

41从零开始学Java41之向上转型与向下转型

前言

配套开源项目资料

一. 类型转型

二. 向上转型

1. 概念

2. 特点

3. 语法

4. 案例

三. 向下转型

1. 概念

2. 特点

3. 语法

4. 案例

四. 结语

五. 今日作业

1. 第一题

作者：孙玉昌，昵称【**一一哥**】，另外【**壹壹哥**】也是我哦

千锋教育高级教研员、CSDN博客专家、万粉博主、阿里云专家博主、掘金优质作者


前言

面向对象的第三个特征是多态，实现多态有三个必要条件：继承、方法重写和向上转型。但是我们现在还不知道什么是向上转型，所以在学习多态之前，我们还要先学习Java的类型转换。本篇文章，**壹哥**就来带大家认识什么是类型转换，看看类型转换都有哪几种情况，以及如何避免类型转换时出现异常。

全文大约【2600】字，不说废话，只讲可以让你学到技术、明白原理的纯干货！本文带有丰富的案例及配图视频，让你更好地理解 and 运用文中的技术概念，并可以给你带来具有足够启迪的思考.....

配套开源项目资料

Github:



GitHub – SunLtd/LearnJava

Contribute to SunLtd/LearnJava development by creating an account on GitHub.

GitHub

Gitee:



一一哥/从零开始学Java

从零开始学Java系列 稀土掘金专栏地址: <https://juejin.cn/column/7175082165548351546> CSDN专...

Gitee

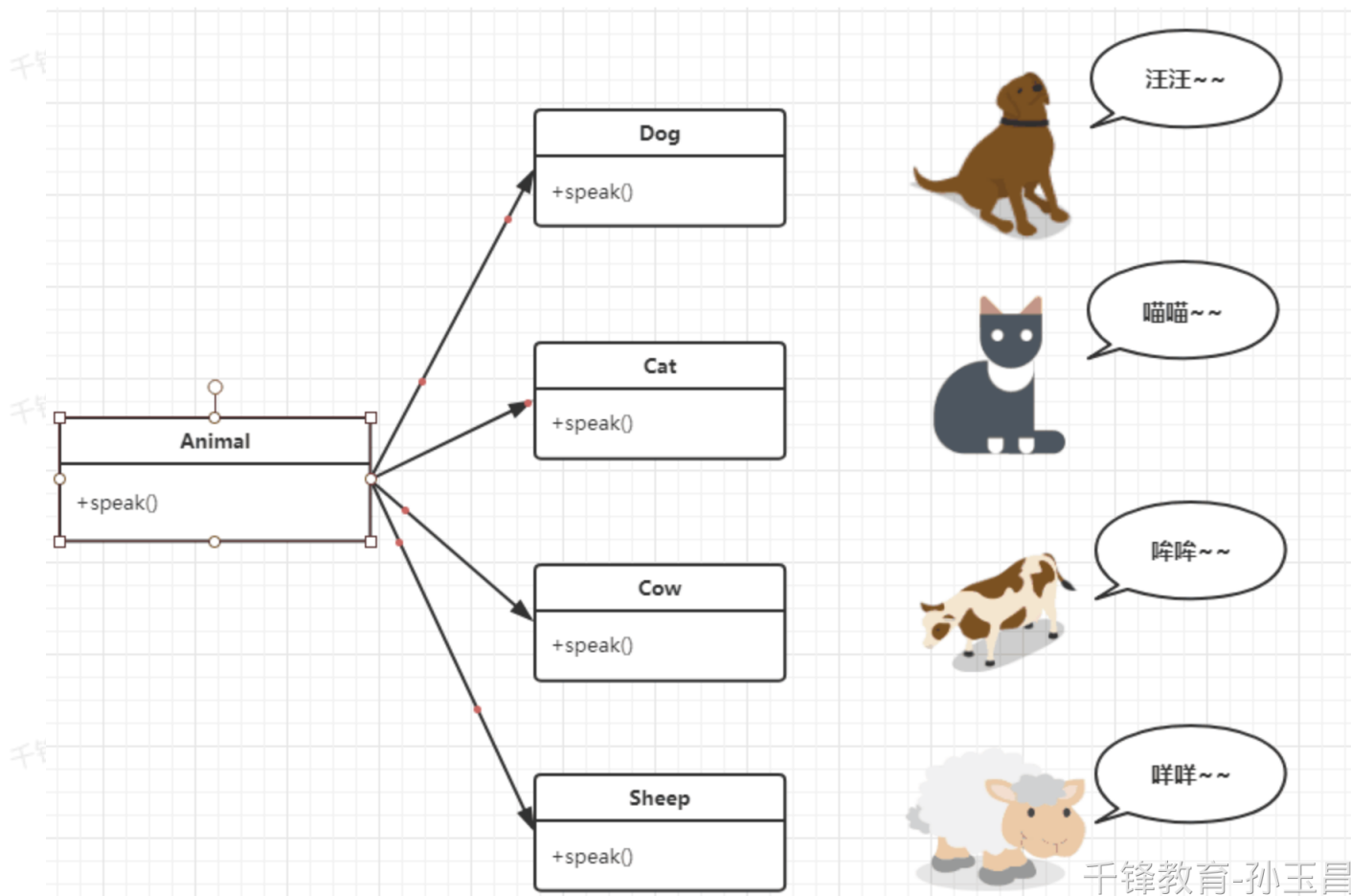
一. 类型转型

将一个类型转换成另一个类型的过程被称为类型转换。我们所说的**对象类型转换**，一般是指两个存在继承关系的对象，而不是任意类型的对象。如果两个类型之间没有继承关系，就不允许进行类型转换，否则会抛出强制类型转换异常(`java.lang.ClassCastException`)。

Java语言允许某个类型的引用变量引用子类的实例，而且可以对这个引用变量进行类型转换。

Java中引用类型之间的类型转换(前提是两个类是直接或间接的父子关系)主要有两种，分别是**向上转型(upcasting)**和**向下转型(downcasting)**。

我们先来看下面这张图：



猫、狗、牛、羊都是动物，所以“动物”是父类，猫狗牛羊是具体的子类。我们可以说猫是动物，狗是动物，牛是动物，羊是动物，这个结论肯定没错！但如果我们说，动物是猫，动物是狗，动物是牛，动物是羊，这个结论一定正确吗？那可就不一定了。**子类肯定符合父类类型，但反之，父类不一定符合子类类型！**

接下来壹哥分别对向上转型和向下转型进行讲解。

二. 向上转型

1. 概念

所谓的向上转型，就是父类引用指向子类的对象，也就是把子类对象直接赋给父类引用变量，此时不用强制转换。比如我们说猫是动物，狗是动物，牛是动物，羊是动物，这就是把子类当成父类来用。父类是子类的上级，我们直接把子类向上提拔转型了。

2. 特点

向上转型具有如下特点：

- 编译类型取决于=号左边，运行类型取决于=号右边；
- 子类可以调用父类的所有成员，但需遵守访问权限；
- 父类不能调用子类的特有成员；
- 最终的运行效果取决于子类的具体实现。

3. 语法

向上转型的基本语法如下所示：

Java | 复制代码

```
1 //左侧是父类引用变量，右侧是创建的子类对象，我们可以把它当作父类使用
2
3 父类类型 引用名 = new 子类类型();
```

4. 案例

接下来我们通过一个案例来演示向上转型该如何实现。

```
1  /**
2   * @author 一一哥Sun
3   * QQ: 2312119590
4   * CSDN、掘金、知乎找我哦
5   *
6   * 定义父类---向上与向下转型
7   */
8  public interface Animal {
9
10     //叫
11     public void speak();
12 }
13
14 //Dog类实现Animal接口
15 public class Dog implements Animal {
16
17     @Override
18     public void speak() {
19         System.out.println("狗子: 汪汪");
20     }
21 }
22
23 //Cat类实现Animal接口
24 public class Cat implements Animal {
25
26     @Override
27     public void speak() {
28         System.out.println("猫子: 喵喵");
29     }
30 }
31
32 }
33
34 //定义一个测试类
35 public class AnimalTest {
36
37     public static void main(String[] args) {
38         //向上转型
39         //父类引用dog变量, 指向子类对象new Dog();
40         //动物 = 狗; 以下代码的意思就是"狗是动物"
41         Animal dog=new Dog();
42         //向上转型后, 可以使用父类Animal中的属性和方法。
43         dog.speak();
44
45         //Animal animal=new Cat();效果等同于如下两行代码:
```

```
46         Cat cat=new Cat();
47         Animal animal=cat;
48         animal.speak();
49     }
50 }
51 }
```

我们在进行向上转型时，可以写成 `Animal animal=new Cat(); Animal animal=cat;` 的形式。就是先创建出cat对象，然后再将cat对象赋值给Animal类型的变量，这样就实现了向上转型。因为Animal是父级类型，Cat是子级类型，Cat是Animal的子类，所以可以直接将cat变量赋值给animal，这就是向上转型。但这种写法比较繁琐，我们通常是直接采用 `Animal cat=new Cat();` 的形式，简洁明了。

我们之所以可以实现向上转型，主要是因为两个类型之间是父子关系。比如在本例中，向上转型就等于说“猫是动物，狗是动物”，因为猫狗都是动物的子类，所以这个结论是确定无误的。

三. 向下转型

1. 概念

向下转型则反之，**也就是一个已经向上转型的子类对象指向父类引用**。向下转型后，可以调用子类类型中所有的成员。向下转型时，必须进行强制类型转换。因为**父类拥有的成员，子类肯定会有，但子类拥有的成员，父类不一定有**。

但要注意，如果父类的引用对象指向的是子类对象，则向下转型时是安全的，即编译时不会出现错误。但如果父类的引用对象是父类本身，那么在向下转型的过程中是不安全的，虽然编译时不会出错，但在运行过程中会出现强制类型转换异常。我们可以使用instanceof运算符来避免出现强制类型转换异常。

2. 特点

向下转型具有如下特点：

- 只能强制转换父类的引用，不能强制转换父类的对象；
- 父类的引用必须指向子类目标类型的对象；
- 向下转型后，父类可以调用子类类型中的所有成员。

3. 语法

向下转型的基本语法如下所示：

▼ Java | 复制代码

```
1 //向下转型使用强制类型转换的格式，将父类引用类型转换为子类引用类型
2
3 子类类型 引用名 = (子类类型) 父类引用；
4
```

4. 案例

接下来我们在上面的案例基础之上，对代码进行改造，在Cat类中增加一个新的方法。

```
1 public class Cat implements Animal {
2
3     @Override
4     public void speak() {
5         System.out.println("猫子: 喵喵");
6     }
7
8     //给其他动物打招呼
9     public void helloAnimal(Animal animal) {
10         //向下转型, 将父类型转为子类型。
11         //此时有可能会出ClassCastException类型转换异常, 因为子类一定属于父类的一
        员, 但父类不一定属于子类。
12         //我们说“猫是动物”一定没问题, 但如果说“动物是猫”, 这个结论未必正确。所以把“动
        物强转成猫”的过程中, 有可能会出异常。
13         //只有两者之间具有明确的父子关系时才能进行强转, 否则就会出现类型转换异常。
14
15         //向下转型时需要进行类型强转
16         Cat cat=(Cat) animal;
17         //向下转型后, 可以使用子类Cat中的属性和方法。
18         cat.speak();
19     }
20 }
21
22 //测试类
23 public class AnimalTest {
24
25     public static void main(String[] args) {
26         Animal animal=new Cat();
27         animal.speak();
28
29         Cat cat=new Cat();
30         //这里我们传参时, 既可以传递animal, 也可以传递cat, 但不能传递dog类型, 否则会
        出java.lang.ClassCastException:
31         //因为class com.yyg.convert.Dog cannot be cast to class com.yyg.co
        nvert.Cat。狗不能被转成猫
32         //cat2.helloAnimal(dog);
33         cat.helloAnimal(animal);
34
35         //Dog dog = new Dog();
36         //这里就会编译出, 不允许把Dog对象类型转换成Cat对象类型
37         //Cat cat = (Cat)dog;
38     }
39
40 }
```


向下转型就是将父类型转为子类型，但要注意，此时有可能会出ClassCastException类型转换异常，因为子类一定属于父类的一员，但父类不一定属于子类。我们说“猫是动物”一定没问题，但如果说“动物是猫”，这个结论未必正确。所以把“动物强转成猫”的过程中，有可能会出异常。只有两者之间具有明确的父子关系时才能进行强转，否则就会出现类型转换异常。

就比如上面的案例中，我们传参时，既可以传递animal，也可以传递cat，但不能传递dog类型，否则会出现java.lang.ClassCastException，因为class com.yyg.convert.Dog cannot be cast to class com.yyg.convert.Cat，狗类不能被转成猫类。

-----正片已结束，来根事后烟-----

四. 结语

至此，壹哥就把类型转换给大家介绍完了，我们来看看类型转换的要点吧：：

- 向上转型是父类引用指向子类的对象，不必强制类型转换；
- 向下转型是子类对象指向父类引用，必须进行强制类型转换；
- 可以强制向下转型最好借助instanceOf进行类型判断；

在下一篇文章中，壹哥会给大家讲解instanceOf关键字的详细用法，敬请关注哦。另外如果你独自学习觉得有很多困难，可以加入壹哥的学习互助群，大家一起交流学习。

五. 今日作业

1. 第一题

设计一个父类Person和子类Student、Teacher，实验将Person与Student和Teacher进行类型转换。