CS 696 Mobile Application Development
Fall Semester, 2010
Doc 5 Collection Classes
Sep 9, 2010

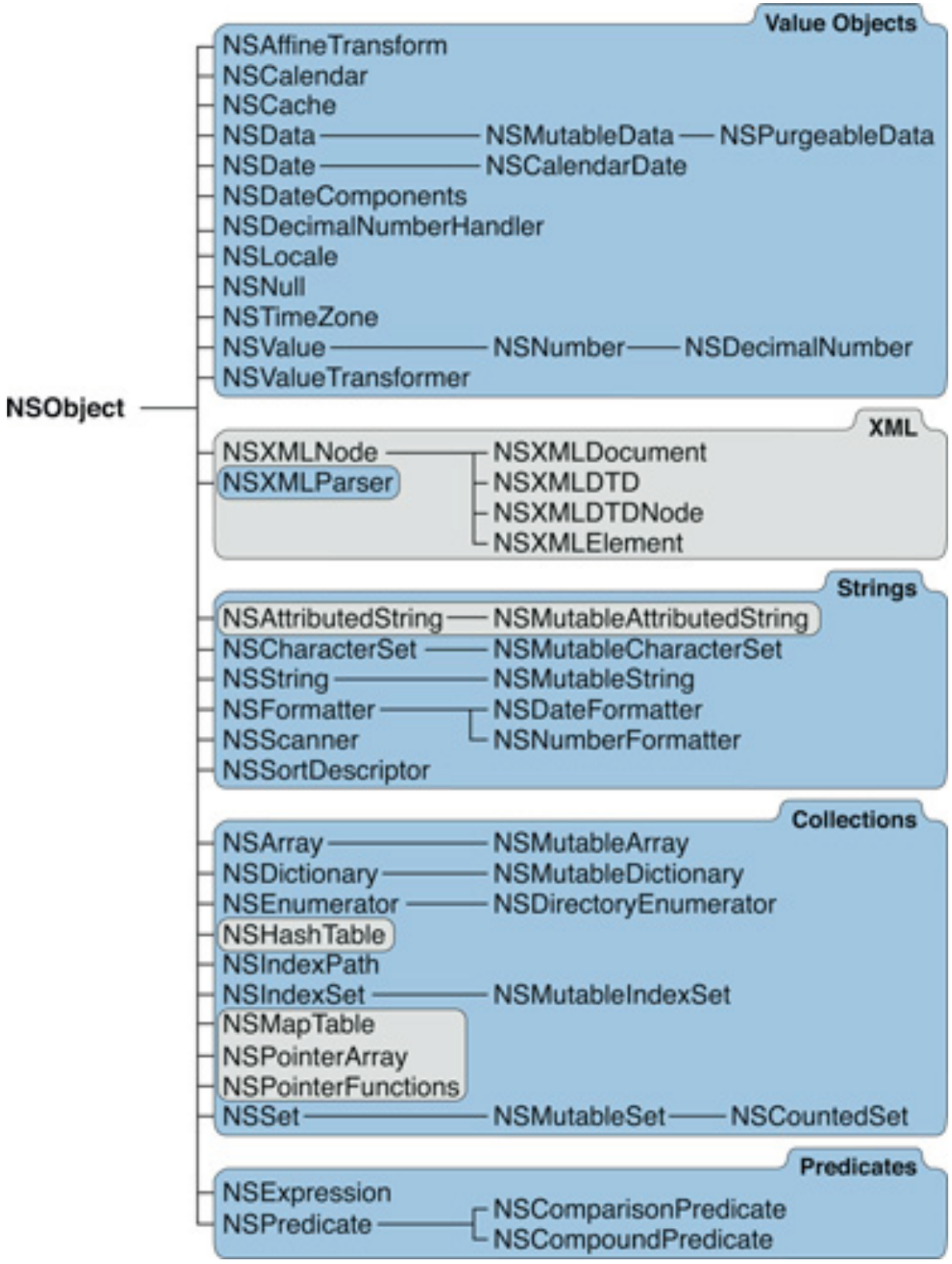
References

The Objective-C 2.0 Programming Language, http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html#//apple_ref/doc/uid/TP30001163

Collections Programming Topics for Core Foundation, http://developer.apple.com/library/mac/#documentation/CoreFoundation/Conceptual/CFCollections/CFCollections.html

Collections Programming Topics Data Management: Data Types & Collections, http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Collections/Collections.html%23//apple_ref/doc/uid/10000034-BBCFIHFH

Foundation Classes



Base Types and Objects

```
NSNumber * example = [NSNumber numberWithInt:10];
NSLog(@"int %i", [example intValue]);
NSLog(@"float %f", [example floatValue]);
example = [NSNumber numberWithFloat:5.632];
NSLog(@"int %i", [example intValue]);
NSLog(@"float %f", [example floatValue]);
```

Some NSNumber Methods

+ numberWithBool:
+ numberWithChar:
+ numberWithDouble:
+ numberWithFloat:
+ numberWithInt:
+ numberWithInteger:
+ numberWithLong:
+ numberWithLongLong:
+ numberWithShort:
+ numberWithUnsignedChar:
+ numberWithUnsignedInt:
+ numberWithUnsignedInteger:
+ numberWithUnsignedLong:
+ numberWithUnsignedLongLong:
+ numberWithUnsignedShort:

– boolValue
– charValue
– decimalValue
– doubleValue
– floatValue
– intValue
– integerValue
– longLongValue
– longValue
– shortValue
– unsignedCharValue
– unsignedIntegerValue
– unsignedIntValue
– unsignedLongLongValue
– unsignedLongValue
– unsignedShortValue

Mutable & Immutable Collection Classes

Immutable	Mutable
NSString	NSMutableString
NSArray	NSMutableArray
NSDictionary	NSMutableDictionary
NSIndexSet	NSMutableIndexSet
NSSet	NSMutableSet NSCountedSet

Mutable versions are subsets of immutable version

NSArray

```
NSArray *sample;
NSDate *aDate = [NSDate date];
NSValue *aValue = [[NSNumber alloc] initWithInt:5];
NSString *aString = @"a string";

sample = [NSArray arrayWithObjects: aDate, aValue, aString, nil];
[aValue release];
```

```
NSDate * result =[sample objectAtIndex: 0];
NSLog(@"date: %@", result);

NSInteger size = [sample count];
```

NSArray

Index starts at zero

Only holds objects

Collections

- retain objects when added

Enumerating

```
NSEnumerator *enumerator = [sample objectEnumerator];
id anObject;

while (anObject = [enumerator nextObject]) {
    NSLog(@"object:%@", anObject);
}

//Fast enumeration
for (id element in sample){
    NSLog(@"object:%@", element);
}

NSEnumerator *reverse = [sample reverseObjectEnumerator];
for (id element in reverse){
    NSLog(@"object:%@", element);
}
```

Note the pointer

```
sample = [NSArray arrayWithObjects:@"a", @"b", @"c", nil];

for (NSString * element in sample){
    NSLog(@"object:%@", element);
}
```

Yes you need the nil

```
[NSArray arrayWithObjects:@"a", @"b", @"c"]; // Runtime error
```

Enumerating with Blocks

```
NSString *findMe=@"c";
[anArray enumerateObjectsUsingBlock:^(id obj, NSUInteger index, BOOL *stop){
    if([obj localizedCaseInsensitiveCompare: findMe] == NSOrderedSame){
        NSLog(@"Object Found: %@ at index: %i",obj, index); *stop=YES;
    }
}];
```

Filtering & Predicates

```
NSArray * words = [NSArray arrayWithObjects: @"dog", @"cat", @"mat", nil];
NSPredicate *filterForA = [NSPredicate predicateWithFormat:@"SELF contains[c] 'a'"];
NSArray * containsA = [words filteredArrayUsingPredicate: filterForA];
//containsA == (cat, mat)

NSValue *small = [NSNumber numberWithInt:1];
NSValue *medium = [NSNumber numberWithInt:10];
NSValue *large = [NSNumber numberWithInt:100];
NSArray * numbers = [NSArray arrayWithObjects: large, small, medium, nil];

NSPredicate *range = [NSPredicate predicateWithFormat:@"SELF > 5 AND SELF < 20"];
NSArray * result = [numbers filteredArrayUsingPredicate: range];
// result == (10)
```


For more information see "Introduction to Predicates Programming Guide"

Person & Name class for examples

```
@interface Person : NSObject { }
@property (retain) Name* name;
@property int age;

+ (Person*) first: (NSString *) firstName last: (NSString *) lastName age: (int) age;
@end
```

```
@interface Name : NSObject { }

@property (copy) NSString* first;
@property (copy) NSString* last;

+ (Name*) first: (NSString*) first last: (NSString*) last;
```

Sorting with Selectors

```
Person* joe = [Person first: @"Joe" last: @"Spade" age: 10];
Person* sally = [Person first: @"Sally" last: @"Adam" age: 5];
Person* shah = [Person first: @"Shah" last: @"Chen" age: 8];
NSArray* people = [NSArray arrayWithObjects:joe,sally,shah,nil];

NSArray * sorted = [people sortedArrayUsingSelector:@selector(age)];
```

Sorting with Functions

```
Person* joe = [Person first: @"Joe" last: @"Spade" age: 10];
Person* sally = [Person first: @"Sally" last: @"Adam" age: 5];
Person* shah = [Person first: @"Shah" last: @"Chen" age: 8];
NSArray* people = [NSArray arrayWithObjects:joe,sally,shah,nil];

NSArray * sorted = [people sortedArrayUsingFunction:ageSort context:NULL];
```

```
NSInteger ageSort(id personA, id personB, void *context) {
        int ageA = [personA age];
        int ageB = [personB age];
        if (ageA < ageB)
                return NSOrderedAscending;
        else if (ageA > ageB)
                return NSOrderedDescending;
        else
                return NSOrderedSame;
}
```

Key-Value Coding

```
Person* joe = [Person first: @"Joe" last: @"Spade" age: 10];
Person* sally = [Person first: @"Sally" last: @"Adam" age: 5];
Person* shah = [Person first: @"Shah" last: @"Chen" age: 8];
NSArray* people = [NSArray arrayWithObjects:joe,sally,shah,nil];
```

```
NSNumber * sum = [list valueForKeyPath:@"@sum.age"];
NSNumber * max = [list valueForKeyPath:@"@max.age"];
NSArray * firstNames = [list valueForKeyPath:@"@unionOfObjects.name.first"];
```

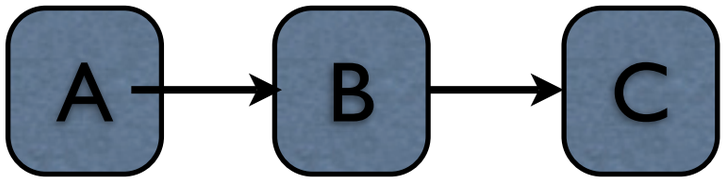
NSMutableArray

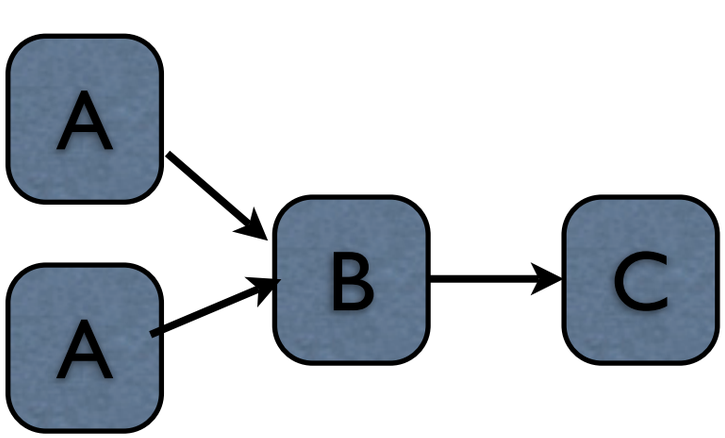
```
NSMutableArray * names = [NSMutableArray arrayWithCapacity:3];
[names addObject:@"Joe"];
[names addObject:@"Shah"];
[names addObject:@"Nguyen"];
NSLog(@"names:%@", names);

NSArray * one = [NSArray arrayWithObject:@"Joe"];
   NSArray * two = [one arrayByAddingObject:@"Shah"];
   NSArray * three = [two arrayByAddingObject:@"Nguyen"];
NSLog(@"three:%@", three);
```

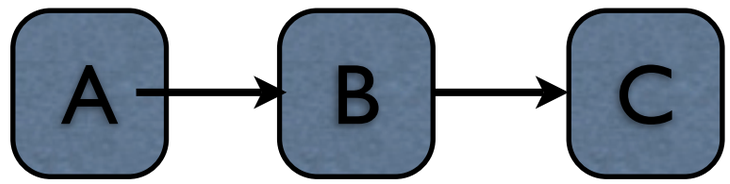
Copying

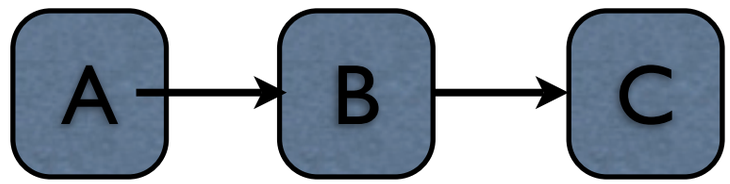
Shallow Copy

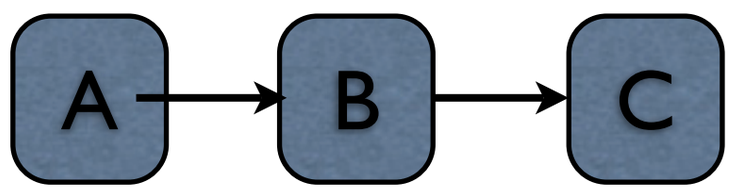




Deep Copy







Copying

Your class has to implement NSCopying protocol before copy works

NSCopying Protocol

- (id)copyWithZone:(NSZone *) zone

Class must define & implement copy

NSObject

- (id) copy

convenience method

Calls copyWithZone:

Copying Collections - Shallow Copy

```
NSArray * shallowCopy = [oldArray copy];

NSArray * shallowCopy = [oldArray copyWithZone: nil];

NSArray * shallowCopy = [NSArray alloc] initWithArray: oldArray];
```

Copying Collections - Deepish Copy

```
NSArray * deepishCopy = [NSArray alloc] initWithArray: oldArray copyItems: YES];
```

Objects in array must implement NSCopying protocol

Copies two levels deep

copyWithZone: supposedly normally copies only one level

Copying Collections - Deep Copy

```
NSArray* trueDeepCopyArray =
    [NSKeyedUnarchiver unarchiveObjectWithData:
        [NSKeyedArchiver archivedDataWithRootObject: oldArray]];
```

NSDictionary

Hash table

```
NSDictionary *courses = [NSDictionary dictionaryWithObjectsAndKeys:
        [NSNumber numberWithInt:63], @"Mathematics",
    [NSNumber numberWithInt:55], @"CS",
    [NSNumber numberWithInt:49], [NSNumber numberWithInt:10], nil];

  NSNumber * value = [courses valueForKey:@"CS"];
  value = [courses objectForKey:[NSNumber numberWithInt:10]];

  NSArray * allKeys = [courses allKeys];
  NSArray * allValues = [courses allValues];

  for (id key in courses) {
     NSLog(@"key: %@, value: %@", key, [courses objectForKey:key]);
```

Dictionary Keys

Keys must implement

- NSCopying protocol
- hash, isEqual (both defined in NSObject)

Dictionary Basics

Keys are copied when added to dictionary

Objects are retained when added

Objects are released when removed

Removing from mutable collections

```
NSMutableDictionary *example = [NSMutableDictionary
            dictionaryWithObjectsAndKeys: @"42",@"Answer",nil] ;
NSString * value = [example objectForKey:@"Answer"];
[example removeObjectForKey: @"Answer"];
//value now released!
```

```
NSString * value = [[example objectForKey:@"Answer"] retain];
[example removeObjectForKey: @"Answer"];
```

Creating & saving NSDictioanaries

```
NSArray * keys = [NSArray arrayWithObjects:@"English",@"CS",@"History",nil];
NSArray * values = [NSArray arrayWithObjects:[NSNumber numberWithInt:72],
                    [NSNumber numberWithInt:55],[NSNumber numberWithInt:49],nil];

NSDictionary *courses = [NSDictionary
                                dictionaryWithObjects: values forKeys: keys] ;

 BOOL didWrite = [courses writeToFile:@"/users/whitney/courses.plist"
                                atomically:YES];

NSDictionary * propertyList = [NSDictionary dictionaryWithContentsOfFile:
                        @"/users/whitney/courses.plist"];
```


All items in the dictionary must be instances of NSData, NSDate, NSNumber, NSString, NSArray, or NSDictionary

/users/whitney/courses.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CS</key>
        <integer>55</integer>
        <key>English</key>
        <integer>72</integer>
        <key>History</key>
        <integer>49</integer>
</dict>
</plist>
```

Sorting Dictionaries with Selectors

```
NSDictionary *courses =  [NSDictionary dictionaryWithObjectsAndKeys:
                    [NSNumber numberWithInt:63], @"Mathematics",
                    [NSNumber numberWithInt:55], @"CS",
                    [NSNumber numberWithInt:49], @"History", nil];
NSArray *sortedKeys =
     [courses keysSortedByValueUsingSelector: @selector(compare:)];
```

```
sortedKeys == ( @"History", @"CS", @"Mathematics")
```

Sorting Dictionaries with Comparators

```
NSDictionary *courses =  [NSDictionary dictionaryWithObjectsAndKeys:
                    [NSNumber numberWithInt:63], @"Mathematics",
                    [NSNumber numberWithInt:55], @"CS",
                    [NSNumber numberWithInt:49], @"History", nil];

NSArray * sortedKeysArray =
        [courses keysSortedByValueUsingComparator: ^(id obj1, id obj2) {
    if ([obj1 integerValue] > [obj2 integerValue]) {
        return (NSComparisonResult) NSOrderedDescending;
    }
    if ([obj1 integerValue] < [obj2 integerValue]) {
        return (NSComparisonResult) NSOrderedAscending;
    }
    return (NSComparisonResult) NSOrderedSame;
}];
```

NSSet

```
NSSet *variables=[NSSet setWithObjects: @"X", @"Y", @"Z", nil];

for (id k in variables) {
    NSLog(@"k:%@", k);
}
[variables containsObject:@"X"];
```

NSCountSet

```
NSSet *variables=[NSCountedSet setWithObjects: @"X", @"Y", @"X", nil];

NSUInteger count = [variables countForObject:@"X"];
```

NSIndexSet

Set of indexes to other collections

```
NSArray * words = [NSArray arrayWithObjects:@"cat",@"rock",@"rat",nil];
NSIndexSet *aWordsIndex=[words indexesOfObjectsPassingTest:
        ^(id aString, NSUInteger index, BOOL *stop){
        char second = [aString characterAtIndex:1];
        if ('a' == second){
          return YES;
        }
        return NO;
   }];

NSUInteger index = [aWordsIndex firstIndex];
while (index != NSNotFound) {
    NSLog(@"word: %@", [words objectAtIndex:index]);
    index = [aWordsIndex indexGreaterThanIndex: index];
}
```