Module 3: Object-Orientation in Go
Topic 1.3: Support for Classes

Structs, again

- Struct types compose data fields

```
type Point struct {
     x float64
     y float64
}
```

- Traditional feature of classes

Structs with Methods

- **Structs and methods** together allow arbitrary data and functions to be composed

```
func (p Point) DistToOrig() {
   t := math.Pow(p.x, 2) +
     math.Pow(p.y, 2)
   return math.Sqrt(t)
}
func main() {
   p1 := Point(3, 4)
   fmt.Println(p1.DistToOrig())
}
```

UCI Division of Continuing Education

Encapsulation in Go

- Making data fields or methods hidden from the programmer
- Might use a `private` keyword in another language
- Example: Point struct, `Scale()` method
- `Scale()` should multiply x and y coordinates by a constant
- Don't trust this to the programmer
  - Might scale one coordinate but not the other
  - Coordinates could become inconsistent
  - Need to **hide x and y coordinates**

Hiding in a Package

- Go can only hide data/methods in a package
- Variables/functions are only exported if their names start with a **capital letter**

```
package data
var x int = 1
var Y int = 2
```

```
package main
import “data"
func main() {
      fmt.Println(Y)
      fmt.Println(x)
}
```

UCI Division of Continuing Education