PlayingCardDeck Class
PlayingCardDeck Inherits From Deck

Objective-C

PlayingCardDeck.h

```c
#import "Deck.h"
@interface PlayingCardDeck : Deck
@end
```

Let's look at one last class. This one is a subclass of Deck and represents a full 52-card deck of PlayingCards.

PlayingCardDeck.m

```c
#import "PlayingCardDeck.h"

@implementation PlayingCardDeck
@end
```
PlayingCardDeck Overrides init

PlayingCardDeck.h

#import "Deck.h"
@interface PlayingCardDeck : Deck
@end

@end

PlayingCardDeck.m

#import "PlayingCardDeck.h"

@implementation PlayingCardDeck
@end
#import "Deck.h"

@interface PlayingCardDeck : Deck
@end

@implementation PlayingCardDeck

-(instancetype)init
{
    // Initialization in Objective-C happens immediately after allocation.
    // We always nest a call to init around a call to alloc.
    // e.g. Deck *myDeck = [[PlayingCardDeck alloc] init]
    // or NSMutableArray *cards = [[NSMutableArray alloc] init]

    // Classes can have more complicated initializers than just plain “init”
    // (e.g. initWithCapacity: or some such).
    // We’ll talk more about that next week as well.

}
@end
PlayingCardDeck.h

#import "Deck.h"
@interface PlayingCardDeck : Deck
@end

PlayingCardDeck.m

#import "PlayingCardDeck.h"

@implementation PlayingCardDeck
-(instancetype)init {

Notice this weird “return type” of instanceType. It basically tells the compiler that this method returns an object which will be the same type as the object that this message was sent to.
We will pretty much only use it for init methods.
Don’t worry about it too much for now.
But always use this return type for your init methods.
}
@end
Only Time You Assign To self

Objective-C

PlayingCardDeck.h

#import "Deck.h"
@interface PlayingCardDeck : Deck
@end

PlayingCardDeck.m

#import "PlayingCardDeck.h"

@implementation PlayingCardDeck

-(instancetype)init
{
    self = [super init];
    if (self) {

    }

    return self;
}
@end

This sequence of code might also seem weird.
Especially an assignment to self!
This is the ONLY time you would ever assign something to self.
The idea here is to return nil if you cannot initialize this object.
But we have to check to see if our super class can initialize itself.
The assignment to self is a bit of protection against our trying to continue to initialize ourselves if our super class couldn’t initialize.
Just always do this and don’t worry about it too much.
PlayingCardDeck.h

#import "Deck.h"

@interface PlayingCardDeck : Deck
@end

PlayingCardDeck.m

#import "PlayingCardDeck.h"

@implementation PlayingCardDeck

- (instancetype)init
{
  self = [super init];
  if (self) {

  }

  return self;
}
@end

Sending a message to super is how we send a message to ourselves, but use our superclass’s implementation instead of our own. Standard object-oriented stuff.
Iterate Through Suits & Ranks

Objective-C

```objc
PlayingCardDeck.h

#import "Deck.h"
@interface PlayingCardDeck : Deck
@end

PlayingCardDeck.m

#import "PlayingCardDeck.h"

@implementation PlayingCardDeck
-(instancetype)init
{
    self = [super init];
    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
            }
        }
    }
    return self;
}
@end
```

The implementation of init is quite simple. We’ll just iterate through all the suits and then through all the ranks in that suit...
alloc & init A PlayingCard

Objective-C

PlayingCardDeck.h
#import "Deck.h"
@interface PlayingCardDeck : Deck
@end

PlayingCardDeck.m
#import "PlayingCardDeck.h"
@implementation PlayingCardDeck
-(instancetype)init
{
    self = [super init];
    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
                PlayingCard *card = [[PlayingCard alloc] init];
                card.rank = rank;
                card.suit = suit;
            }
        }
    }
    return self;
}
@end

Then we will allocate and initialize a PlayingCard and then set its suit and rank.

We never implemented an init method in PlayingCard, so it just inherits the one from NSObject. Even so, we must **always** call an init method after alloc.
Add The Card To The Deck

Objective-C

Finally we just add each PlayingCard we create to our self (we are a Deck, remember).

```c
@interface PlayingCardDeck : Deck
@end

@implementation PlayingCardDeck
-(instancetype)init
{
    self = [super init];
    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
                PlayingCard *card = [[PlayingCard alloc] init];
                card.rank = rank;
                card.suit = suit;
                [self addCard:card];
            }
        }
    }
    return self;
}
@end
```
That’s It For PlayingCardDeck

Objective-C

And that’s it!
We inherit everything else we need to be a Deck of cards
(like the ability to drawRandomCard) from our superclass.
Key References

Paul Hegarty

CS193P: iPhone Application Development.
Course Taught at Stanford University, Fall 2013.
Online version available on iTunes U