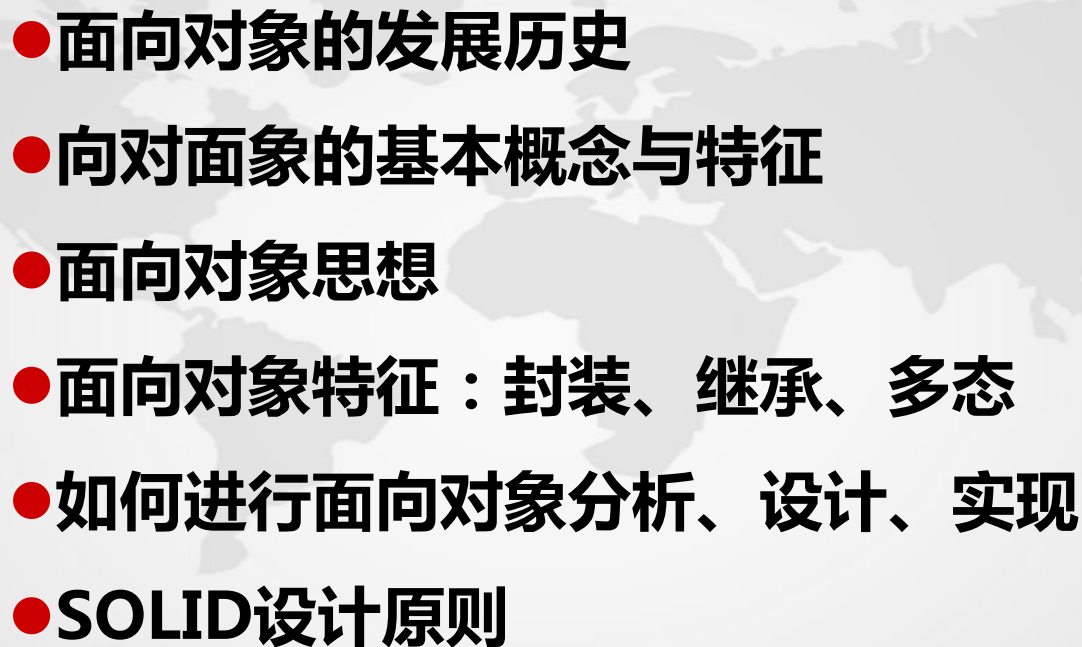
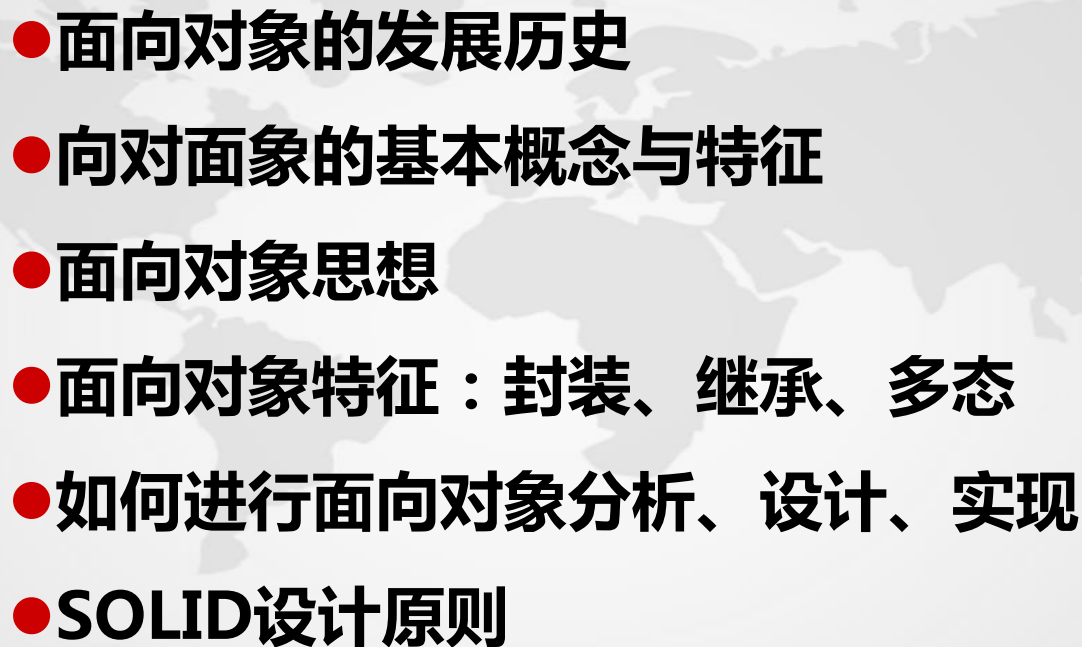



面向对象 OO★

—— 无对象，不编程 ——

- 
- 
- 
- **面向对象的发展历史**
 - **面向对象的基本概念与特征**
 - **面向对象思想**
 - **面向对象特征：封装、继承、多态**
 - **如何进行面向对象分析、设计、实现**
 - **SOLID设计原则**

面向对象的历史

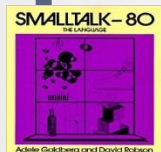
20世纪
80年代至今



80年代以来，面向对象扩展到其他领域，在许多工程和领域应用获得极大的发展。

1986年，在美国举办了“面向对象编程、系统、语言和应用（OOPSLA'86）”国际会议，面向对象普及到全世界，从此**“没有对象还编什么程”**

20世纪
70年代



SmallTalk语言诞生，强调对象概念，引入了对象，类，方法，实例，采用动态联编和单继承机制
1980年，施乐公司推出SmallTalk80，商品化的OO语言

20世纪
60年代



Simula语言提出了对象的概念，使用了类，支持继承

20世纪
50年代



ALGO语言设计者尝试提供封装保护
程序块结构开始广泛应用于高级语言中

面向对象

面向数据

数据稳定

数据隐藏

数据有形

聚焦问题领域

程序 = 数据 + 算法
构建程序可视为构造模型

Example: Matrix addition

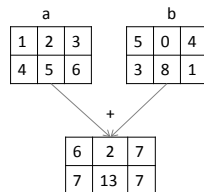
Traditional version:

a and b are a two-dimensional arrays

```
for i = 0, lines
  for j = 0, cols
    a(i, j) = a(i, j) + b(i, j)
  end for
end for
```

a and b are variables of type Matrix

a = a.add(b)



面向过程

面向算法

算法易变

数据可见

算法无形

聚焦单个问题

你有对象吗？



```
it include a stream>  
using namespace std;  
int main() {  
    cout << "Hello World!" << endl;  
    return 0;  
}
```

对象 Object = 物体
对象是世界中的物体在
人脑中的映象

对象是世界上的物体
在电脑中的映象

抽象



姓名
性别
身高

.....

走路();
吃饭();
说话();
思考();

.....

学号
班级
专业
课程

.....

选课();
学习();
考试();

.....



编号
工龄
职称
职务

.....

授课();
监考();

.....



爱好

.....

逛街();
亲吻();

.....



面向对象思想

把客观对象**抽象**成属性数据和对数据的相关操作，把内部细节和不相关的信息隐藏起来，把同一个类型的客观对象的属性数据和操作绑定在一起，**封装**成**类**，并且允许分成不同层次进行抽象，通过**继承**实现属性和操作的共享，实现类的泛化（通用性），通过**多态**来实现类的同一个任务（动作）分派到不同的对象去执行



Java表示法:
`class` 类名 {
 //类主体
}

- 同类对象的模版
- 抽象的结果: 类
`class` 手机

- 万物皆可为对象
- 即可有形，也可无形

消息
MESSAGE

类
CLASS

对象
OBJECT

- 对象的操作，对象之间通信
- 接受消息对象、消息名、参数
手机A.打电话('13912345678')

对象实例=Instance
(零件)

类=Class
(模子)

• 与护照, 身份证内容一致, 正确填写。

• 本人在广东省内的实际居住地址, 电话, 手机。无固定电话就相应填“无”。

• 工作单位真实填写, 无职, 退休, 学生分别填写“无职”, “退休”, “学生”。

• 本人出国期间可联系到的亲属朋友同事等。

• 与护照内容一致。

• 真实填写

• 旅行社统一填写

• 旅游路线由旅行社统一填写, 客人不用填。

• 访日目的统一在“观光”处打勾。

• 写出赴日旅游的其中一个同行行人, 没有的就写“无”。

• 外币栏客人不用填。

• 填写上一次的出境记录, 没有的就写“无”。

• 在日亲属有无要求务必如实填写。

• 填写申请日期。

• 由本人亲笔签名(幼儿由监护人代签)。

申请人姓名(汉字)	张明	(照片)
(拼音)	ZHANG MING	二寸
出生年月日:	1978年8月8日	出生地: 广东省/市
身份证号码:	44010000001234567	
性别:	<input checked="" type="checkbox"/> 男 <input type="checkbox"/> 女	
现住所: 中国	广东省/市	广州市白云区机场西乐嘉路1号
住宅电话:	020-81234567	手机号码: 13912345678
单位名称(学校名):	广之旅	电话号码: 020-86338888
职位:	职员	
紧急联络人(中国国内的亲属等)		
姓名:	张大明	与申请人的关系: 父子
地址:	广州市白云区机场西远景路1号	电话号码: 020-86330000
护照签发地:	广东	护照号码: G12345678
护照签发日期:	2008年7月1日	有效期限: 2018年6月30日
犯罪记录	<input checked="" type="checkbox"/> 无 <input type="checkbox"/> 有	
组团社名称:	不填	申请日期: 不填
参团费用:	不填 人民币	
预定出发日:	不填	预定归国日: 不填
旅游路线:	不填	
访日目的:	<input type="checkbox"/> 视察旅行 <input checked="" type="checkbox"/> 观光 <input type="checkbox"/> 修学旅行 <input type="checkbox"/> 探亲或访友 <input type="checkbox"/> 交流	
	<input type="checkbox"/> 其他()	
同行者姓名(亲属、朋友、同事等):	王梅	与同行者关系: 夫妻
出境所携外币:	日元 不填 元 美元 不填 美元	
出境纪录:	无 <input checked="" type="checkbox"/> 有 07年10月 共5日	前往国家名: 新加坡
在日亲属:	<input checked="" type="checkbox"/> 无 <input type="checkbox"/> 有 在日亲属姓名: 关系:	
职业:	家庭电话:	
工作单位(学校名):		
填写申请日期	2008年11月1日	申请人签名: 张明



对象必须实例化才能使用(Java/C++/C#):
`class Student{ String: 学号; };`

`Student 初二博士 = new Student();`
`初二博士.学号 = 'WBD1234567';`

类是虚的, 对象是实的, 对象是类的举例

为什么要
封装？



面向对象主要特征：封装

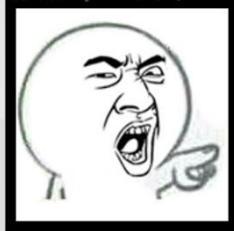
封装是一种信息隐蔽技术，它体现于类的说明。封装使数据和加工该数据的方法（函数）封装为一个整体，以实现独立性很强的模块，使得用户只能见到对象的外特性，而对象的内特性对用户是隐蔽的。

* 能不给就不给，最小原则，逢人只说三分话，未可全抛一片心



关我屁事

放开她，让我来！



- ① 将变化隔离
- ② 便于使用
- ③ 提高重用性
- ④ 提高安全性



封装的三个层次

私有
Private
仅自己用

存款金额
房产
股票
黄金
债券

保护
Protected
亲友可用

计算资产();
投资();

公有
public
大家可用

查询存款();
查询资产();



方法



Java表示法

```
class 财富 {  
    private float 存款;  
    private float 房产;  
    private float 股票;  
    private float 黄金;  
    private float 债券;
```

属性

```
    protected 计算资产(): float;  
    protected 投资(项目);
```

```
    public 查询存款(): float;  
    public 查询资产(): float;  
    public 设置存款(存款: float);  
    public 设置股票(股票: float);
```

.....

```
}
```

get, set方法

龙生龙，凤
生凤，老鼠
生崽会打洞

面向对象主要特征：继承

继承性是子类自动共享父类数据和方法的机制，它由类的派生功能体现。一个类直接继承其它类的全部描述，同时可修改和扩充。

继承具有传递性，继承分为单继承（一个子类只有一父类）和多重继承（一个类有多个父类）。

王叔叔好～

呃...

乖～又长
大了啊～

继承简化了人们对事物的认识和描述，能够按层次抽象出共性，可以有效复用，让类与类之间产生联系，是多态的前提

* A是(is)一种B，则A应该从B继承下来

Java表示法



基类/父类

打洞()

子类/派生类



打洞()
卖萌()



打洞()
做饭()



打洞()
耍猫()

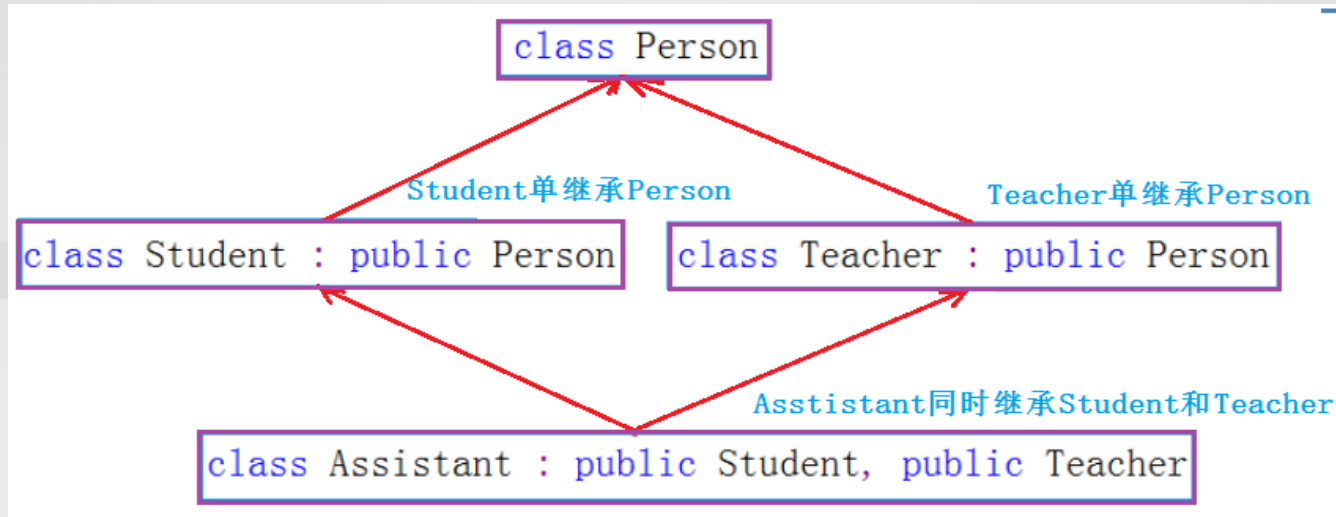
```
class 老鼠 {  
    private int 年龄;  
    protected Date 生日;  
    public String 名字;  
    public void 打洞() { System.out.println("老会打洞"); }  
};
```

```
class 米妮 extends 老鼠 {  
    public void 卖萌() { System.out.println("米妮会卖萌"); }  
}
```

```
class 料理鼠王 extends 老鼠 {  
    public void 做饭() { System.out.println("料理鼠会做饭"); }  
}
```

```
class 杰里 extends 老鼠 {  
    public void 耍猫() { System.out.println("Tom和Jerry玩"); }  
}
```

```
杰里 x = new 杰里();  
x.打洞();  
x.耍猫();  
x.年龄 = 30; // ? ? ? ? 这里可以吗?  
x.生日 = '1989-6-4'; // 这里可以用吗?  
x.名字 = '维尼熊';
```



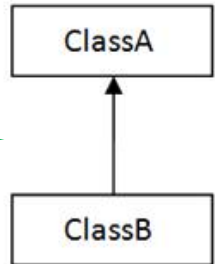
史上首例！第一個擁有三個父母親基因的嬰兒誕生

作者 Amy Lin | 发布日期 2016 年 09 月 29 日 6:05 | 分類 醫療科技 [Follow](#) [G+](#) [Like 1.3k](#) [Share](#)

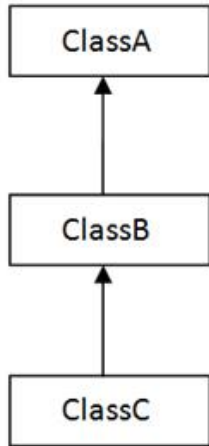


单继承：父类只有一个，单亲繁殖（蟑螂）
多继承：同时有多个父类，比如同时继承自父亲和母亲

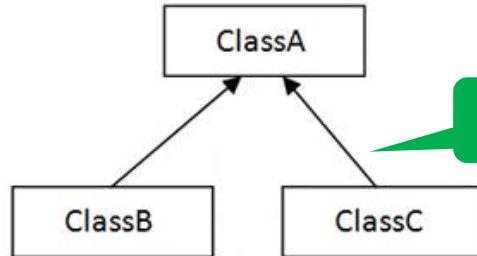
简单继承



1) Single

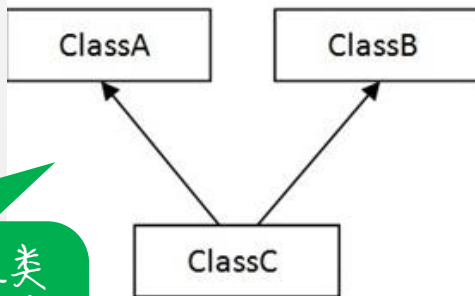


2) Multilevel

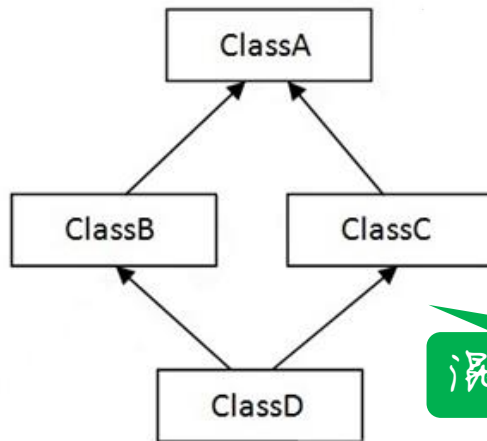


3) Hierarchical

多级继承

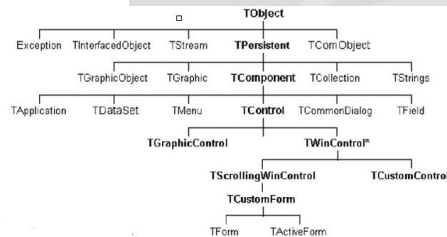


4) Multiple



5) Hybrid

层次继承



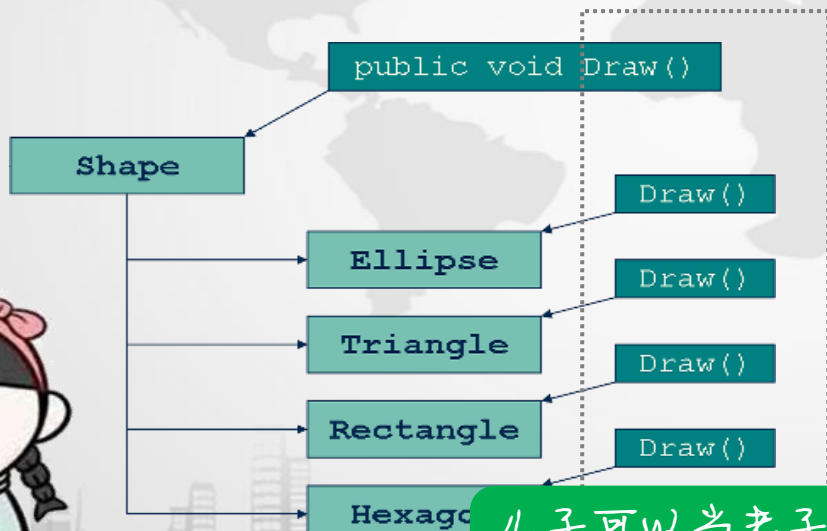
混合继承

多重继承，子类有多个父类
一般不建议多重继承，许多语言不支持多重继承，必须采用接口实现

面向对象主要特征：多态

对象根据所接收的消息会产生行动，同一消息为不同的对象接受时可产生完全不同的行动，这种现象称为多态性。多态提高了代码的扩展性和可维护性

* 龙生九子各有所好。前浪死在沙滩上，一浪更比一代浪。



关你屁事

儿子可以当老子用，
老子不能当儿子用

Java表示法

```
class Shape {
    public void Draw() { };
};
```

```
class Ellipse extend Shape {
    public Draw() { // 画椭圆的代码; }
}
```

```
class Triangle extend Shape {
    public Draw() { // 画三角形的代码; }
}
```

```
class Rectangle extend Shape {
    public Draw() { // 画长方形的代码; }
}
```

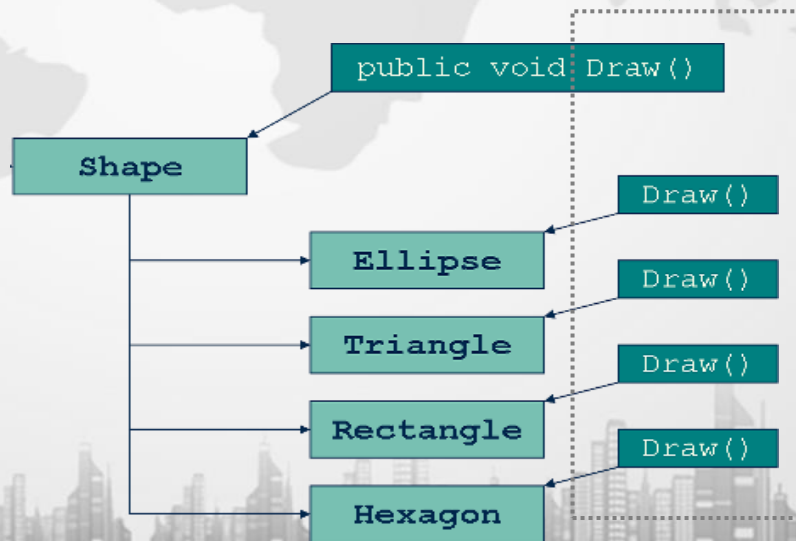
```
class Hexagon extend Shape {
    public Draw() { // 画六边形的代码; }
}
```

```
Shape x = new Ellipse();
// new Triangle()...
x.Draw();
```

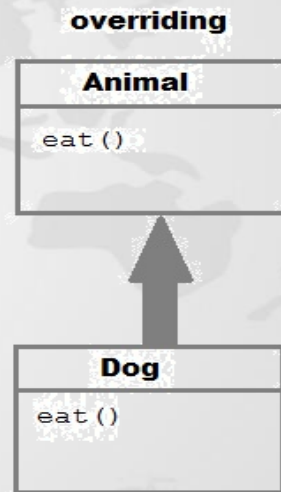
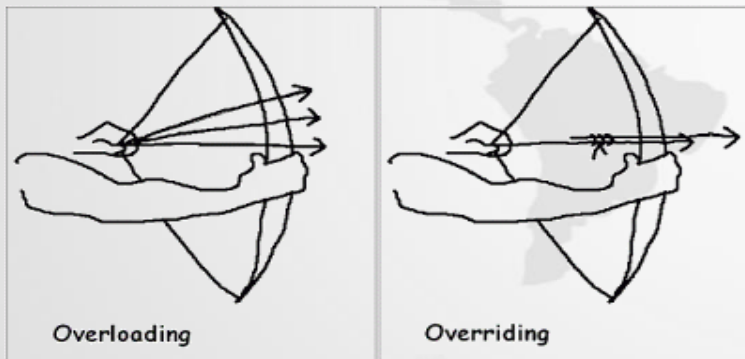
覆盖(Override)

父类和子类有同名并且参数形式相同的方法，子类对父类的允许访问的方法的实现过程进行重新编写，返回值和形参都不能改变。即外壳不变，核心重写！重写的好处在于子类可以根据需要，定义特定于自己的行为。

* 规定只是规定，执行要看个人。



Override 覆盖
PK
Overload 重载



虚方法= virtual

* 我规定……我先干，你随意……

当从父类中继承的时候，虚函数和被继承的函数具有相同的签名。但是在运行过程中，运行系统将根据对象的型别，自动地选择适当的实现运行。虚函数是面向对象编程实现多型的基本手段。

```
class Person {  
    public void Earn() {  
        System.out.println("人要赚钱");  
    }  
}
```

```
class Father extends Person {  
    public void Earn() {  
        System.out.println("老子赚钱靠上班");  
    }  
}
```

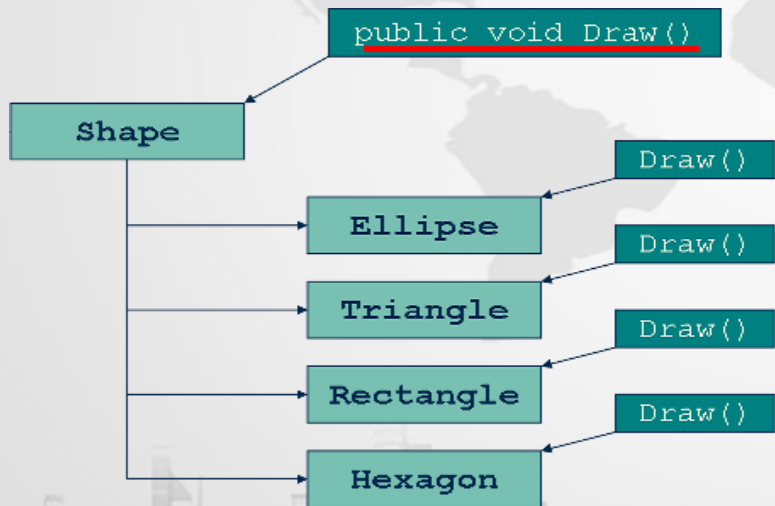
```
class Son extends Father {  
    public void Earn() {  
        System.out.println("儿子赚钱靠炒股");  
    }  
}
```

```
public class test {  
    public static void main(String[] args) {  
        Person son = new Son();  
        Person father = new Father();  
        father.Earn();  
        son.Earn();  
    }  
}
```


抽象方法= Abstract

只能由儿子实现，自己并不能实现。
有抽象类的方法不能实例化

* 我假设要……你必须要……



Java表示法

```
class Shape {
    abstract public void Draw() { };
};
```

```
class Ellipse extend Shape {
    public Draw() { // 画椭圆的代码; }
}
```

```
class Triangle extend Shape {
    public Draw() { // 画三角形的代码; }
}
```

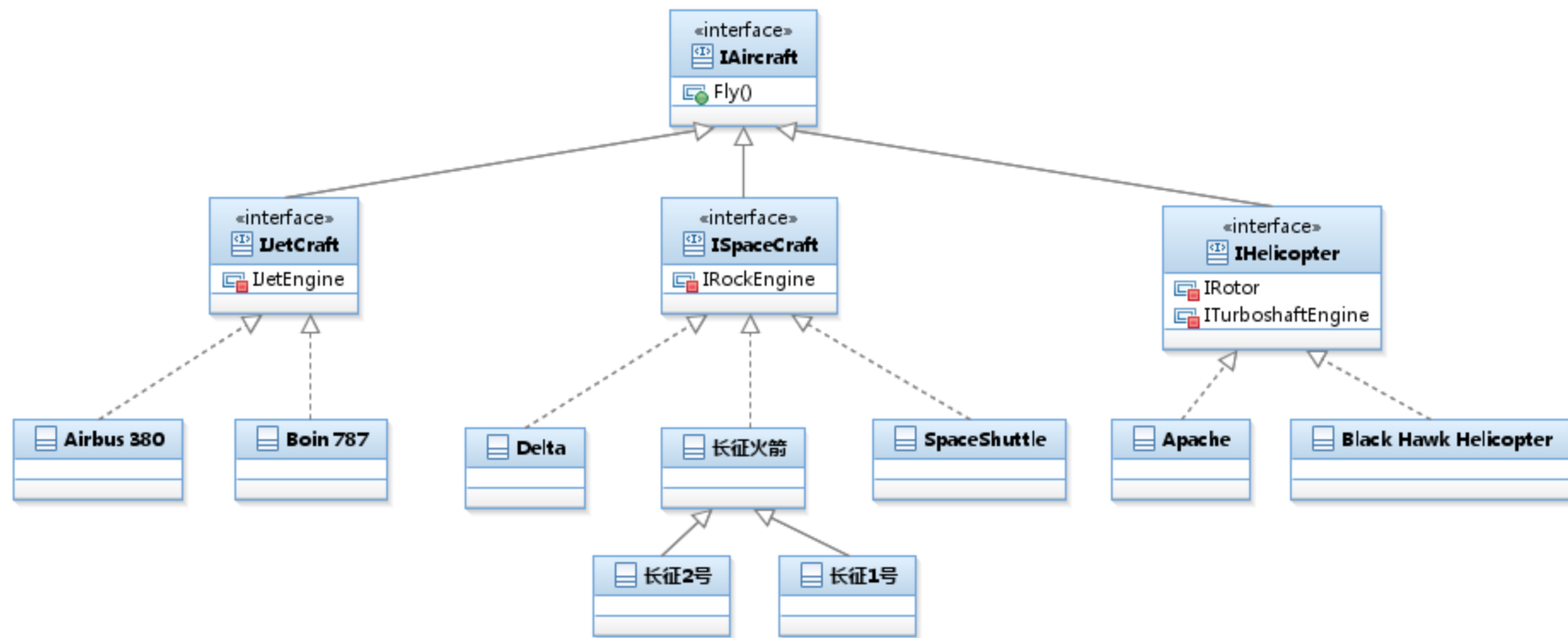
```
Shape x = new Ellipse();
// new Triangle()...
x.Draw();
```

```
Shape x = new Shape(); // ❌
```

接口: Interface

实现本接口的类或接口必须拥有的一组规则的集合。接口是纯抽象类，没有任何具体实现，主要用于体现一些纯概念的东西，比如飞行器是一个抽象的概念，它可以有飞行这个行为。

* 如果你是……则必须能做……





Animals

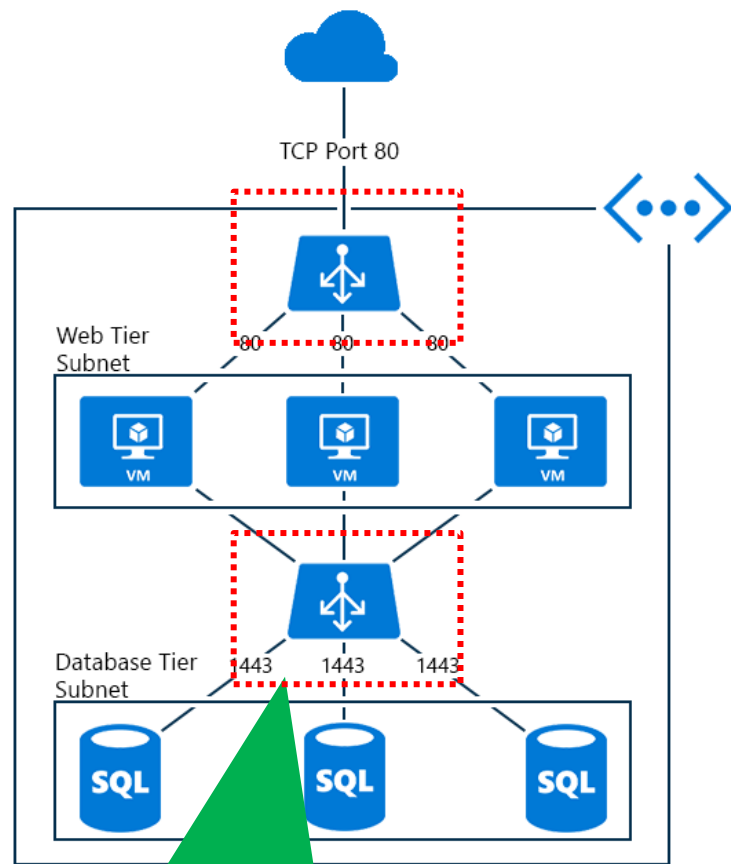
Call Speak();

www.javarefer.com

我想Talk(),
你却只能
Speak()

处理复杂问题的原则：抽象
人最核心的能力，人思维能力高低
抽象的好坏，关系到设计的好坏

我能怎么办
我也很绝望



计算机中的任何问题，都
可以通过添加一层来解决

练习：请抽象一下支付接口概念？

收款帐号
付款帐号
付款金额
付款密码

收款帐号
付款帐号
付款金额

鉴权();

左边灵活性不足，未封装敏感信息
无法扩充功能，只能密码支付，无法刷脸支付、指纹支付等



滚动 VS 移动

```
class Animal {  
    private String name;  
    public String getName() {  
        return this.name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public void Eat() {  
        System.out.println(this.name + "吃东西");  
    }  
}
```

```
class Dog extends Animal {  
    public Dog(){  
        setName("狗");  
    }  
}
```

```
public void Eat() {  
    System.out.println(getName() + "吃骨头");  
}  
}
```

```
class Cat extends Animal {  
    public Cat(){  
        setName("猫");  
    }  
}
```

```
public void Eat()  
{  
    super.Eat();  
    System.out.println(getName() + "吃鱼");  
}
```

```
public void Eat(String food) {  
    super.Eat();  
    System.out.println(getName() + food);  
}  
}
```

```
public class test {  
    public static void main(String[] args) {  
        Dog dog = new Dog();  
        dog.Eat();  
    }  
}
```

```
Cat cat = new Cat();  
cat.Eat();  
cat.Eat("骨头");
```

```
Animal animal1 = new Dog();  
animal1.Eat();
```

```
Animal animal2 = new Cat();  
animal2.Eat();  
}  
}
```

类Shape

Private

Color 属性
Height 属性
Width 属性
SetName();

Protected

Area 属性;
DoTransform(); virtual;
DoZoom();

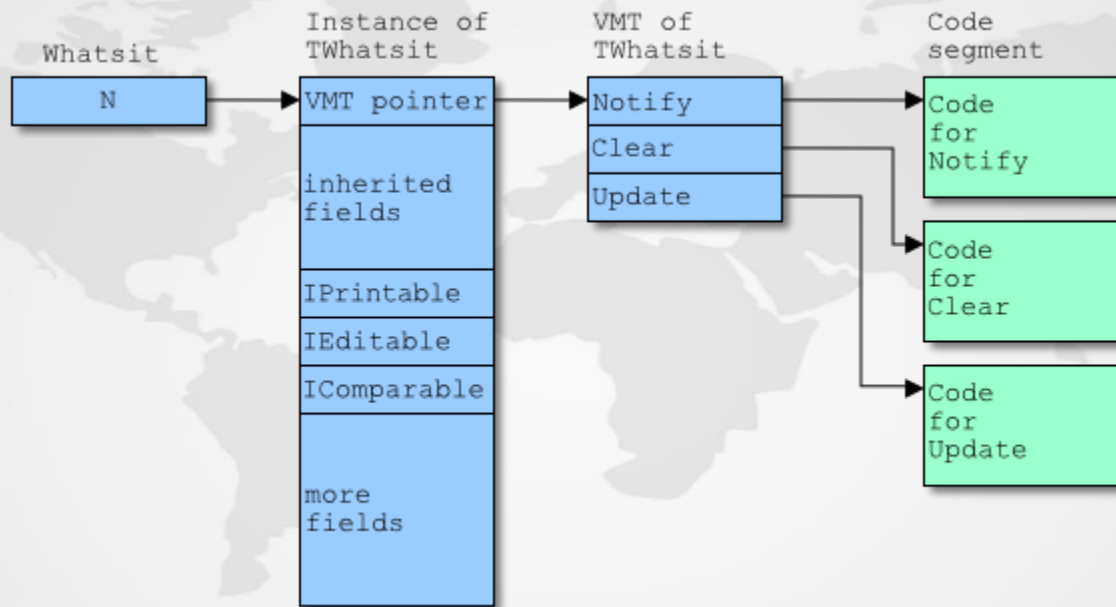
Public

Name属性
Move();
Draw(); virtual;

若类Rectangle继承自类Shape，那么请问：

1. 类Rectangle的对象可以访问Color、Height、Width属性吗？
2. 类Rectangle的对象可以访问Area属性吗？
3. 类Rectangle的对象可以访问SetName()方法吗？
4. 类Rectangle的对象可以访问DoTransform()、DoZoom()吗？
5. 类Rectangle的对象可以访问Transform()、Move()、GetArea()、Draw()吗？
6. 类Rectangle的对象可以访问Name属性吗？
7. 类Rectangle的对象可以override DoTransform()吗？
8. 类Rectangle的对象可以override DoZoom()吗？
9. 类Rectangle的对象可以override Move()吗？
10. 类Rectangle的对象可以override Draw()吗？

如何实现多态？



VMT, DMT

如何进行面向对象分析、设计、编程？

确定对象	确定系统中所有客观对象
	确定每个客观对象的属性
确定操作	确定每个对象的操作
确定对象间的联系	分析所有对象中的联系
	确定对象间传递的消息和模式(同步、异步)
分类对象	将具有相同特性的对象归在一起，确定“类”
确定结构	确定类间的继承关系
	确定类自建整体与局部、个体与集体的关系
内部实现	设计每个类的内部实现（数据结构和方法）
实例化和消息通信	创建所需的对象（类的实例），实现对象间应有的联系（发消息）

SOLID 设计原则

单一职责原则

- 一个类有且只有一个职责



开闭原则

- 对扩展开放，对修改关闭



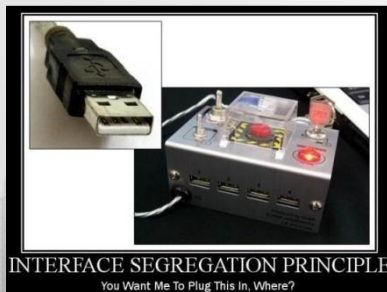
里氏替换原则

- 子类型必须能够替换它们的基类



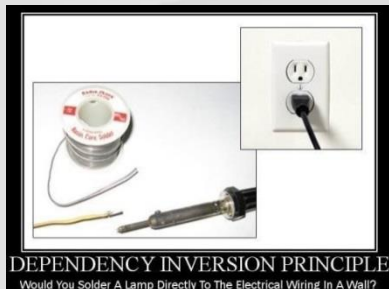
接口隔离原则

- 用户不应该被迫依赖他们不使用的接口



依赖倒置原则

- 高层次的模块不应该依赖于低层次的模块，而是，都应该依赖于抽象



隔离易变因素

符合人类日常思维，容易交流

增强了系统的应变能力

通过建模和继承，容易规模化

面向对象的优点

对象具有一致性

相对独立性

封装和继承让代码容易维护和修改

对软件复用提供了强有力的支持，快速编码



实训作业

请利用面向对象技术，分析、抽象以下内容：

现在客户要开发一个云平台的洗车管理系统为全国所有洗车店老板提供服务，每个洗车店老板可以购买一个账户来使用系统；每个洗车店老板可以管理自己的员工、顾客和车辆；把传统的洗车卡电子化，客户充值后有若干次洗车次数，顾客每次洗车后，扣除洗车卡的次数并短信通知；每个员工按洗车数量提成；

要求：

1. 列出系统中相关的对象
2. 抽象出系统所有的对象类及其属性，给出封装的不同等级
3. 给出系统中所有的类及其继承层次

请用Java编写代码，实现以下功能：

1. 抽象飞行器接口，包含飞行Fly()方法; 转载货物load()方法;设置名称setName();
2. 实现两个类，客机和火箭，从飞行器继承；
3. 客机可以设定载客人数；火箭可以设定运载货物重量；
4. 客机可以装载旅客，以喷气发动机在大气层内飞行；火箭可以转载卫星，以火箭发动机在外太空飞行；
5. 对火箭和飞机，每个类都分别实例化两个对象，例如"长征5号"载重20吨和"大力神"载重50吨；空客A380，载客530人；C919载客199人。
6. 所有功能，以System.out.println()打印文字即可，并以Java main类测试和输出。

所有代码，保存为一个java文件提交。

1. _____是对象的模版，即对一组具有相同数据和相同操作的对象的定义
2. 面向对象主要特征中，_____是子类自动共享父类数据和方法的机制，它由类的派生功能体现；同一消息为不同的对象接收时可产生完全不同的行动，这种现象称为_____。
3. 下列不属于面向对象方法的基本特征是_____。
A. 多态性 B. 封装性 C. 继承性 D. 抽象性
4. 软件系统开发过程中，了解问题所涉及的对象，对象间的关系和作用，然后构造问题的对象模型，这是利用面向对象技术来实现_____。
A. OOA B. OOD C. OOI D. OOP
5. 关于软件建模的用途，说法错误的是_____。
A. 软件建模可以帮助系统设计
B. 软件建模可以使具体的设计细节和需求分开
C. 通过软件建模可以利用模型全面把握复杂的系统
D. 软件件模可以直接生成最终的软件产品