random value (1-13)
     */
    constructor(s, v){
        this.suitsArray = this.getSuits();
        this.faceValuesArray = this.getFaceValues();
        this.faceValue = this.faceValuesArray[v];//0 thru 12
        this.suit = this.suitsArray[s];//0 thru 3
        this.value = v;
    }

    /**
     * returns a list of suit names
     */
    getSuits(){
        return ["Spades","Diamonds","Clubs","Hearts"];
    }

    /**
     * returns a list of face values
     * each index in the list represents the
     * actual value of the card A = 0, J = 10, Q = 11, K = 13
     */
    getFaceValues(){
        return ["A","2","3","4","5","6","7","8","9","10","J","Q","K"];
    }

    /**
     * returns the suit of the card
     */
    getSuit(){
        return this.suit;
    }

    /**
     * returns the value of the card
     */
    getValue(){
        return this.value;
    }
```

```
    /**
     * returns the face value of the card
     */
    getFaceValue(){
        return this.faceValue;
    }

    /**
     * creates a button with displays the card
     * as the background
     */
    makeCardButton(){

        this.cardImage = "card" + this.suit + this.faceValue + ".png";
        this.cardButton = document.createElement("button");
        this.cardButton.style.position = "absolute";
        this.cardButton.style.backgroundImage = "url('images/" + this.cardImage
+ "')";
        this.cardButton.style.width = Card.width + "px";
        this.cardButton.style.height = Card.height + "px";
        return this.cardButton;
    }

}
Card.height = 71;
Card.width = 53;
```

Refer to the Card library above. Write code that could be used to create the following “hand” of cards,

Ace of spades, King of hearts, Queen of hearts, Jack of clubs, 7 of diamonds

Refer to the Card you created above, write code that could be used to print the value and suit of the Ace of Spades Card.

□ Review the Deck of Cards library

The Deck of Cards library, is another library that will enable us to build a 52 card deck from the Cards.

DeckOfCards.js

```
/**
 * Creates a deck of 52 cards
 * @author Pluska
 */
class DeckOfCards{

    constructor(){
      this.numSuits = 4;
      this.numValues = 13;
      this.cardCount = 0;
      this.cardArray = [];
      this.cardButtons = [];
      this.next = -1;

        for(var s = 0; s< this.numSuits;s++){
            for(var v = 0; v <this.numValues;v++ ){

              this.cardArray[this.cardCount] = new Card(s,v);
              this.cardCount++;
            }
       }
    }

    getCards(){
      return this.cardArray;
    }
}
```

Once again, it is not completely necessary to understand what is going in this library. Only need to know how to interact with it!

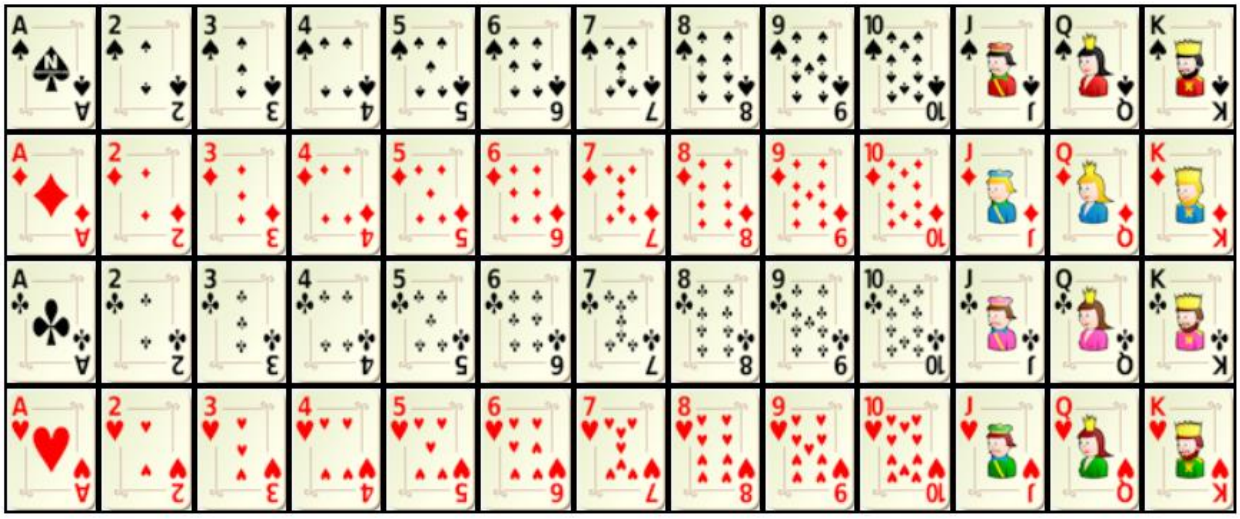
Write code that could be used to create a new Deck of Cards.
Assign the deck you created to a new array called deck.
Print out all the values in the deck of cards

□ Write the dealCards function

Our Card and DeckOfCard libraries are all we need to get started with dealing, shuffling, and manipulating our cards. The code you wrote above should look similar to that below.

```
//Creates a standard 52 deck of cards
var newDeck = new DeckOfCards();
//Gets the new deck of cards as an array
var cardArray = newDeck.getCards();
```

The code above creates a newDeck from the DeckOfCards library. We then retrieve all the cards in the newDeck and assign them to an array called cardArray. When we run our code we generate the following,



Each card in our cardArray was created from the Card library.

What is the index of the 6 of spades?
What card is at cardArray[21]
What card is at cardArray[cardArray.length – 1]
Write code that creates a new array called dealt. Then populate the dealt array with the first seven cards from the cardArray.

Write a function dealCards that accepts a parameter – d. In the body of dealCards, write code that could be used to populate the dealt array with the number of cards specified by d.

□ Write the shuffle cards function

The DeckOfCards library builds a sorted deck. So the dealt cards are always sorted too. Playing cards requires that we have a shuffled deck. We can shuffle our deck by swapping random cards in the deck over and over again. In the spaces below, you will write code that will allow you to create a shuffled deck of cards.

The signature for the swapCards function is as follows,

```
function swapCards(int i1, int i2)
```

i1 and i2 represent the indices of Cards created from the Card library. Write the swapCards method below which swaps two Cards in the cardArray.

The signature for the shuffleCards function is as follows,

```
function shuffleCards()
```

Write a loop to shuffle the cards. Inside this loop you should create two random numbers which represent the indices of two cards you want to swap. Then call swapCards. Your loop can be repeated as many times as you like to ensure the cards are adequately shuffled.

□ Add more functionality to the CardDealer

We can write additional functions in CardDealer.js to add functionality to our program. Because we will be calling

Consider the line of code below, which creates an array of 7 cards and assigns them to a Card array data type called dealt,

```
dealt = dealCards(7);
```

The card at a given index can be accessed as follows,

```
var firstCard = dealt[0];
```

Likewise, the value of the card expressed as a number (0 - 12) can be accessed by calling the getValue method in the Card library,

```
var cardVal = firstCard.getValue();
```

In the space below, write the getHighestCard function. The getHighestCard function should find the card with the highest value in dealt cards and assign it to highest. The getHighestCard function has the following signature,

```
function getHighestCard()
```

```
var highest;
```

In the space below, write the getLowestCard function. The getLowestCard method should find and return the card with the lowest value in a given array of dealt cards. The getLowestCard function has the following signature,

```
function getLowestCard()
```

```
var lowest;
```

Now that we know how to get the value of each Card, write the sumDeal function, which sums the values in the dealt array and prints it to the console. The sumDeal function has the following signature,

```
function sumDeal()
```

Write a function that could be used to count and return the number of each suit in a given array of dealt cards. The suit of a card can be accessed using the getSuit function in the Card library. Below, is an example of how to get the suit of the card at index 0 in the dealt array.

```
var suitOfCard = dealt[0].getSuite();
```

The number of each suite should be stored in the suite array,

```
var suitArray = [0, 0, 0, 0];
```

Where index 0 can represent clubs, index 1 can represent diamonds, etc.

The printSuits function has the following signature,

```
function getSuits()
```

Write a function that could be used to count and return the number of each card of given value in the dealt array. The value of a card can be accessed using the getValue method in the Card library. Below, is an example of how to get the value of the card at index 0 in the dealt array.

```
var valueOfCard = dealt[0].getValue();
```

The number of each value should be stored in the ValuesArray,

```
var valuesArray = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
```

Where index 1 can represent the number of cards with a value of 1, index 2 can represent the number of cards with a value of 2, etc.

The getValues function has the following signature,

```
function getValues()
```

☐ Receive credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.