Object Oriented
vs
Golang Approach

Saber Mesgari

Is Golang OO?

- Yes and No

- What is Object Oriented Programming?
  - is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as *attributes;* and code, in the form of procedures, often known as *methods*
  -
  - an Object's procedures can access and often modify the attributes of the object with which they are associated

  - an Object's **internal state is protected** from outside world (encapsulated) leveraging **private/protected/public** visibility of attributes and methods

  - an Object is frequently defined in OO languages as an instance of a Class

Is Golang OO?

- How Languages Implement it?
  - Encapsulation (**possible** on package level in Go)
  - Composition (**possible** through embedding in Go)
  - Polymorphism (**possible** through Interface satisfaction in Go)
  - Inheritance (Go **does not provide**)

Structs

```
type Creature struct {
    Name string
    Real bool
}
```

Methods

```
func (c Creature) Dump() {
  fmt.Printf("Name: '%s', Real: %t\n", c.Name, c.Real)
}
```

Embedding

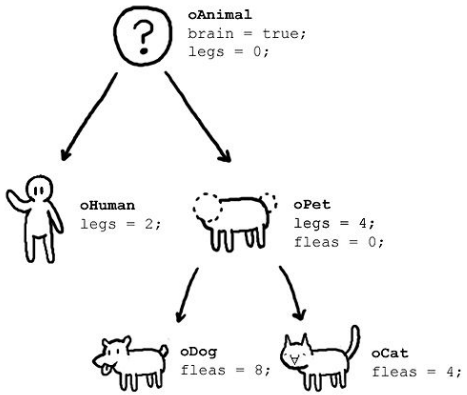
```
type FlyingCreature struct {
  Creature
  WingSpan int
}


dragon := &FlyingCreature{
    Creature{"Dragon", false, },
    15,
}
fmt.Println(dragon.Name)
fmt.Println(dragon.Real)
fmt.Println(dragon.WingSpan)
```

Encapsulation

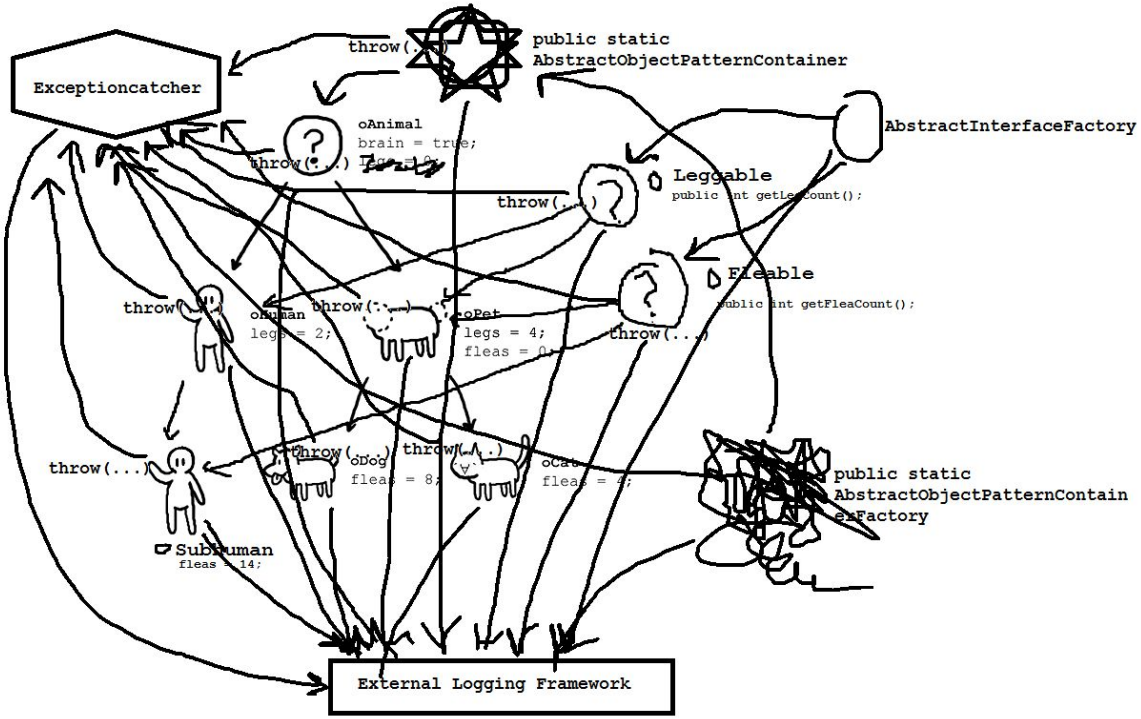
```
type foo struct {
}
func (f foo) Foo1() {
    fmt.Println("Foo1() here")
}
func (f foo) Foo2() {
    fmt.Println("Foo2() here")
}
func (f foo) Foo3() {
    fmt.Println("Foo3() here")
}
func NewFoo() Fooer {
    return &Foo{}
}
```

Inheritance

What OOP users claim



What actually happens



Constructors

```
type Creature struct {
    Name string
    Real bool
}

func NewCreature(name string, real bool) Creature {
    return Creature{Name: name, Real: real}
}
```

Pointer Reciever vs Value Reciever

```
type Vertex struct {
    X, Y float64
}
func (v *Vertex) Scale(f float64) {
    v.X = v.X * f
    v.Y = v.Y * f
}
func (v *Vertex) Abs() float64 {
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}
func main() {
    v := &Vertex{3, 4}
    fmt.Printf("Before scaling: %+v, Abs: %v\n", v, v.Abs())
    v.Scale(5)
    fmt.Printf("After scaling: %+v, Abs: %v\n", v, v.Abs())
}
```