链滴

# 10 多态

作者：Gao-Eason
原文链接：https://ld246.com/article/1649668756564
来源网站：链滴

**10 多态**

**定义**

**● Java 引用变量有两个类型：编译时类型：由声明**
**变量时使用的类型决定运行时类型：由实际赋给该变量的对象**
**定**

**● 当编译时类型和运行时类型不一致，就可能出现所谓的多态（Polymorphism）**

```
SuperClass obj = new SubClass();Interface obj = new ImplementingClasses();
```

**● 子类对象赋给父类变量 或 实现类对象赋给接口变量，该对象可以有多种形态，在运行时期**
**表现出 子类 或 实现类 特征（调用 子类 或 实现类 的覆盖方法）**

**前提**

**● 继承（类和类）或实现（接口和实现类）**

```
class Animal {}class Dog extends Animal {}class Cat extends Animal {}Animal a = null;a = new Dog();a = new Cat();
```

**作用**

**● 屏蔽不同子类或实现类之间的实现差异，从而可以做到通用编程**

**注意**

**● 只有实例方法才有多态的效果（调用子类的实例方法**

*），类方法和字段都没有（访问**父类**字段或调用时**父类**类方法）**

**● 编译时会检查父类（编译时类型）**
**否存在该实例方法** **，若不存在会编译报错；**运行时**先从**子类（运行时类型）找该**
**例方法**，找到就执行，没找到就去**父类**找**

**● 当对象有多种形态，需要调用只存在子类的实例方法时，可以**
**该引用变量进行强制类型转换**

public class SuperClass { String name = "父类的实例变量"; public void superMethod() { System.out.println("父类中普通的实例方法"); } protected void test() { System.out.println "父类中被覆盖的实例方法"); }}public class SubClass extends SuperClass { String name = "子类的实例变量"; public void subMethod() { System.out.println("子类中普通的实例方法"); } @Override public void test() { System.out.println("子类中覆盖父类的实例方法"); } public static void main(String[] args) { SuperClass p = new SubClass(); System.out println(p.name); // 父类的实例变量 System.out.println(((SubClass)p).name); // 子类的实例量 p.superMethod(); // 父类中普通的实例方法 p.test(); // 子类中覆盖父类的实例方法 / p.subMethod(); // 编译报错，SuperClass 中没有 subMethod() ((SubClass)p).subMethod(); // 子类中普通的实例方法 }}

**引用变量的强制类型转换**

**● 引用类型之间的转换只能在具有继承关系的两个类型之间进行，否则编**
**报错**

**● 只能将一个引用变量的类型强制转换成该变量实际引用的对象可以被定义成的类型**
*，否则会引发 ClassCastException 异常**

**● 语法格式：(Type)object**

// 向上转型：把子类对象赋给父类引用变量（多态）Animal a = new Dog();Object obj = new Dog );// 强制类型转换：把父类对象赋给子类引用变量Dog d = (Dog) a;Cat c1 = (Cat) a; // ClassCastEx eption: Dog cannot be cast to CatCat c2 = (Cat) new Animal(); // ClassCastException// 在进行 制类型转换之前，先用 instanceof 运算符判断是否可以成功转换，从而避免出现 ClassCastException 异常if (a instanceof Dog) { Dog d = (Dog) a;}

**instanceof 运算符**

**● 判断该对象是否是某一个类 / 子类 / 实现类的实例，如果是，返回 true**

**● instanceof 运算符前面操作数的编译时类型要么与后面的类**
**同****，要么与后面的类**具有继承关系**，否则会引起编译错误****

复制代码Object obj = "ABC";obj instanceof String; // tureobj instanceof Object; // tureobj ins anceof Math; // falseString str = "ABC";str instanceof Math; // String 类与 Math 类没有继承关 ，编译报错// 获取当前对象的运行时类型obj.getClass(); // class java.lang.Stringobj.getClass() == tring.class; // tureobj.getClass() == Object.class; // flase