# 抽象

## 概念

把想法从实例中抽取出来，根据他们的功能创建类

## 目的

在于把行为和实施细节分开

## 优点

1：提高代码复用性；强制子类实现父类中没有实现的功能

2：提高代码的扩展性，便于后期的代码维护

## 注意

1.不能实例化，只能通过通过调用子类的重写和继承，抽象的多态化，抽象类声明的对象做上转型对象

2. abstract，final，static，static



# 抽象类/方法abstract—强制继承

## 概述

抽象就是看不懂的

抽象类： abstract class A{

}

抽象方法： public abstract void cry(); --没有方法体

## 特点

1.抽象类不一定有抽象方法，有抽象方法的一定是抽象类或接口

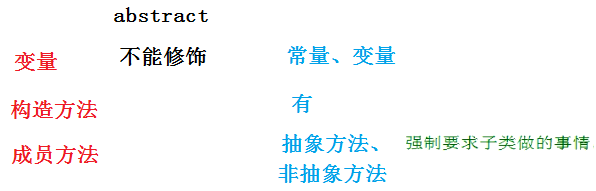
2. abstract不能用new创建对象，

通过调用子类的重写和继承。抽象的多态化，抽象类声明的对象做上转型对象。

3.抽象类的子类

抽象类，类（重写抽象类的所有抽象方法）

## 抽象类的成员构成—抽象类体



# 接口-抽象的最高实现

为了克服Java单继承的缺点，Java使用了接口，一个类可以实现多个接口。

## 概述

狭义：Java中的interface

广义：对外 提供规则

声明：interface 接口

实现：class 类名 implements 接口名

java中的接口主要是使用来拓展定义类的功能，可以弥补java中单继承的缺点。

## 特点

1.接口只有抽象方法，有抽象方法的一定是抽象类或接口

2. 不能用new创建对象，

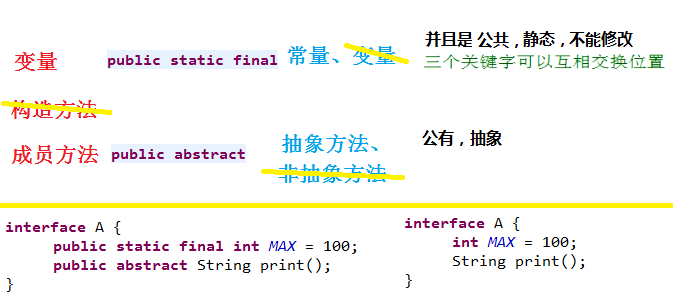
通过调用子类的实现

3.接口的子类

抽象类，

类（重写抽象类的所有抽象方法） --推荐

## 接口的成员构成—接口体



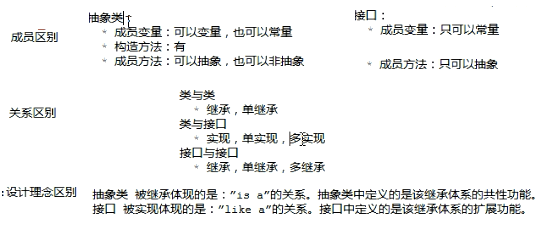
## 实现接口



# 抽象类与接口区别

含有abstract修饰符的class即为抽象类，abstract 类不能创建的实例对象。含有abstract方法的类必须定义为abstract class，abstract class类中的方法不必是抽象的。abstract class类中定义抽象方法必须在具体(Concrete)子类中实现，所以，不能有抽象构造方法或抽象静态方法。如果的子类没有实现抽象父类中的所有抽象方法，那么子类也必须定义为abstract类型。

接口（interface）可以说成是抽象类的一种特例，接口中的所有方法都必须是抽象的。接口中的方法定义默认为public abstract类型，接口中的成员变量类型默认为public static final。





# 类，接口之间的关系

