面向对象

面向对象的基本概念

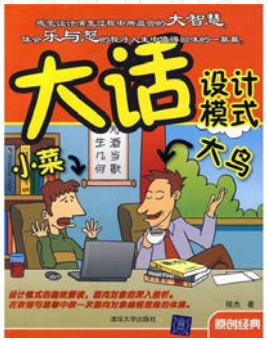
new 一个对象

面向对象的基本概念

- 本质上就是解决如何用程序描述世界的问题
- 讨论如何把实际存在的东西映射成程序的类和对象
- 一种程序设计的思路、思想、方法

面向对象的基本概念

- 程序设计层面的概念
- 设计模式：前人的程序设计经验



抽象类与接口

半成品与协议

什么是接口

- 接口，直观理解就是一种约定

- 举例，输入设备接口：

```
interface InputDevice{
    fun input(event: Any)
}
```

接　口

- 不能有状态
- 必须由类对其进行实现后使用

抽象类

- 实现了一部分协议的半成品
- 可以有状态，可以有方法实现
- 必须由子类继承后使用

抽象类和接口的共性

- 比较抽象，不能直接实例化
- 有需要子类（实现类）实现的方法
- 父类（接口）变量可以接受子类（实现类）的实例赋值

抽象类和接口的区别

- 抽象类有状态，接口没有状态
- 抽象类有方法实现，接口只能有无状态的默认实现
- 抽象类只能单继承，接口可以多实现
- 抽象类反映本质，接口体现能力

子承父业的故事

虎父无犬子


孙策
激昂
魂姿
制霸
武烈帝
孙坚
英魂
回合开始阶段，若你已受伤，可选择一名其他角色执行下列两项中的一项：
1. 摸X张牌，然后弃一张牌.
2. 摸一张牌，然后弃X张牌.
X为你已损失的体力值，每回合限一次.
林
孙权
制衡
救援

这是迄今为止
我所知道的最容易的
《Teach Yourself C++ in 21 Days》

继承（实现）语法要点

- 父类需要 open 才可以被继承
- 父类方法、属性需要 open 才可以被覆写
- 接口、接口方法、抽象类默认为 open
- 覆写父类（接口）成员需要 override 关键字

继承（实现）语法要点

- class D: A(), B, C
- 注意继承类时实际上调用了父类构造方法
- 类只能单继承，接口可以多实现

接口代理

- class Manager(driver: Driver): Driver by driver
- 接口方法实现交给代理类实现

接口方法冲突

- 接口方法可以有默认实现
- 签名一致且返回值相同的冲突
- 子类（实现类）必须覆写冲突方法
- super<[父类（接口）名]>.[方法名]([参数列表])

类及其成员的可见性

就不给你看

可见性对比

Kotlin	Java
private	private
protected	protected
-	default（包内可见）
internal（模块内可见）	-
public	public

object

单例

object

- 只有一个实例的类
- 不能自定义构造方法
- 可以实现接口、继承父类
- 本质上就是单例模式最基本的实现

伴生对象与静态成员

public static void main✕

伴生对象与静态成员

- 每个类可以对应一个伴生对象
- 伴生对象的成员全局独一份
- 伴生对象的成员类似 Java 的静态成员

伴生对象与静态成员

- 静态成员考虑用包级函数、变量替代
- JvmField 和 JvmStatic 的使用

方法重载和默认参数

看我七十二变

方法重载

- Overloads
- 名称相同、参数不同的方法
- Jvm函数签名的概念：函数名、参数列表
- 跟返回值没有关系

默认参数

- 为函数参数设定一个默认值
- 可以为任意位置的参数设置默认值
- 函数调用产生混淆时用具名参数

方法重载与默认参数

- 二者的相关性以及@JvmOverloads
- 避免定义关系不大的重载
- 不好的设计：
  - List.remove(int)
  - List.remove(Object)

扩展成员

二次加工

扩展成员

- 为现有类添加方法、属性
  - fun X.y(): Z { ... }
  - val X.m 注意扩展属性不能初始化，类似接口属性
- Java 调用扩展成员类似调用静态方法

属性代理

曹操与献帝

属性代理

- 定义方法：
  - val/var <property name>: <Type> by <expression>
- 代理者需要实现相应的 setValue/getValue 方法
- lazy 原理剖析

数据类

超时坑要塞

数据类

- 再见，JavaBean
- 默认实现的 copy、toString 等方法
- componentN 方法
- allOpen 和 noArg 插件

内部类

你被包围了！

内部类

- 定义在类内部的类
- 与类成员有相似的访问控制
- 默认是静态内部类，非静态用 inner 关键字
- this@Outter，this@Inner 的用法

匿名内部类

- 没有定义名字的内部类
- 类名编译时生成，类似 Outter$1.class
- 可继承父类、实现多个接口，与 Java 注意区别

枚 举

数学不好的进！

枚　举

- 实例可数的类，注意枚举也是类
- 可以修改构造，添加成员
- 可以提升代码的表现力，也有一定的性能开销