java 中的继承

作者：rocflight
原文链接：https://ld246.com/article/1556264854392
来源网站：链滴

什么是继承?

在java中，难免会出现有很多类具有相同的数据和行为，它们甚至在抽象层面可以归为同一个种类。然具有相同的数据和行为，那么我们在定义这些类的时候也会重复性的去声明这些数据和行为。**为了便，我们可以把这些相同的数据和行为抽出来放在单独的一个类里，我们称这个类为父类，任何想要有父类据和行为的类可以通过继承来得到，通过继承得到数据和行为的类，我们称这些类为子类。**

定义父类和子类

定义父类非常简单，和定义一个普通的类一样，只是这个父类封装了会重复性的数据和行为。

定义一个子类也非常简单，使用extends关键字即可。

```
/**
 *   子类名称 extends  父类名称
 *    狗类继承了动物类，狗也是动物
 */
 public class Dog extends Animal{

 }
```

方法覆盖

方法覆盖也叫方法重写。虽然父类封装了相同的行为，但子类们有时候面对某一种相同的行为表现的各不相同。

比如猫和狗都继承了动物，动物都具有叫的行为，但猫叫和狗叫就不相同。所以我们就可以说猫和狗写了叫的行为。

```
public class Animal {
    /**
    *     叫的行为
    */
    public void call(){
        //父类是抽象的，我们不知道“动物”怎么叫，就给它一个空的实现
        System.out.println("");
    }
}

public class Dog extends Animal {
    /**
     * 覆盖（重写）叫的方法
     */
    @Override
    public void call() {
        System.out.println("汪汪汪");
    }
}

public class Cat extends Animal {
    /**
     * 覆盖（重写）叫的方法
     */
```

```
    @Override
    public void call() {
        System.out.println("喵喵喵");
    }
}
```

子类构造器

构造器也称构造方法，是用来构造对象的，new关键字其实就是调用构造器。

构造器被调用的时候还要完成对成员变量的赋值，如果是无参构造器，则根据每个成员变量的类型赋默认值。

由于子类不能访问父类的私有成员，所以子类构造方法在执行的时候必须先调用父类构造器，优先完对父类私有成员的赋值。 不然，那些复用的数据没有值，那也就没什么意义了。

实例化一个对象的过程如下：

1.首先根据new关键字来调用指定的构造器来创建对象。

2.如果是无参构造器，首先先调用父类的无参构造器，再调用子类的无参构造器

3.如果是有参构造，首先先调用父类的有参构造器，再调用子类的有参构造器

多态

字义就很容易理解，多种形态，动物可以是狗，可以是猫，也可以是其他任何动物

那么，我们是不是可以这么来写代码呢。

```
Animal animal = new Dog();
```

```
Animal animal = new Cat();
```

这就是多态，子类的对象可以使用父类的类型来接收，反之则不可以，我们可以说狗是一只动物，却可以说动物是一只狗，这样显得就很奇怪。

优点

多态可以使我们动态决定自己想要什么，想要狗就是狗，想要猫就是猫。由此可知多态十分灵活，特容易扩展和复用。

```
Animal animal = new Dog();
```

```
animal = new Cat();
```

缺点

```
Animal animal = new Dog();
```

此时，animal的类型虽然是Dog，但animal归根结底是父类的引用指向了子类，所以animal不能访问og的成员和方法。

其实，也算不上什么缺点。

阻止继承

有了继承和多态带来的好处，那么我们是不是可以去随便继承，随便复用代码。

那显然是不行的，java的继承是单继承，这本身就告诉我们继承要有意义的去继承，不能看见谁都要

“祖宗”，这样会造成一个类混乱和复杂，职责不明确，导致我们不知道这个类到底要干什么。

也可以使用final关键字来限制别的类去继承和重写某些重要的方法，导致方法本身的含义被破坏。

final关键字就保证了类不允许被继承

抽象类

还记得方法重写的时候，有这么一段代码

```
public class Animal {
    /**
    *     叫的行为
    */
    public void call(){
        //父类是抽象的，我们不知道“动物”怎么叫，就给它一个空的实现
    }
}
```

父类都是抽象出来的，继承链中越往上越抽象，往往抽象的父类并不需要有具体的实例对象，只是为达到子类能复用的目的，在面对可以有多种表现的行为方法，父类也并不知道怎么去实现，如上面代所示，即便实现也只是一个空的方法，面对这种情况，抽象类就诞生了。

抽象类的定义

使用abstract关键字来定义一个抽象类

```
public abstract class Animal{
    public abstract void call();
}
```

在有抽象方法的时候，必须声明类为抽象类，且抽象类不能被实例化，抽象方法必须被重写。

Object类

Object类是java中所有类的始祖，是所有类的父类，每个类都是由它扩展而来。

虽然每个类并没有使用 extends Obejct 去继承Object，**但如果这个类没有明确指出其父类，那这个就默认继承了Object。**

仅为本人复习笔记，复习自 java 核心技术卷 1