OBJECTIVE C

James Gahn
G.J. Zynda
Robert Williamson

INSTALLATION

- For what we will be doing in class, the addiator server will serve us well

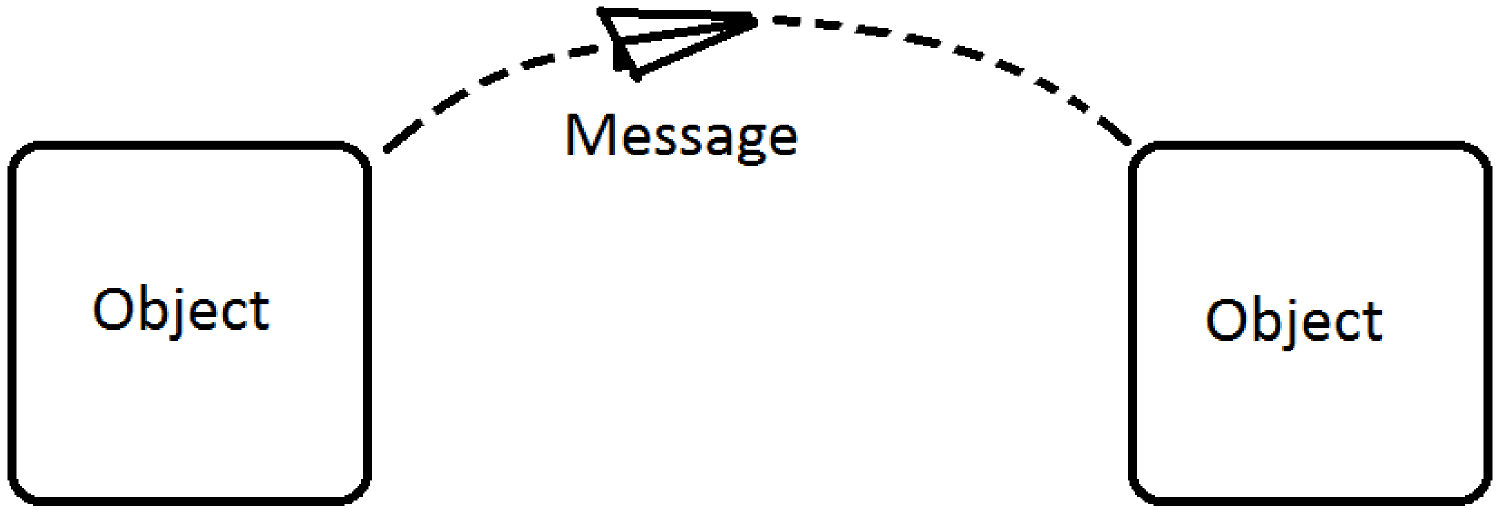
HISTORY

- Objective C is a superset of the C language
  - Creators wanted to add object-oriented programming to what was the trusty C language of the era
  - Recently, the majority of Objective C programs utilize Apple’s Cocoa API for applications on Mac OS and iPhone

Quiz #1

OBJECT INTERACTION MODEL

- Objects pass messages to one another
- Based on the style of Smalltalk



Quiz #2, 3

MESSAGES

- In most languages, calling an object's methods appears as:

```
object.method(arguments)
```

- In Objective C, methods are messages that we pass to the object
- Any message can be passed, though only the ones defined will make the object do anything

```
[object message]
[object message:arg1 message:arg2 …]
```

MESSAGE PASSING

- There are built-in functions for objects that handle messages we aren’t expecting
- We can use these functions to forward these errant messages to the proper receivers

(We aren’t gonna show you this, just believe us)

OBJECTS

- Objects in Objective C are always referenced by pointers
  - Objects are not simple types and hold much more than several bytes of data
- A built-in static method named ‘alloc’ is used like ‘malloc’ to easily reserve space for the object
  - Similarly, the message ‘free’ can be passed to release the reserved memory

HELLO WORLD…

- …will not be shown, because Objective C and C are so similar, the standard "hello world" programs look almost identical
  - The main difference is the .m extension in place of .c and #import in place of #include

CREATING CLASSES

- Classes require two separate blocks of code for declaration
  - @interface
    - Like Java's abstract classes, they allow us to declare every field and every method (static or instance)
  - @implementation
    - Like actually defining the class

Quiz #4

THE ID TYPE

- The type `id` is an object reference which can refer to any specific Class type
  - Think of the Object class in Java
    - Any object can be passed to a method expecting an Object, but the variable it is assigned to has very limited functionality
- `id` can be used to pass around objects whose type we don't care about

A SIMPLE OBJECT

- test.m

```
#import <objc/Object.h>
#import <stdio.h>
@interface test : Object {
    char* shout;
}
-(void)shout;
@end
```

Continued on next slide…

A SIMPLE OBJECT (CONTINUED…)

```
@implementation test
-(id)init{
    self = [super init];
    shout = “Hello, World!”;
    return self;
}
-(void)shout{
    printf(“%s\n”,shout);
}
@end
```

USING OUR OBJECT

```
int main(){
    test* t; //remember it's a pointer
    t = [test alloc]; //reserve memory
    t = [t init]; //initialize
    [t shout]; //send it a message
    return 0;
}
```

The initialization lines are usually condensed to:

```
t = [[test alloc] init];
```

COMPILING AND RUNNING

- On either Addiator or Cygwin:

```
gcc -o test test.m -lobjc
```

- Which will build, and to run:

```
./test
```

- Which should output:

```
Hello, World!
```

Quiz #5

CREATING A TREE

- Hands on example

Strength and Weaknesses

- Strengths
  - Very east to develop in on Mac OS
  - Very east to learn if you have experience with C
  - iPhone!
- Weaknesses
  - Hard to develop in on Windows
  - GCC Objective-C is always behind Apple’s Objective-C
- Good Projects
  - iPhone app
- Bad Projects
  - Any project that doesn’t require objects