



Computing & Information Sciences
FLORIDA INTERNATIONAL UNIVERSITY

Mobile Application Development

lecture5

Spring 2012- COP 4655 U1

M/W 6:25pm – ECS 138

Steve Luis

Agenda

- Accessors
- Coding Review
- Programming assignment #2



Accessors

- Method that gets or sets the value of an instance variable: called “getter” and “setter”

```
@interface Fraction: NSObject
```

```
{
```

```
int numerator;
```

```
int denominator;
```

```
}
```

```
- (void) setNumerator: (int) n;
```

```
- (void) setDenominator: (int) d;
```

```
- (int) numerator;
```

```
- (int) denominator;
```

```
@end
```

@implementation Fraction

```
- (void) setNumerator: (int) n {  
    numerator = n;  
}  
- (void) setDenominator: (int) d {  
    denominator = d;  
}  
- (int) numerator {  
    Return numerator;  
}  
- (int) denominator {  
    return denominator;  
}  
@end
```

```
Fraction *myFraction = [[Fraction alloc] init];
```

```
[myFraction setNumerator: 1];
```

```
[myFraction setDenominator: 2];
```

```
NSLog(@"Numerator = %i and Denominator = %i",  
[myFraction numerator], [myFraction  
denominator]);
```

Output: Numerator = 1 and Denominator = 2

Synthesized Accessor Methods

- a getter or setter method created by the Objective C compiler.
- Use `@property` in the interface and `@synthesize` in the implementation
- You can only use dot notation to reference instance variables that implement synthesized accessor methods.

```
@interface Fraction: NSObject
{
int numerator;
Int denominator;
}
@property int numerator, denominator;
@end
```



```
@implementation Fraction
@synthesize numerator, denominator;
@end
```

```
=====
Fraction *myFraction = [[Fraction alloc] init];
```

```
[myFraction setNumerator: 1];
[myFraction setDenominator: 2];
```



```
NSLog(@"Numerator = %i and Denominator = %i",
[myFraction numerator], [myFraction denominator]);
```



Output: Numerator = 1 and Denominator = 2

```
@implementation Fraction
@synthesize numerator, denominator;
@end
```

=====

```
Fraction *myFraction = [[Fraction alloc] init];
```

```
myFraction.numerator = 1;
myFraction.denominator = 2;
```



```
NSLog(@"Numerator = %i and Denominator = %i",
myFraction.numerator, myFraction.denominator);
```



Output: Numerator = 1 and Denominator = 2

Dot notation

- `instance.property` to get a value
- `instance.property = value` to set.

Objective-C Review Points

- A header file (.h) where you declare instance variables, properties, and methods. The implementation (.m) file where you write your methods.
- Dynamically typed, you don't call methods, you send messages. This means that the Objective-C runtime does not care what type your object is, only whether it will respond to the messages you send it.
- Properties in Objective-C are "synthesized" with the @synthesize keyword to create the getter and setter methods.
- Objective-C does not support name spaces. This is why you'll see Objective-C classes with two (or more) letter prefixes
- Objective-C doesn't provide garbage collection. Apple uses a techniques called reference counting and automatic reference counting.

Assignment

- Program #2: Fraction Calculator Kochan Chapter 21
- Build the Fraction Calculator using a View Based Application Template.
- Add the convert button noted on exercise #1 on page 479.
- Please comment the code that you add.
- Program #2 is due Weds. Feb 1st at 11pm.
- Participation 2 assignment due Weds. Feb 1st at 11pm.

Programming Assignments

Dos and Don'ts

- DO:
 - Discuss how to use Xcode
 - Objective C syntax
 - Framework Libraries
 - Design patterns and coding best practices
 - Walkthru of sample programs
- DON'T:
 - Give a copy of your code to another student (unless you are working on a team assignment)
 - Tell/show a student “This is how I did it”