# 父类和子类

## 多态

1.1 基本概念

(1) 多态指允许不同类的对象对同一个函数调用做出响应。即同一个方法可以根据调用对象的不同而调用不同的方法。

(2) 实现多态的技术称为动态绑定（dynamic binding），是指在程序执行期间判断调用对象的实际类型，从而调用不同的方法。比如是父类就调用父类的方法，是子类就调用子类的重写方法。

(3) 多态的作用是消除类型之间的耦合关系。

(4) 多态的三个必要条件：要有继承，要有重写，父类引用指向子类对象。

1.2多态的实现

(1) 实现接口中的方法

(2) 继承父类进行方法重写

(3) 同一个类中进行方法重载。

## 抽象类

* 1. 简介

1. 关键字abstract用于修饰类和方法，分别称为抽象类和抽象方法。

public abstract class chou{} //抽象类

public abstract void chouxiang(int i){} //抽象方法

1. 抽象类中不一定要有抽象方法，但是含有抽象方法的类一定是抽象类。
2. **抽象方法必须为 public 或者 protected（因为如果为 private，则不能被子类继承，子类便无法实现该方法），缺省情况下默认为 public。**
3. 抽象类不能用来创建对象；
4. 如果一个类继承于一个抽象类，则子类必须实现父类的抽象方法。如果子类没有实现父类的抽象方法，则必须将子类也定义为为 abstract 类。

# Object类

## Object类的常用方法

1. clone() ：创建并返回此对象的副本。
2. equals(Object obj)：是否等于其他对象。
3. finalize()：Java 允许使用 finalize() 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时被调用。finalize() 方法是在垃圾收集器删除对象之前对这个对象调用。
4. getClass()：反射用法，返回一个该对象的对象。
5. hashCode()：返回该对象的哈希值。
6. toString()：返回对象的字符串形式。
7. wait()：使当前进程进入等待状态，把锁交出来，直到有另外的线程调用notify()
8. notify()：唤醒正在等待的某一个线程
9. notifyAll()：唤醒全部在等待状态的线程。

# 类的反射

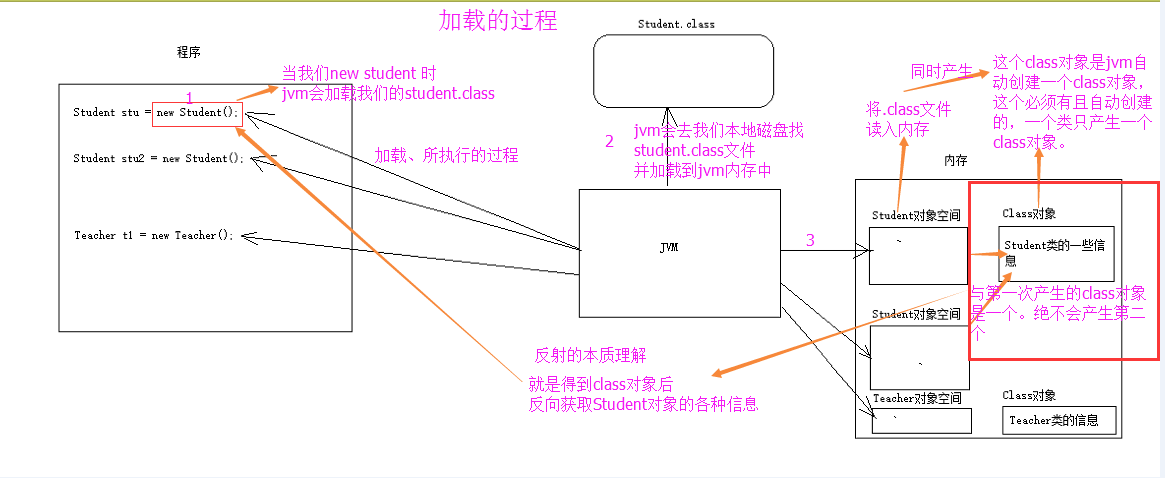
## 简介

1. AVA 反射机制是在运行状态中，**对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；**这种动态获取的信息以及动态调用对象的方法的功能称为 java 语言的反射机制。
2. 要想解剖一个类, 必须先要获取到该类的字节码文件对象。而解剖使用的就是 Class 类中的方法. 所以先要获取到每一个字节码文件对应的 Class 类型的对象.
3. **反射就是把 java 类中的各种成分映射成一个个的 Java 对象**

例如：一个类有：成员变量、方法、构造方法、包等等信息，利用反射技术可以对一个类进行解剖，把个个组成部分映射成一个个对象。

## Class对象

1. Class 对象的由来是将 class 文件读入内存，并为之创建一个 Class 对象。一个类只能创建一个Class对象。



1. Class 类的实例表示正在运行的 Java 应用程序中的类和接口。也就是 jvm 中有 N 多的实例每个类都有该 Class 对象。（包括基本数据类型）
2. Class 没有公共构造方法。**Class 对象是在加载类时由 Java 虚拟机以及通过调用类加载器中的defineClass 方法自动构造的。**也就是这不需要我们自己去处理创建，JVM 已经帮我们创建好了。

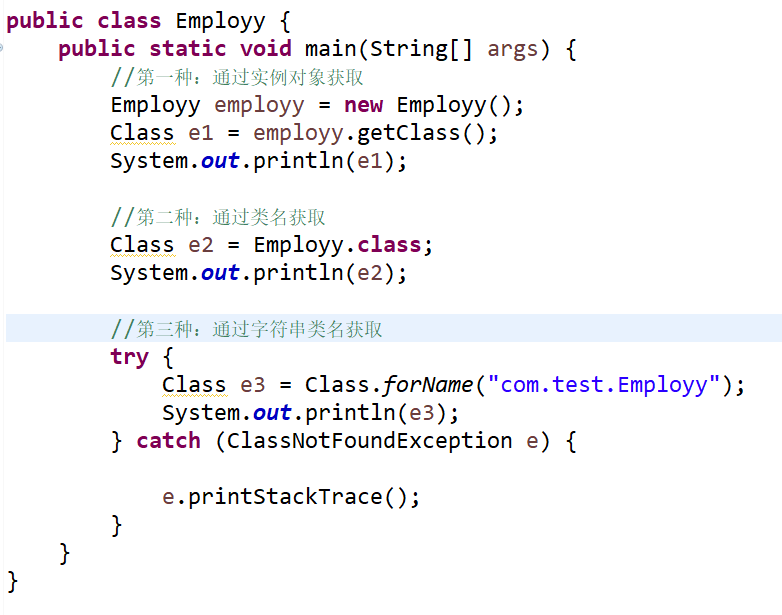
## 反射的应用

* 1. 获取Class对象的三种方式

(1) Object自带的方法 getClass();

(2) 任何数据类型（包括基本数据类型）都有一个 “静态” 的 class 属性

(3) 通过 Class 类的静态方法：forName（String className）(常用)



* 1. 注意事项

1. 在运行期间，一个类，只有一个 Class 对象产生。
2. 三种方式常用第三种，第一种对象都有了还要反射干什么。第二种需要导入类的包，依赖太强，不导包就抛编译错误。一般都第三种，一个字符串可以传入也可写在配置文件中等多种方法。
3. **一般获取反射是为了拿到实例对象的信息！**

## 获取构造方法

通过Class对象可以获取某个类中的：构造方法、成员变量、成员方法；并访问成员；

1. 批量获取的：

public Constructor[] getConstructors()：所有"公有的"构造方法

public Constructor[] getDeclaredConstructors()：

获取所有的构造方法(包括私有、受保护、默认、公有)

1. 获取单个构造方法：

Public Constructor getConstructor(构造方法参数类型)

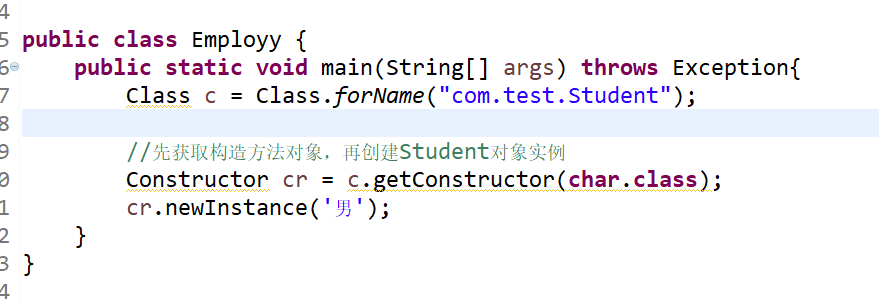
获取单个的"公有的"构造方法。比如无参是null，字符是char.class，整数是int.class

public Constructor getDeclaredConstructor(构造方法参数类型):

获取"某个构造方法"可以是私有的，或受保护、默认、公有；

1. 调用构造方法：（先获取构造方法对象，再调用newInstance方法）

Constructor-->newInstance(Object... initargs)



## 获取成员变量

(1)`批量获取

Field[] getFields():获取所有的"公有字段"

Field[] getDeclaredFields():获取所有字段，包括：私有、受保护、默认、公有；

(2) 获取单个的：

public Field getField(String fieldName):获取某个"公有的"字段；

public Field getDeclaredField(String fieldName):获取某个字段(可以是私有的)

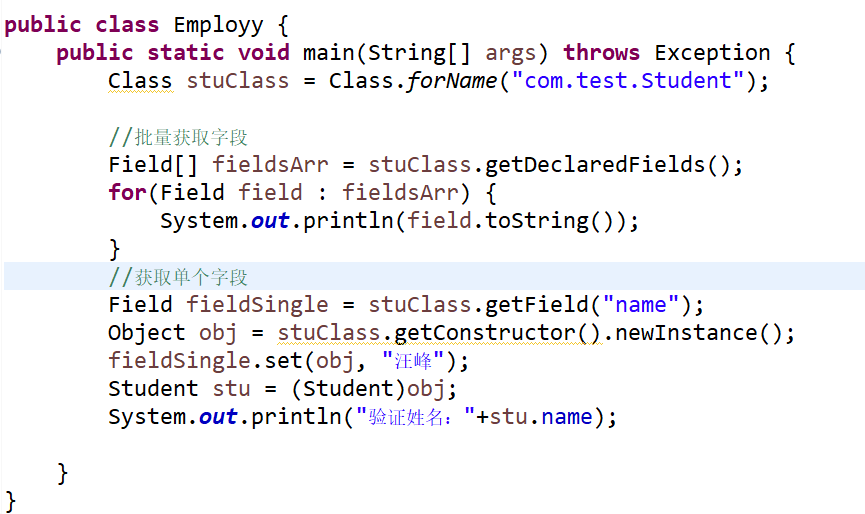
(3) 设置字段的值：

Field --> public void set(Object obj,Object value):

参数说明：

obj:要设置的字段所在的实例对象；

value:要为字段设置的值；



## 获取成员方法

1. 批量获取方法

public Method[] getMethods()

获取所有"公有方法"；（包含了父类的方法也包含Object类）

public Method[] getDeclaredMethods()

获取所有的成员方法，包括私有的(不包括继承的)

1. 获取单个方法**（如果是多参数，用逗号隔开）**

public Method getMethod(String name,Class<?>... parameterTypes):

参数： name : 方法名； Class ... : **形参的Class类型对象**

public Method getDeclaredMethod(String name,Class<?>... parameterTypes)

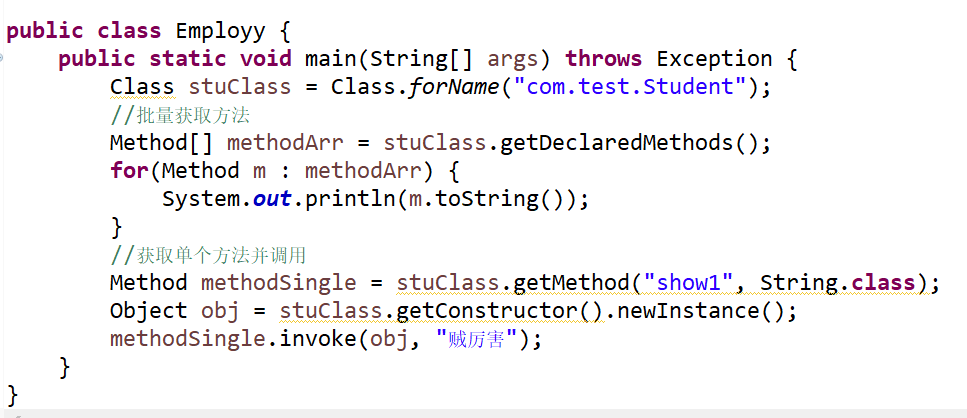
1. 调用方法：

Method --> public Object invoke(Object obj,Object... args):

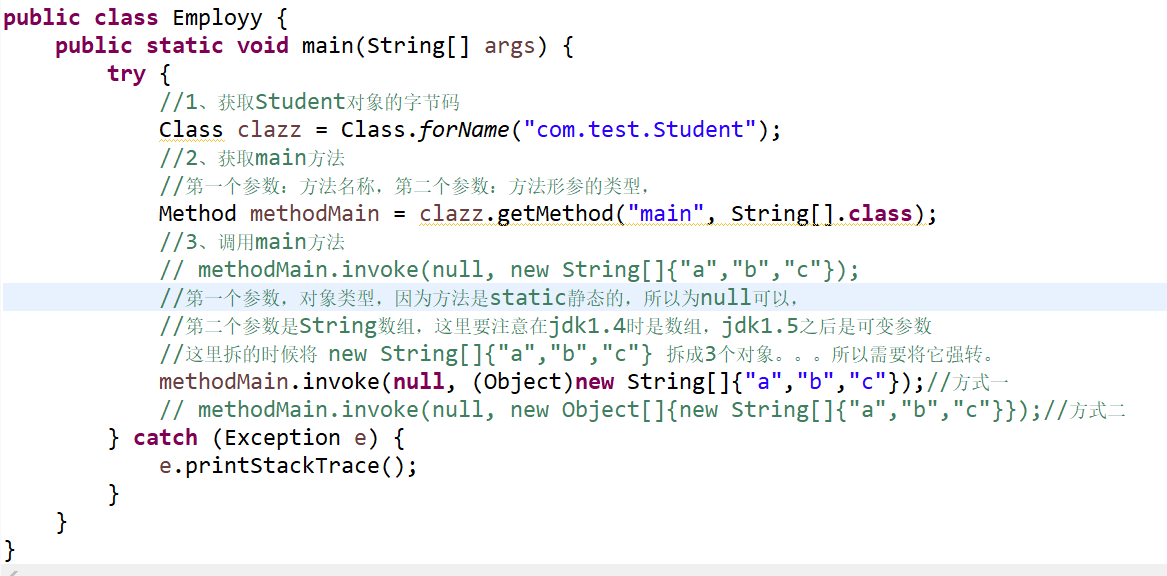
参数说明：

obj : 要调用方法的对象；

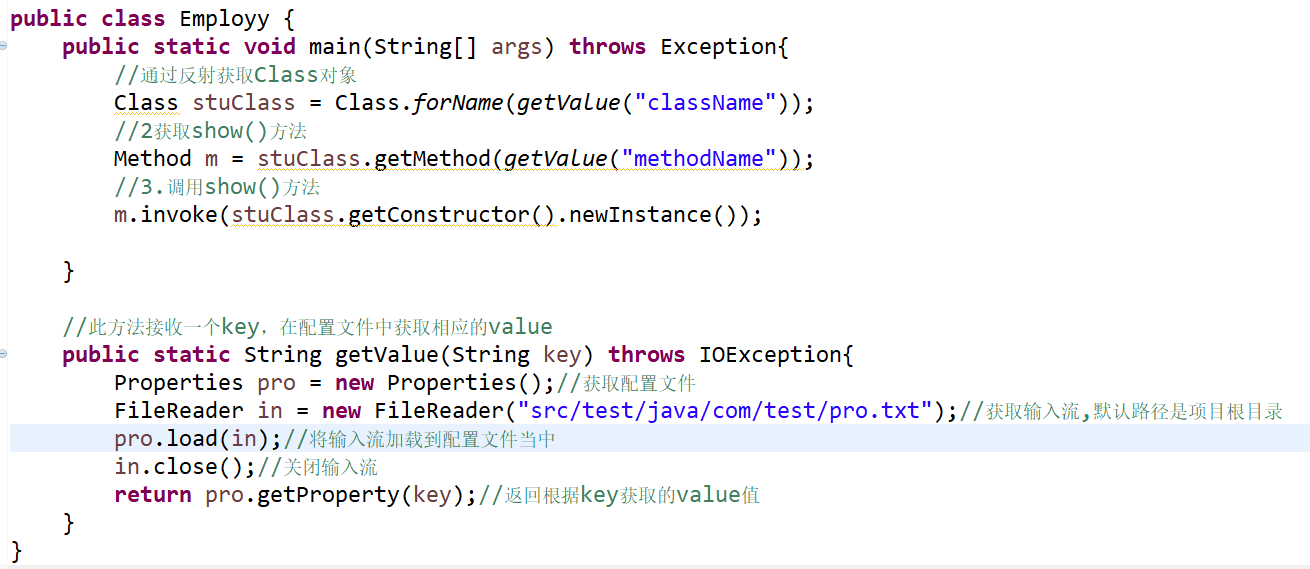
args:调用方式时所传递的实参；

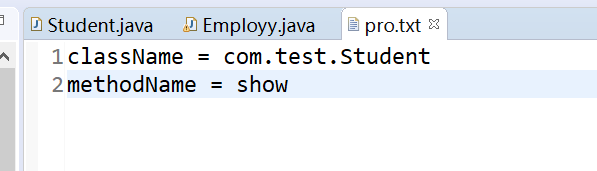


## 反射main方法



## 通过反射读取配置文件





1. FielReader(“”)的默认路径是项目的根目录，要自己更改路径。
2. 将类名和方法名放在配置文件中，更改类名的时候只需要更改配置文件而不用更改代码。

## 反射越过泛型检查

