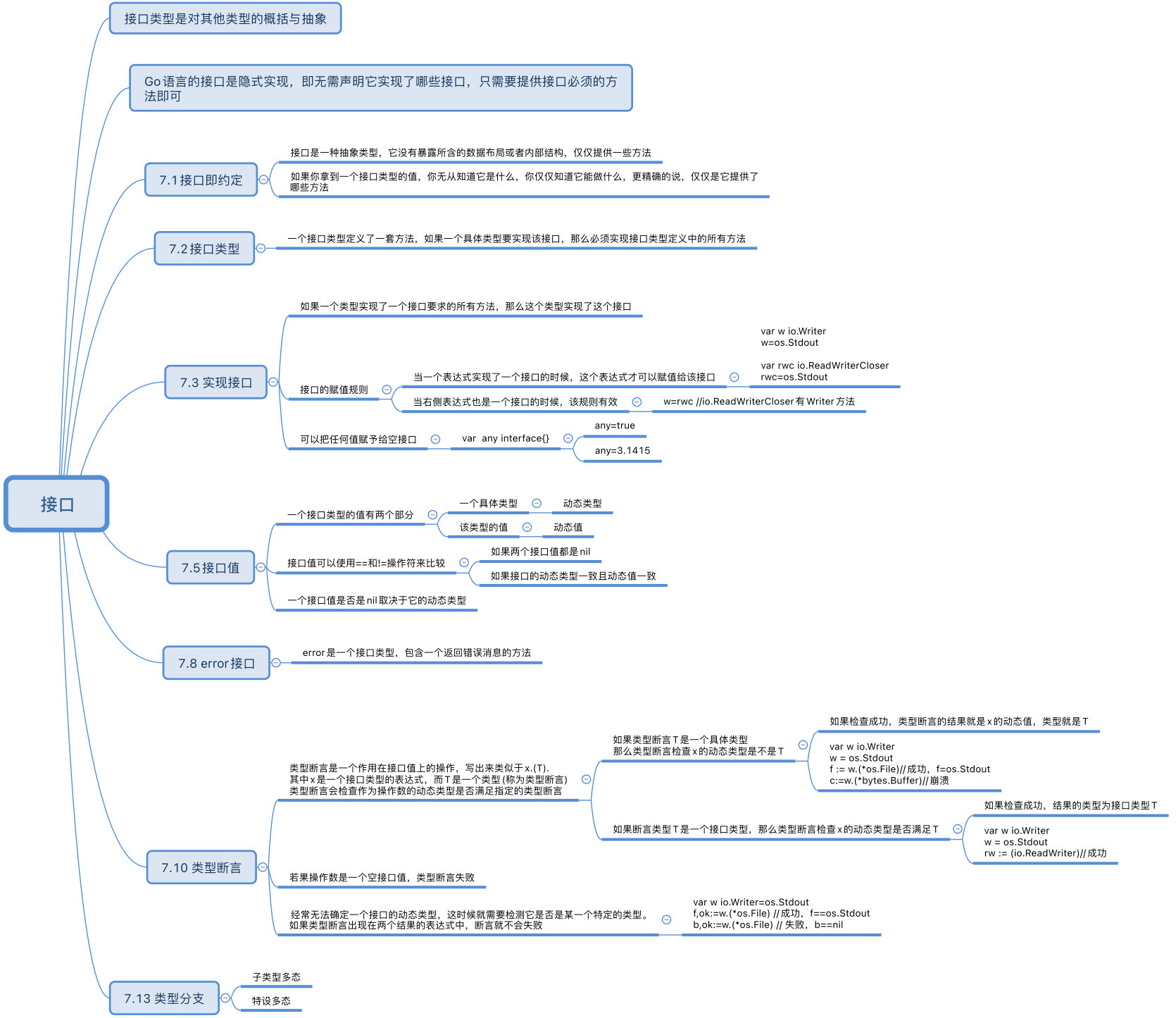
接口
接口类型是对其他类型的概括与抽象
Go语言的接口是隐式实现，即无需声明它实现了哪些接口，只需要提供接口必须的方法即可
7.1接口即约定
接口是一种抽象类型，它没有暴露所含的数据布局或者内部结构，仅仅提供一些方法
如果你拿到一个接口类型的值，你无从知道它是什么，你仅仅知道它能做什么，更精确的说，仅仅是它提供了哪些方法
7.2接口类型
一个接口类型定义了一套方法，如果一个具体类型要实现该接口，那么必须实现接口类型定义中的所有方法
7.3 实现接口
如果一个类型实现了一个接口要求的所有方法，那么这个类型实现了这个接口
接口的赋值规则
当一个表达式实现了一个接口的时候，这个表达式才可以赋值给该接口
var w io.Writer
w=os.Stdout
var rwc io.ReadWriterCloser
rwc=os.Stdout
当右侧表达式也是一个接口的时候，该规则有效
w=rwc //io.ReadWriterCloser有Writer方法
可以把任何值赋予给空接口
var any interface{}
any=true
any=3.1415
7.5接口值
一个接口类型的值有两个部分
一个具体类型
动态类型
该类型的值
动态值
接口值可以使用==和!=操作符来比较
如果两个接口值都是nil
如果接口的动态类型一致且动态值一致
一个接口值是否是nil取决于它的动态类型
7.8 error接口
error是一个接口类型，包含一个返回错误消息的方法
7.10 类型断言
类型断言是一个作用在接口值上的操作，写出来类似于x.(T).
其中x是一个接口类型的表达式，而T是一个类型(称为类型断言)
类型断言会检查作为操作数的动态类型是否满足指定的类型断言
如果类型断言T是一个具体类型
那么类型断言检查x的动态类型是不是T
如果检查成功，类型断言的结果就是x的动态值，类型就是T
var w io.Writer
w = os.Stdout
f := w.(*os.File)//成功，f=os.Stdout
c:=w.(*bytes.Buffer)//崩溃
如果断言类型T是一个接口类型，那么类型断言检查x的动态类型是否满足T
如果检查成功，结果的类型为接口类型T
var w io.Writer
w = os.Stdout
rw := (io.ReadWriter)//成功
若果操作数是一个空接口值，类型断言失败
经常无法确定一个接口的动态类型，这时候就需要检测它是否是某一个特定的类型。
如果类型断言出现在两个结果的表达式中，断言就不会失败
var w io.Writer=os.Stdout
f,ok:=w.(*os.File) //成功，f==os.Stdout
b,ok:=w.(*os.File) // 失败，b==nil
7.13 类型分支
子类型多态
特设多态