Praktikum Entwicklung von Mediensystemen mit iOS

Wintersemester 2012 / 2013

Prof. Heinrich Hußmann, Dr. Alexander De Luca, Fabius Steinberger
Today

• Organization
• Introduction to iOS programming
• Assignment 1
Organization

- 6 ECTS
- Bachelor: Vertiefendes Thema
- Master: Gruppenpraktikum
- Wednesday 16 - 18, Amalienstr. 17 A107
- http://www.medien.ifi.lmu.de/lehre/ws1213/pem/
Roadmap

- **October, November:** weekly lectures and individual assignments
- **December, January:** app development in teams, milestone meetings and presentations
- **February:** final presentation (probably 6.2.2013)
iOS

- Mobile operating system by Apple for iPhone, iPad and Apple TV
- Based on Unix, derived from OS X
- 400 million iOS devices sold, 25% share of smartphone market, 65% share of mobile web consumption
- Latest release: iOS 6.0 (September 2012)
Layers of iOS

Cocoa Touch
Multi-touch, Web View, Map Kit, Camera, Image Picker...

Media
Core Audio, PDF, Core Animation, Quartz 2D, OpenGL...

Core Services
Core Location, Preferences, Address Book, Preferences...

Core OS
File System, Kernel, Power Management, Security...
User input

- GUI controls: buttons, sliders, switches
- Multi-touch gestures: tap, pinch, rotate, swipe, pan
- Accelerometer: shaking, rotating
Development Environment


XCode
XCode

- **Source editor:** code completion, syntax highlighting, context-sensitive information

- **Interface builder:** UI elements library and inspector, split editor to connect UI with code, Storyboards

- **Compiler:** C, C++, Objective-C

- **iOS Simulator:** run and test apps on a Mac

- **More:** refactoring, version control, debugging, analysis
  
  (https://developer.apple.com/technologies/tools/)
XCode

Compile and run
Device and simulator selection
Source editor
Show/hide sidebars

File navigator
Utilities sidebar (API info)
Contents of an XCode project

- Source code files (.h and .m)
- Interface files (.storyboard and .xib)
- Libraries (.framework)
- Resources, e.g. images (.png)
- App configuration file (Info.plist)
Objective-C

- Language for programming iOS and Mac apps, also used by Apple to create much of OS X, iOS, APIs
- Superset of C
- Object-orientated


Elements of Objective-C

<table>
<thead>
<tr>
<th>Java</th>
<th>Objective-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyClass.java</td>
<td>Header.h</td>
</tr>
<tr>
<td></td>
<td>Implementation.m</td>
</tr>
<tr>
<td>Methods and method calls</td>
<td>Methods and messages</td>
</tr>
<tr>
<td>Attributes, setters, getters</td>
<td>Properties, instance variables</td>
</tr>
<tr>
<td>Constructor</td>
<td>Initializer</td>
</tr>
<tr>
<td>Interface</td>
<td>Protocol</td>
</tr>
<tr>
<td>Garbage Collection</td>
<td>Automatic Memory Management (ARC)</td>
</tr>
</tbody>
</table>
Methods

• Definition (in .h):

- (void) doSomething;
- (void) doSomethingWithA: (NSString *) a andB: (NSString *) b;

• Implementation (in .m):

- (void) doSomething {
  // do something
}
- (void) doSomethingWithA: (NSString *) a andB: (NSString *) b {
  // do something with a and b
}

• Method call (“message”) (in .m):

[<self> doSomething];
NSString* a = @"a";
NSString* b = @"b";
[<self> doSomethingWithA:a andB:b];
Instance Variables ("ivars")

- Like private attributes in Java
- Definition (in .h): `NSString* _name;`
- Use (in .m):
  ```
  _name = @"Max";
  labelText = _name;
  ```
- You don’t have to use the underscore ( _ ), but it’s good practice. Otherwise you accidentally mix up ivars and properties (see next slide).
Properties

- Auto-creation of an instance variable (private) as well as a getter and setter (public)

- Definition (in .h):
  ```
  @property(strong, nonatomic) NSString *name;
  ```

- Using getters/setters (in .m):
  ```
  [self setName:@"Max"];  
  self.name = @"Max";

  NSString *labelText = self.name; 
  labelText = [self name];
  ```

- Using the instance variable (in .m):
  ```
  _name = @"Max";
  labelText = _name;
  ```

- **strong/weak**: refers to ownership. Always use strong except for properties that point to a parent.

- **nonatomic/atomic**: use nonatomic to avoid multi-threading issues.

- **self.name**: this syntax does NOT access the variable itself. It's a getter/setter, just like the other syntax.

- **_name**: Use this instance variable in custom setters/getters and in init-methods only. In any other case, use the getter/setter.
Object Initialization

- **Object:** MyClass *myObject = [[MyClass alloc] init];

- **Object with parameter:** MyClass *myObject = [[MyClass alloc] initWithParameter: parameter];

- **String:** NSString *hello = @"Hello";

  NSString *helloWorld = [NSString stringWithFormat:@"%@ World", hello];

- **Array:** NSArray *colors = @["Green", "Red", "Yellow"];

  NSMutableArray *mutableColors = [NSArray arrayWithArray: @["Green", "Red", "Yellow"] mutableCopy];

- **THERE ARE NO NULL POINTER EXCEPTIONS.** So if your app compiles but doesn’t work properly, make sure your objects aren’t nil.
Hello World

• New XCode Project: Single View Application

• In the storyboard, drag a text label and a switch onto the screen
Hello World

• Open the assistant editor and ctrl-drag the text label into ViewController.h. Enter a name and click Connect. You now have access to the UI element in your code. Do the same for the switch.

• Again, ctrl-drag the switch into the code. This time, select Action instead of Outlet. Enter a name and click Connect. You now have a listener method that is called by the OS when the user changes the value of our switch.
Hello World

• Close the assistant editor and go to ViewController.m. Complete the IBAction method:

```swift
- (IBAction)switchChangedValue:(id)sender {
    if (self.mySwitch.on) {
        self.myTextLabel.text = @"Hello World";
        NSLog(@"Hello World");
    } else {
        self.myTextLabel.text = @"";
    }
}
```

• Open the debug area and run the code.
UIViewController

- One of the most important classes in iOS programming
- You have to subclass UIViewController when creating a new screen
- Provides methods for managing the view hierarchy throughout its life cycle and for reacting to events (also great for debugging), e.g.
  - viewDidLoad:
  - viewWillAppear:
  - viewDidLoad:
  - viewWillAppear:
  - (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
duration:(NSTimeInterval)duration;

  UIViewController_Class/Reference/Reference.html)
App Delegate

• Every app must have an App Delegate.

• Provides methods for managing the app throughout its life cycle (also great for debugging), e.g.

  - application:didFinishLaunchingWithOptions:
  - applicationDidBecomeActive:
  - applicationDidEnterBackground:
  - applicationWillEnterForeground:
  - applicationWillTerminate:


• There are lots of protocols (often named Delegate), e.g. for managing the keyboard, table views, date pickers.
Top 3 Resources

1. iOS Developer Library
   or https://developer.apple.com/library/ios

2. RAYWENDERLICH
   Tutorials for iPhone / iOS Developers and Gamers
   http://www.raywenderlich.com/tutorials

3. stackoverflow
   https://stackoverflow.com
   stackoverflow ios app delegate
Assignment 1

• Individual assignment
• Get to know XCode and Objective-C
• Due next Wednesday 12:00, upload to Uniworx

• Questions?