# set方法和get方法

## set方法和get方法的使用场合

@public的成员可以被随意赋值，应该使用set方法和get方法来管理成员的访问（类似机场的安检、水龙头过滤，过滤掉不合理的东西），比如僵尸的生命值不能为负数

## set方法

1. 作用：用来设置成员变量，可以在方法里面过滤掉一些不合理的值
2. 命名规范：

* 方法都是以set开头，而且后面跟上成员变量名，成员变量名的首字母必须大写
* 形参名称不要跟成员变量同名

## get方法

1. 作用：返回对象内部的成员变量
2. 命名规范：get方法的名称一般就跟成员变量同名

## 成员变量的命名规范

* 成员变量都以下划线 \_ 开头
* 可以跟get方法的名称区分开
* 可以跟其他局部变量区分开，一看到下划线开头的变量，肯定是成员变量

## 代码示例

#import <Foundation/Foundation.h>

// 声明

@interface Car : NSObject

{

int \_wheels; // 轮子个数

}

/\*set方法\*/

- (void) setWheels:(int)wheels;

/\*get方法\*/

- (int) wheels;

@end

@implementation Car

// set方法的实现

- (void) setWheels:(int)wheels

{

// 对外面传进来的轮子数进行过滤

if (wheels<=0)

{

wheels = 1;

}

\_wheels = wheels;

}

// get方法的实现

- (int) wheels

{

return \_wheels;

}

@end

## 封装的好处

* 过滤不合理的值
* 屏蔽内部的赋值过程
* 让外界不必关注内部的细节

## 练习

1. Dog类，属性：weight、speed，方法：吃、跑
2. Person类，属性：dog、age，方法：喂狗、遛狗

# 类方法

## 基本概念

直接可以用类名来执行的方法（类本身会在内存中占据存储空间，里面有类\对象方法列表）

## 类方法和对象方法对比

1. 对象方法

* 以减号-开头
* 只能让对象调用，没有对象，这个方法根本不可能被执行
* 对象方法能访问实例变量（成员变量）

1. 类方法

* 以加号+开头
* 只能用类名调用，对象不能调用
* 类方法中不能访问实例变量（成员变量）
* 使用场合：当不需要访问成员变量的时候，尽量用类方法

1. 类方法和对象方法可以同名

## 示例

* 设计一个方法输出类名
* 计算器类

# self关键字

## 成员变量和局部变量同名

* 当成员变量和局部变量同名时，采取就近原则，访问的是局部变量
* 用self访问成员变量，区分同名的局部变量

## 使用细节

1. 出现的地方：所有的OC方法中（对象方法\类方法），不能出现在函数
2. 作用

* 使用 "self->成员变量名" 访问当前方法调用的成员变量
* 使用 "[self 方法名];" 来调用方法（对象方法\类方法）

## 常见错误

* 低级错误：用self去调用函数
* 类方法中用self调用对象方法，对象方法中用self调用类方法
* self死循环

## 示例

* 狗在跑之前叫两声
* 计算器类，计算平均值的调用求和方法

# 习题

1. 设计Car类，一个对象方法跟其他车子比较车速，返回速度差，一个类方法比较两辆车的车速，返回速度差