Introduction

The best way to get familiar with objects and classes and the concept of encapsulation is to start with familiar, real-world objects. This assignment is step one in an exercise by which you will model a deck of cards.

Card Class

A class is a blueprint for similar objects. Certainly, a Card class can be used to make the **52 cards** of a **standard** playing deck (no jokers).

The primary questions when creating a class are:

- What are the attributes (data members) of a playing card?
- What are the behaviors (methods, function members) of a playing card?

Attributes

Every card has a suit and a value (for face cards just keep going: Jack == 11, Queen == 12, etc.). These are the *instance variables* for a Card object.

For the suit attribute use the enum class shown below (this is why enums exist in the first place).

```
public enum Suit {
      SPADE,  // ordinal value is 0
      HEART,  // ordinal value is 1, etc.
      DIAMOND,
      CLUB,
}
```

Note: An enum is like a class that has a fixed list of static values. Each value has a name.

The Card class will need a constructor. Remember the purpose of a constructor -- to create a valid instance of the class. This leads to these questions:

- What are the parameters to the constructor?
- What to do if the parameters are invalid? (Hint: throw an exception)

Your constructor should accept two parameters as shown below:

```
public Card(Suit s, int val)
```

Methods

We normally think of cards as inert objects that don't do anything. However, when we model anything in the computer we should provide some "computer" behavior. This computer behavior includes encapsulation, comparison, and printability.

Encapsulation

This means that you should **not** make the attributes public. You should define "getters" for all of the attributes. If you believe that code outside of the class should also be able to change an object's value then you should also define "setters" also. But think about it, once created, should you be able to change the nature of a card object?

You will need the following getters:

```
Suit getSuit()
int getValue()
```

Comparison

It would be wonderful if one card can compare itself to another card, just like we can compare one number to another number. It is a property of the real number system that when you compare any two real numbers there are only three possibilities: 1) the first number is strictly less than the second number 2) the first number is equal to the second number 3) the first number is strictly greater than the second number.

This property allows us to sort numbers and, eventually, we will be sorting cards. But there is a catch here -- you cannot use the operators <, ==, or > to compare cards (darn!) -- you have to implement a method to do this.

Let's call this method `compare()` and let's assume `card1` and `card2` are `Card` objects. We want our compare method to work like this:

```
result = card1.compare(card2);
```

What's happening? `card1` is calling the `Card` class's `compare()` method passing the `card2` object. What is the result? By convention the result is an integer (int) whose value reflects the three possibilities when you compare real numbers.

- result < 0 => card1 < card2
- result == 0 => card1 = card2
- result > 0 => card1 > card2.

Printability

At this point you will implement a print method for a card object. In the future we will learn about the `toString()` method and how to implement it. Here is the signature of the print method:

```
void print()
```

Here is some sample output that clearly indicates the face value and suit of each of four cards:

```
value: 2 suit: CLUB
value: 12 suit: HEART
value: 10 suit: SPADE
value: 2 suit: DIAMOND
```

Note: For face cards (Jack, Queen, King, Ace) just print the value.

Now a good thing about the `Suit enum` class is that each value has its own `toString()` method provided by the system. So to get a printable value of `suit` variable you can use: `suit.toString()`.

Assignment

Load the starter project Card.zip into Eclipse. Fill in the `Card` class with its data member and methods.

In the `main()` method, after you have filled in all of the methods of Card, uncomment the lines. The output should look something like this:

```
value: 2 suit: CLUB
value: 12 suit: HEART
value: 10 suit: SPADE
value: 2 suit: DIAMOND
value: 2 suit: CLUB < value: 12 suit: HEART
value: 2 suit: CLUB = value: 2 suit: DIAMOND
value: 12 suit: HEART > value: 10 suit: SPADE
```

Submit your Card class, `Card.java` to the Canvas assignment. Also, write a comment with your reflection on "what I learned" when doing this assignment.