Software Quality
First, An Aside
Memory allocation

• Static
• Dynamic
C++

You provide destructors for your classes.

delete grid;

Objective-C:

? 

Let’s do a brief review of memory allocation
Static Memory Allocation
(space determined at compile time)

Space allocated when doSomething is called.
Released when doSomething returns.

```c
void doSomething
{
    int myArray[81];
    ...
}
```
Static Memory Allocation
(space determined at compile time)

Space allocated when doSomething is called.
Released when doSomething returns.

```c
int* doSomething
{
    int myArray[81];
    ...
    return myArray;
}
```

BIG PROBLEM
Dangling Pointer

STACK
Dynamic Memory Allocation (run time)

```c
void doSomething((int) arraySize)
{
    int* myArray = new int[arraySize];
    ...
}
```

new says:
Do NOT put this onto the STACK
Put it on the HEAP
Where is HEAP?
How is HEAP managed?

Heap Space for myArray
Used space

9/11/13
Dynamic Memory Allocation (run time)

(int*) doSomething((int) arraySize)
{
    int* myArray = new int[arraySize];
    ...

    return myArray;
}

myArray may be used outside of doSomething function. myArray created by the function doSomething, but exists outside of doSomething’s environment. -- you must tell system when it is ok to release it!
C++: Releasing dynamically allocated memory:
delete[] myArray;
C++: Releasing dynamically allocated memory:
delete[] myArray;
delete grid; You provide class destructor method.
C++: Releasing dynamically allocated memory:
delete[] myArray;
delete grid; You provide class destructor method.

Objective-C: Releasing dynamically allocated memory:
??
C++: Releasing dynamically allocated memory:

```
delete[] myArray;
delete grid; You provide class destructor method.
```

Objective-C: Managing dynamically allocated memory:

```
In iOS .. All we Care About
```
NSObject is the root class of iOS class hierarchies.
iOS: reference count

When retainCount=0 memory is available for released
MyClass* A = [ [MyClass alloc] init];

If MyClass is derived from NSObject then
NSObject’s init initializes retainCount

Provided you call the init of the superclass NSObject

Tutorial 2:
- (id) initWithFrame:(CGRect)frame
  {
    self = [super initWithFrame:frame];
    if (self) {
      // Initialization code
      [self setBackgroundColor:[UIColor blackColor]];
    }
    return self;
  }

Where is retainCount?
MyClass* A = [ [MyClass alloc] init];

[ A release ];

Release causes retain count to decrement

retainCount = 1
retainCount = 0
MyClass* A = [ [MyClass alloc] init];
MyClass* B = A;

Does not cause retain count to increment.
MyClass* A = [ [MyClass alloc] init]
MyClass* B = A
[A release]

PROBLEM
May work until memory re-used
Why??

Zombie: pointer to de-allocated memory
MyClass* A = [ [MyClass alloc] init];
MyClass* B = A;
[B retain];

Causes retain count to increment

retainCount = 1
retainCount = 2
GOOD NEWS:

Automatic Reference Counting (ARC) inserts retains and releases for you.
Is it magic?

No – ARC is fallible
GOOD NEWS:
Automatic Reference Counting (ARC) inserts retains and releases for you.

BAD NEWS:
Doesn’t always work

GOOD NEWS:
Xcode tools help track down memory problems.
Software Quality

We’ll talk a lot about software quality and how to ensure it as well as assess it.
What do we mean by software quality?

- Functionality
- Usability
- Performance
- Reliability
- Supportability
Supportability

• What is your coding policy?
• Names, Variables, Static Code, etc.
• Exercise: without sharing info, each of you write down your Coding Policy for:
  • Classes, Methods, Variables, etc.

• How do yours compare to published standards? (e.g. PEP8 or GNU Standards or ?)
How can we assess the quality of software?

Users/Customers' perspective

Developers' perspective
How can we assess the quality of software?

Users/Customers' perspective

Does it do what it is supposed to?
Does it do anything it shouldn’t?
How should we improve it?

Developers' perspective

Is the code correct?
Is the code clear, readable?
Is the code extensible?
Sudoku.v1: User/customer perspective

• Does it do what it is supposed to do?
• Does it do anything it shouldn’t do?
• How can we improve it?
Does it do what it is supposed to do?

• Feature: Display initial grid

• User story: As a player I want to view the initial Sudoku grid in order to figure out my first move. The initial grid should have blanks and should have a unique solution.

• Feature: cells can be tapped
Sudoku.v1

• Does it do what it is supposed to do?
  – Does it display a grid?
  – Is it a valid grid with blanks?
  – Can we tap cells?

Let’s critique v1!
Sudoku.v1

• Does it do what it is supposed to do?
  – Does it display a grid?
  – Is it a valid grid with blanks?
  – Can we tap cells?
• Does it do anything it shouldn’t?

Let’s try to break v1!
Sudoku.v1

• Does it do what it is supposed to do?
  – Does it display a grid?
  – Is it a valid grid with blanks?
  – Can we tap cells?

• Does it do anything it shouldn’t do?

• What ways could it be improved?
Improvements

• Aesthetics (not very pretty)
• More interesting grids to solve

Add user stories as appropriate to backlog.
How can we assess the quality of software?

**Users/Customers’ perspective**

Does it do what it is supposed to?  
Does it do anything it shouldn’t?  
How should we improve it?

**Developers’ perspective**

Is the code correct?  
Is the code clear, readable?  
Is the code extensible?
Sudoku.v1: Developer’s perspective

• Is the code correct?
• Is the code clear, readable?
• Is the code extensible?

Tests
Code reviews
Sudoku.v1: Developer’s perspective

- Is the code clear and readable?
- Is the code extensible?
- Is the code correct?

Tests
Code reviews
Code Review

Other professionals look at and critique your code

• Formal

• Informal
  – Pair programming
  – Over the shoulder review
  – Walkthrough
  – Etc.

Developer walks through code
Reviewers ask question, critique, evaluate
Produce recommended changes
Sudoku v1 code walkthrough

• Correct
• Clear and readable
  – Names
  – Formatting
  – Comments
  – Method length, parameters, level of abstraction
• Extensible

Let’s do a walkthrough of v1!

http://www.codinghorror.com/blog/2006/05/the-ten-commandments-of-egoless-programming.html
Coding standards

• You will decide as a group
• Hwk and lab this week
• This is the Apple Bible place:

Lab

1\textsuperscript{st} Part:

Working with your group, create and use a design rubric

2\textsuperscript{nd} Part:

Further your design and implementation Sudoku
Logistics

Sudoku Project: Make sure you can demo on your own machine or Portable machine.
To install on iPad you need to develop on a Portable machines.

Game Project: Same Rules
Summary

• Code quality – Sudoku.v1
  – User/customer’s perspective
    • Does it do what it should?
    • Does it do anything it shouldn’t?
    • How can it be improved?
  – Developer’s perspective
    • Is the code clear, readable?
    • Is the code extensible?
    • Is the code correct?

Demo, QA test
Instrumentation tests,
Code walkthrough

9/11/13
The End
But first ... a recap of what you learned
(and a few things you didn’t)
about Objective-C
C++ vs. Objective-C

• Methods
Syntax: method calls

<table>
<thead>
<tr>
<th>C++ method calls</th>
<th>Objective-C message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td>myObject-&gt;doSomething()</td>
<td>[myObject doSomething]</td>
</tr>
</tbody>
</table>
## Syntax: method calls

<table>
<thead>
<tr>
<th>C++ method calls</th>
<th>Objective-C message</th>
</tr>
</thead>
<tbody>
<tr>
<td>myObject-&gt;doSomething()</td>
<td>[myObject doSomething]</td>
</tr>
<tr>
<td>myObject-&gt;doSomething(x)</td>
<td>[myObject doSomething:x]</td>
</tr>
</tbody>
</table>
## Syntax: method calls

<table>
<thead>
<tr>
<th>C++ method calls</th>
<th>Objective-C message</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>myObject-&gt;doSomething()</code></td>
<td><code>[myObject doSomething]</code></td>
</tr>
<tr>
<td><code>myObject-&gt;doSomething(x)</code></td>
<td><code>[myObject doSomething:x]</code></td>
</tr>
<tr>
<td><code>myObject-&gt;doSomething(x,y)</code></td>
<td><code>[myObject doSomething:x :y]</code></td>
</tr>
</tbody>
</table>
Method signatures

C++
void createCell( (int) row, (int) column);

Objective-C
-(void) createCell: (int) row : (int) column

Each argument preceded by a colon
Method signatures

C++
void createCell( (int) row, (int) column);

Objective-C
-(void) createCell: (int) row  : (int) column;
-(void) createCellAtRow: (int) row andColumn: (int) column;

Perfectly legal
Vastly superior

Why?

Description of argument
Objective-C
-(void) createCellAtRow: (int) row andColumn: (int) column;

The name of a method is called a selector:
createCellAtRow: andColumn:

A message contains the object pointer, its selector, and the arguments passed
[grid createCellAtRow: row andColumn: column]

Tutorial2:
[theButton addTarget:self action:@selector(buttonPressed:) forControlEvents:UIControlEventTouchUpInside];
C++ vs. Objective-C

- Method
- Accessor Methods
Objective-C Accessor methods

@interface Grid: UIView
{
    int numberCells;
}
-(int) getNumberCells;
-(void) setNumberCells: (int) newNumberCells;
@end
Objective-C Accessor methods

@interface Grid : UIView
{
    int numberCells;
}
-(int) getNumberCells;
-(void) setNumberCells: (int) newNumberCells;
@end

@implementation Grid
-(int) getNumberCells
{
    return numberCells;
}
-(void) setNumberCells: (int) newNumberCells
{
    numberCells=newNumberCells;
}
@end

9/11/13
Objective-C Accessor methods

@interface Grid: UIView
{
    int numberCells;
}
-(int) getNumberCells;
-(void) setNumberCells: (int) newNumberCells;
@end

@implementation Grid
-(int) getNumberCells
{
    return numberCells;
}
-(void) setNumberCells: (int) newNumberCells
{
    numberCells=newNumberCells;
}
@end

@interface Grid: UIView
@property int numberCells;
@end

@implementation Grid
@synthesize numberCells;
@end

Xcode 4.4 and later synthesizes automatically

@implementation
@synthesize numberCells;
@end

9/11/13
Objective-C Accessor methods

@interface Grid : UIView
{
    int numberCells;
}
-(int) getNumberCells;
-(void) setNumberCells: (int) newNumberCells;
@end

@implementation Grid
-(int) getNumberCells
{
    return numberCells;
}
-(void) setNumberCells: (int) newNumberCells
{
    numberCells=newNumberCells;
}
@end

@interface Grid : UIView
@property int numberCells;
@end

@implementation
@synthesize numberCells;
@end

Tutorial 2:
theButton.backgroundColor = [UIColor whiteColor]
Objective-C Accessor methods

@interface Grid : UIView
{
    int numberCells;
}
-(int) getNumberCells;
-(void) setNumberCells: (int) newNumberCells;
@end

@implementation Grid
-(int) getNumberCells
{
    return numberCells;
}
-(void) setNumberCells: (int) newNumberCells
{
    numberCells = newNumberCells;
}
@end

@interface Grid : UIView
@property int numberCells;
@end

@implementation
@synthesize numberCells;
@end

PUBLIC data member?

Tutorial 2:
theButton.backgroundColor = [UIColor whiteColor]
Objective-C Accessor methods

@interface Grid :
    UIView
{
    int numberCells;
}
@end

@interface Grid :
    UIView
@property int numberCells;
@end

@implementation Grid
-(int) getNumberCells
{
    return numberCells;
}
@end

@implementation Grid
-(void) setNumberCells: (int) newNumberCells
{
    numberCells = newNumberCells;
}
@end

Tutorial 2:
theButton.backgroundColor = [UIColor whiteColor]
Objective-C Accessor methods

@interface Grid: UIView
{
    int numberCells;
}
-(int) getNumberCells;
@end

@implementation Grid
-(int) getNumberCells
{
    return numberCells;
}
@end

@interface Grid: UIView
@property (readonly) int numberCells;
@end

@implementation
@synthesize numberCells;
@end
C++ vs. Objective-C

- Method
- Accessor methods
- Objects
## C++ vs. Objective-C

<table>
<thead>
<tr>
<th>C++: static &amp; dynamic objects</th>
<th>Objective-C: dynamic objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid grid;</td>
<td>Grid* grid = [[Grid alloc] init];</td>
</tr>
<tr>
<td>Grid* grid = new Grid;</td>
<td></td>
</tr>
</tbody>
</table>

Objective-C is a superset of C. structs can be created statically.
<table>
<thead>
<tr>
<th>C++: static &amp; dynamic objects</th>
<th>Objective-C: dynamic objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid grid;</td>
<td>Grid* grid = [[Grid alloc] init];</td>
</tr>
</tbody>
</table>
C++ vs. Objective-C

- Method
- Data members
- Objects
- Constructors
You provide constructor methods for your classes.

C++
Grid* grid = new Grid(numberCells);

Objective-C:
Grid* grid = [[Grid alloc] initWithNumberCells: numberCells];

You provide initialization methods for your classes.
C++ vs. Objective-C

- Method
- Data members
- Objects
- Constructors
- Destructors