Porting Swift 2.0 to Obj-c

Advanced NSOperation example

Ste Prescott

- Work at 3Squared
- @ste_prescott
- http://github.com/steprescott

What do we want?

Swift!

When do we want it?

NOW!

When do we get it?

When risk is low.

For now stick with what you know.

Do you even swift?

- Lack of experience or knowledge
- Risk of late delivery
- Is Swift stable enough?
- Requirement to move project over to Swift 2.0

Advanced NSOperations

- Dave DeLong (Fan of bowties)
- Sample Code written in Swift 2.0
- Swift 2.0 is still in beta

Porting Swift to Obj-C

- Basic Swift knowledge required
- Good knowledge of Obj-C needed

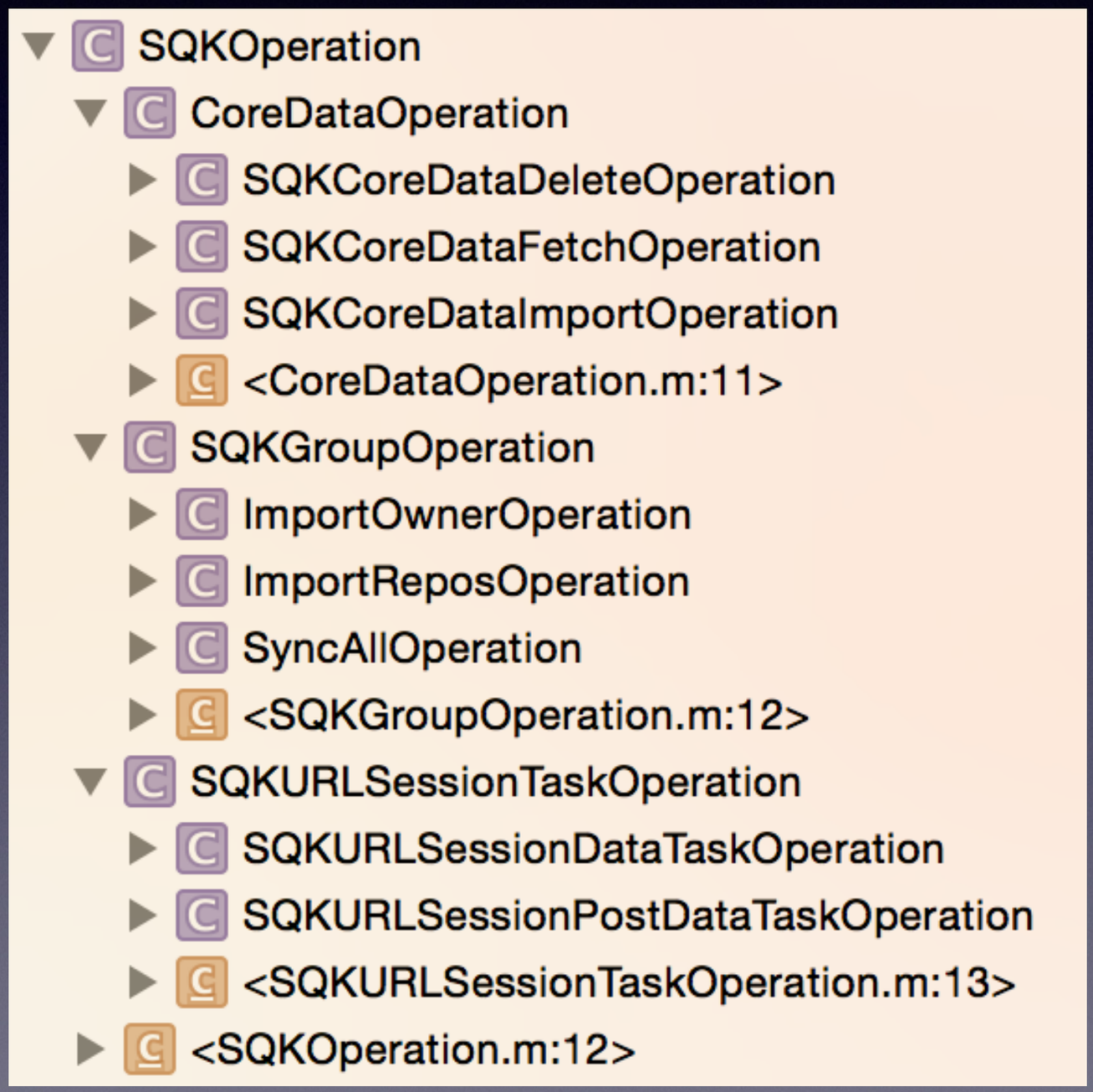
SQKOperation background

- Needed a tool to reduce boilerplate code for downloading and parsing JSON

- Wanted to reduce nesting of methods as much as possible but still guaranteeing execution order

- Keep off the main thread as much as possible

SQKOperation Class structure



Operation comparison

- Operation
  - keyPathsForValuesAffectingIsReady()
  - keyPathsForValuesAffectingIsExecuting()
  - keyPathsForValuesAffectingIsFinished()
  - **State Management**
  - canTransitionToState(_:)
  - willEnqueue()
  - *_state*
  - *stateLock*
  - *state*
  - *ready*
  - *userInitiated*
  - *executing*
  - *finished*
  - evaluateConditions()
  - **Observers and Conditions**
  - *conditions*
  - addCondition(_:)
  - *observers*
  - addObserver(_:)
  - addDependency(_:)
  - **Execution and Cancellation**
  - start()
  - main()
  - execute()
  - *_internalErrors*
  - cancelWithError(error:)
  - produceOperation(_:)
  - **Finishing**
  - finishWithError(_:)
  - *hasFinishedAlready*
  - finish(errors:)
  - finished(_:)
  - waitUntilFinished()
- <(_:_:)
- ==(_:_:)

- *@interface SQKOperation()*
  - *operationCompletionBlock*
  - *state*
  - *internalConditions*
  - *internalErrors*
  - *ownerOperations*
- @implementation SQKOperation
  - +keyPathsForValuesAffectingValueForKey:
  - -init
  - -initWithCompletionBlock:
  - -conditions
  - **State Management**
  - -setState:
  - -willEnqueue
  - -setUserInitiated:
  - -isReady
  - -evaluateConditions
  - -isExecuting
  - -isFinished
  - -isCancelled
  - **Conditions**
  - -addCondition:
  - -addDependency:
  - -addOwnerOperation:
  - -removeAllConditionOfClass:
  - -removeCondition:
  - -removeAllConditions
  - -removeOwnerOperation:
  - -removeFromOwnerOperations
  - **Execution and Cancellation**
  - -start
  - -execute
  - -cancel
  - -replacedWithOperation:
  - -cancelByNotAddingToQueue
  - **Finishing**
  - -finish
  - **Errors**
  - -operationErrors
  - -addError:
  - -addErrors:
  - -operation:didFinishWithErrors:
  - **Debug**
  - +stringForOperationState:

Swift goodness

- Generics
- Associated values
- Safety
- Easier to read

```
struct MutuallyExclusive<T>: OperationCondition {
    static var name: String {
        return "MutuallyExclusive<\(T.self)>"
    }

    static var isMutuallyExclusive: Bool {
        return true
    }

    init() { }

    func dependencyForOperation(operation: Operation) -> NSOperation? {
        return nil
    }

    func evaluateForOperation(operation: Operation, completion: OperationConditionResult -> Void) {
        completion(.Satisfied)
    }
}
```

Generics

- Ability to add a mutually exclusive condition to an operation of any type
- Obj-C workaround only allowed mutually exclusivity per-operation Class type.

```
addCondition(MutuallyExclusive<UIViewController>())
```

Associated values

- Obj-C workaround was for the condition to add the error to the operation rather than the error being associated with the enum

```
enum OperationConditionResult: Equatable {
    case Satisfied
    case Failed(NSError)

    var error: NSError? {
        if case .Failed(let error) = self {
            return error
        }

        return nil
    }
}
```

Safety

- **guard** keyword
- No need for if(object)

- Namespaces

```
case .Pending:
    // If the operation has been cancelled, "isReady" should return true
    guard !cancelled else {
        return true
    }
```

```
- (void)addError:(NSError *)error
{
    if(error)
    {
        [self.internalErrors addObject:error];
    }
}
```

Easier to read

- Obj-C is ugly in comparison

```objc
SQKCoreDataFetchOperation *ownerFetchRequest = [[SQKCoreDataFetchOperation alloc] initWithContextManager:[ContextManager sharedInstance]
                                                                                             fetchRequest:[Owner sqk_fetchRequest]
                                                                                          completionBlock:^void (SQKOperation *operation, NSArray *errors){
                                                                                              /**
                                                                                               * As the operation that has just finished may be any subclass of
                                                                                               * Operation we need to check and cast to the correct type.
                                                                                               */
                                                                                              if([operation isMemberOfClass:[SQKCoreDataFetchOperation class]])
                                                                                              {
                                                                                                  /**
                                                                                                   *  Once it is casted we can get the objects that the fetch
                                                                                                   *  operation fetched.
                                                                                                   */
                                                                                                  SQKCoreDataFetchOperation *op = (SQKCoreDataFetchOperation *)operation;
                                                                                                  Owner *owner = (Owner *)[op.fetchedItems lastObject];

                                                                                                  /**
                                                                                                   *  Tell the view contrller to move to the next view
                                                                                                   *  to show the data of the newly imported Owner.
                                                                                                   */
                                                                                                  dispatch_async(dispatch_get_main_queue(), ^{

                                                                                                      self.importedOwner = (Owner *)[[ContextManager mainContext] objectWithID:owner.objectID];
                                                                                                      [self.activityIndicator stopAnimating];
                                                                                                      self.goButton.hidden = NO;
                                                                                                      [self performSegueWithIdentifier:ShowOwnerSegueIdentifier sender:self];
                                                                                                  });
                                                                                              }
                                                                                          }];
```

Overall

- Not difficult to port Swift to Obj-C
- You will miss some Swift features
- Swift 2.0 is round the corner

Questions