Advanced Persistence
CMSC 436
Midterm

- Mean: 66% (median 68%)
- Std Dev: 14.4%
- Max: 95%
- Grade distribution:
  - A: 80.1% (25)
  - B: 65.7% (50)
  - C: 51.3% (50)
  - other (18)
- Regrade requests until Sunday night (Halloween)
Core Data

*Core Data* is a database

- Object-relational model (object graph)
- On disk, usually backed by SQLite
- Supports set and fetch
- Handles serialization/deserialization for you
- Very performant

Easiest way to add Core Data to your app — select checkbox at app creation

See


for a good overview
Components of Core Data

xcdatamodeld Where you declare your core data model
  - Object types are *Entities*
  - Auto-generates code
  - Can generate swift files

Persistence.swift Defines a PersistenceController
  - Singleton instance `.shared`
  - Creates a container for objects
  - Container has a `viewContext` for in-memory cache
  - Lets you create a special instance for preview

Wiring into app Set `viewContext as \.managedObjectContext` in environment

Using in view Bind `\.managedObjectContext` from environment
Data Model

Define your *Entities* in the `xcdatamodeld` editor:

Default *Codegen* is *Class Definition*

Switch to *Manual/None* if you’re going to manually generate
Generating Code

Combining Thingy+CoreDataClass.swift and Thingy+CoreDataProperties.swift:

```swift
import Foundation
import CoreData

@objc(Thingy)
public class Thingy: NSManagedObject {
    @NSManaged public var a: Int16
    @NSManaged public var b: String?
    @NSManaged public var c: Bool
    @NSManaged public var anotherThingy: AnotherThingy?
}

extension Thingy : Identifiable {
}
```

```
@objc(Thingy)
public class Thingy: NSManagedObject {
    @NSManaged public var a: Int16
    @NSManaged public var b: String?
    @NSManaged public var c: Bool
    @NSManaged public var anotherThingy: AnotherThingy?
}

extension Thingy : Identifiable {
}
```
Using Core Data in View (and more Navigation!)

Note: Xcode might generate broken template!

```swift
struct ContentView: View {
    @Environment(
        \.managedObjectContext)
    private var viewContext

    @FetchRequest(
        sortDescriptors: [NSSortDescriptor(keyPath: \Thingy.a, ascending: true)],
        animation: .default)
    private var items: FetchedResults<Thingy>

    var body: some View {
        NavigationView {
            List {
                ForEach(items) { item in Text("Item at \(item.a)") }
                .onDelete(perform: deleteItems)
            }
            .toolbar {
                ToolbarItem(placement: .navigationBarLeading) { EditButton() }
                ToolbarItem(placement: .navigationBarTrailing) {
                    Button(action: addItem) {
                        Label("Add Item", systemImage: "plus")
                    }
                }
            }
        }
    }

    private func addItem() { ... }
    private func deleteItems(offsets: IndexSet) { ... }
}
```
Adding and Deleting Items

`viewContext.save()` persists the changes

```swift
private func addItem() {
    withAnimation {
        let newItem = Thingy(context: viewContext)
        newItem.a = 0
        newItem.b = ""
        newItem.c = false

        do {
            try viewContext.save()
        } catch {
            let nsError = error as NSError
            fatalError("Unresolved error \(nsError), \(nsError.userInfo)"
        }
    }
}

private func deleteItems(offsets: IndexSet) {
    withAnimation {
        offsets.map { items[$0] }.forEach(viewContext.delete)

        do {
            try viewContext.save()
        } catch {
            let nsError = error as NSError
            fatalError("Unresolved error \(nsError), \(nsError.userInfo)"
        }
    }
}
```