# 枚举类

底层就是一个类，枚举类型其实就是定义的一个个类，都是静态的常量，默认继承Enum，并且是设置私有的构造函数，不给外界实例化

像枚举的形式使用

一个个个列举

源码 目的 主要分析，学习，了解

static

final常量

封装性，set/get

继承性 Enum

重写

构造函数私有化

# 枚举扩展

主要是构造函数的定义

# 常用API

值，名字，其他值

# 为何使用枚举？

方便，直观，不容易出错

# 内部类

一个类定义在其它类中，这个类就叫内部类

类的内部，而不是同一个文件内（还是两个类）

两个类，一个类中private

其他类能使用这个private

所以为能使用其他类中种的private

在这个类中定义一个类

定义多个类，那么就使用内部类

我们必须要定义类，而且只能也在这里用一次。别的地方不需要

首先，要定类，能访问private ，只能在当前类中使用

考虑内部类

枚举类型 类的特例

内部类 类的特例

分类

动态的内部类

就是成员内部类

静态的内部类

类内部类

匿名内部类

没有名字的内部类

内部类都有类的名字

Inner

Inner2

就是没有名字的类

少写了一个接口的实现类火灾这个类的子类

只要用一次，就没必要定义实现类或者子类了

不建议大家在一个文件中定义多个类

在编译阶段就已经识别了

静态的方法，重写，有 ，调用的时候要根据定义的类型的，类似于属性

方法 就是多态的表现

静态方法没有多态的表现

# 垃圾回收

A a=new A()

new 内存中开辟空间

开辟之后有没有释放空间

a 对象有可能后面不再需要了

用完之后需要释放空间

构造函数 分配空间的

析构函数 ~ 释放空间

内存不够用程序员手动释放空间

保准每个程序员都用完之后释放空间吗？

java中变得简单了，jvm自动回收内存。

重点它要知道这个对象是不是不用了

算法问题

java程序员没有这个释放内存的习惯

交给GC做了

垃圾回收器

Obect o=new Object()

…..

…..

…

o=null;快速告诉GC ，不用，你可以收回了

# 核心类库

异常类

常用类 Math String 包装类 ，格式化 等

输入输出流

当程序发生异常了，我们要去处理的事情

但是发生了错误。程序就会终止

异常 可以捕获，可以控制的，程序不会终止

错误 没法捕获的，程序遇到错误就终止运行

提高代码的可读性，减少代码逻辑处理

只要有错误，那就捕获，处理，不至于程序终止