* **多态性，一个事务的多种表现形态，在java中有两种体现：**

（1）方法的**重载（overload）**和**重写（overwrite）**。

（2）**对象的多态性**——可以直接应用在抽象类和接口上。

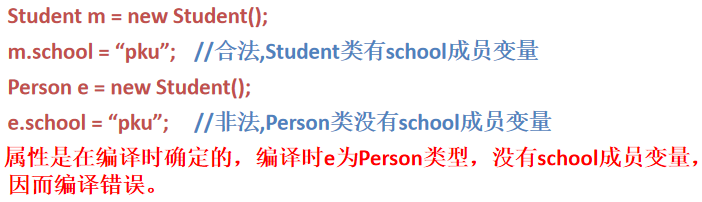
* **Java引用变量有两个类型：编译时类型和运行时类型**
* 编译时类型由**声明该变量时使用的类型（父类类型）**决定，运行时类型由**实际赋给该变量的对象（子类的对象）**决定。
* **编译时类型和运行时类型不一致**，就出现多态（Polymorphism）。
* **对象的多态性——在Java中，子类的对象可以替代父类的对象使用**
* 一个变量只能有一种确定的数据类型
* 一个引用类型变量可能指向（引用）多种不同类型的对象

Person p = new Student();

Object o = new Person(); //Object类型的变量o，指向Person类型的对象

o = new Student(); //Object类型的变量o，指向Student类型的对象

* 子类可看做是特殊的父类，所以父类类型的引用可以指向子类的对象：向上转型（upcasting）。
* **一个引用类型变量如果声明为父类的类型，但实际引用的却是子类对象，那么该变量就不能再访问子类中添加的属性和方法**



* **虚拟方法调用（Virtual Method Invocation）——主要应用**

**通过父类的引用指向子类的对象实体，当调用方法时，实际执行的是子类重写父类的方法。**

* 正常的方法调用

Person e = new Person();

e.getInfo();

Student e = new Student();

e.getInfo();

* 虚拟方法调用（多态情况下）

Person e = new Student();

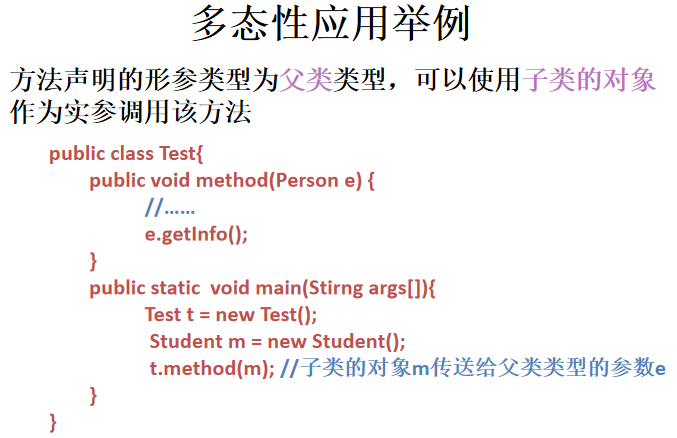
e.getInfo(); //调用Student类的getInfo()方法

* 编译时类型和运行时类型

编译时e为Person类型，而方法的调用是在运行时确定的，所以调用的是Student类的getInfo()方法。——**动态绑定**

// 向上转型  
Person p2 = new Woman();  
p2.eat();  
p2.walk();  
  
// ((Woman) p2).shopping(); 向下转型  
Woman w = (Woman)p2;  
w.shopping();

* **多态小结**
* 前提
* 要有类的继承
* 要有子类对父类方法的重写
* 成员方法：
* 编译时：要查看**引用变量所属的类**中是否有所调用的方法。
* 运行时：调用实际**对象所属的类**中的重写方法。
* 成员变量：
* **不具备多态性**，只看引用变量所属的类。



* **instanceof操作符（判断对象x是否是类A的一个实例，是返回true）**
* 要求x所属的类与类A必须是子类和父类的关系，否则编译错误。
* 如果x属于类A的子类B，x instanceof A值也为true。

public class Person extends Object {…}

public class Student extends Person {…}

public class Graduate extends Person {…}

--------------------------------------------------------

public void method1(Person e) {

if (e instanceof Person)

//处理Person类及其子类对象

if (e instanceof Student)

//处理Student类及其子类对象

if (e instanceof Graduate)

//处理Graduate类及其子类对象

}