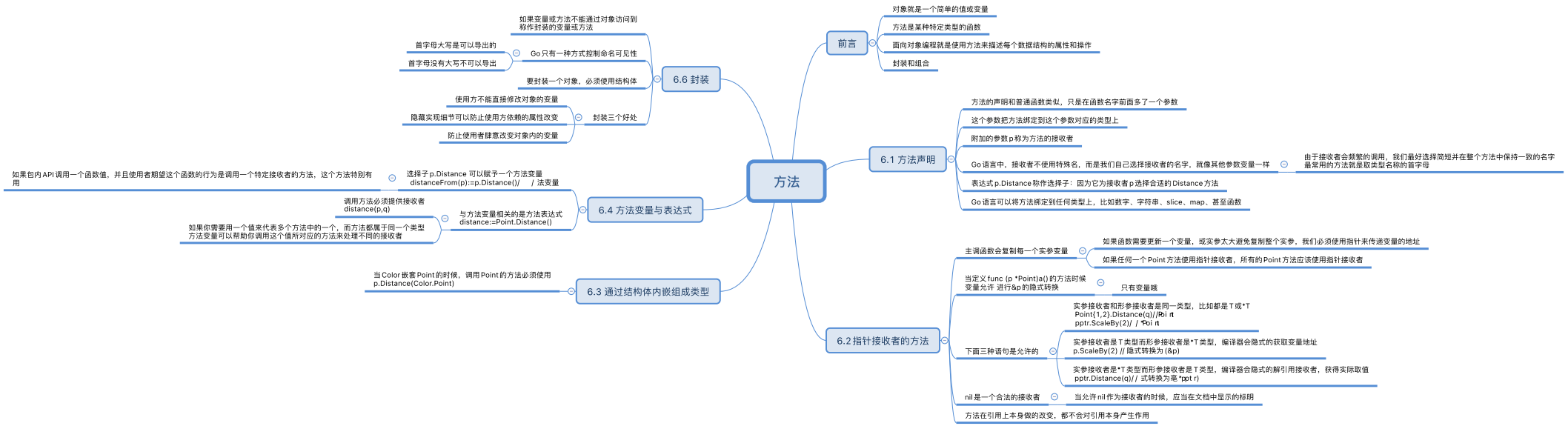
方法
前言
对象就是一个简单的值或变量
方法是某种特定类型的函数
面向对象编程就是使用方法来描述每个数据结构的属性和操作
封装和组合
6.1 方法声明
方法的声明和普通函数类似，只是在函数名字前面多了一个参数
这个参数把方法绑定到这个参数对应的类型上
附加的参数 p 称为方法的接收者
Go 语言中，接收者不使用特殊名，而是我们自己选择接收者的名字，就像其他参数变量一样
由于接收者会频繁的调用，我们最好选择简短并在整个方法中保持一致的名字
最常用的方法就是取类型名称的首字母
表达式 p.Distance 称作选择子：因为它为接收者 p 选择合适的 Distance 方法
Go 语言可以将方法绑定到任何类型上，比如数字、字符串、slice、map、甚至函数
6.2 指针接收者的方法
主调函数会复制每一个实参变量
如果函数需要更新一个变量，或实参太大避免复制整个实参，我们必须使用指针来传递变量的地址
如果任何一个 Point 方法使用指针接收者，所有的 Point 方法应该使用指针接收者
当定义 func (p *Point)a() 的方法时候
变量允许 进行&p 的隐式转换
只有变量哦
下面三种语句是允许的
实参接收者和形参接收者是同一类型，比如都是 T 或 *T
Point{1,2}.Distance(q)//Poi nt
pptr.ScaleBy(2)// Poi nt
实参接收者是 T 类型而形参接收者是 *T 类型，编译器会隐式的获取变量地址
p.ScaleBy(2) // 隐式转换为 (&p)
实参接收者是 *T 类型而形参接收者是 T 类型，编译器会隐式的解引用接收者，获得实际取值
pptr.Distance(q)// 式转换为毫 *ppt r)
nil 是一个合法的接收者
当允许 nil 作为接收者的时候，应当在文档中显示的标明
方法在引用上本身做的改变，都不会对引用本身产生作用
6.3 通过结构体内嵌组成类型
当 Color 嵌套 Point 的时候，调用 Point 的方法必须使用 p.Distance(Color.Point)
6.4 方法变量与表达式
选择子 p.Distance 可以赋予一个方法变量 distanceFrom(p):=p.Distance()// 法变量
如果包内 API 调用一个函数值，并且使用者期望这个函数的行为是调用一个特定接收者的方法，这个方法特别有用
与方法变量相关的是方法表达式 distance:=Point.Distance()
调用方法必须提供接收者 distance(p,q)
如果你需要用一个值来代表多个方法中的一个，而方法都属于同一个类型 方法变量可以帮助你调用这个值所对应的方法来处理不同的接收者
6.6 封装
如果变量或方法不能通过对象访问到 称作封装的变量或方法
Go 只有一种方式控制命名可见性
首字母大写是可以导出的
首字母没有大写不可以导出
要封装一个对象，必须使用结构体
封装三个好处
使用方不能直接修改对象的变量
隐藏实现细节可以防止使用方依赖的属性改变
防止使用者肆意改变对象内的变量