Java 常见异常类型及原因分析（上）

这里介绍几种常见的异常并对原因进行分析，包括：

- **NullPointerException** 异常；
- **ClassCastException** 异常；

❶ NullPointerException 异常

顾名思义，NullPointerException 是空指针异常。但是在 Java 中没有指针，怎么会有空指针异常呢？

在 C++中，声明的指针需要指向一个实例（通过 new 方法构造），这个指针可以理解为地址。在 Java 中，虽然没有指针，但是有引用（通常称为对象引用，一般直接说对象），引用也是要指向一个实例对象（通过 new 方法构造）的，从这种意义上说，Java 中的引用与 C++中的指针没有本质的区别，不同的是，处于安全的目的，在 Java 中不能对引用进行操作，而在 C++中可以直接进行指针的运算，例如 book++等。

所以这里的 NullPointerException 虽然不是真正的空指针异常，但本质上差不多，是因为引用没有指向具体的实例，所以当访问这个引用的方法的时候就会产生这种异常。例如下面的代码：

```
String str = "这是一个测试用的字符串！";
System.out.println(str.length());
```

这段代码没有问题，但是如果改成下面的代码：

```
String str=null ;
System.out.println(str.length());
```

就会产生 NullPointerException 异常了：

```
Exception in thread "main" java.lang.NullPointerException
        at com.imooc.test.Test.main(Test.java:35)
```

那这种异常通常是如何产生的呢？比较多见的是下面的两种情况：

a) 把调用某个方法的返回值直接赋值给某个引用，然后调用这个引用的方法。在这种情况下，如果返回的值是 null，必然会产生 NullPointerException 异常。

例如：

```
public static void main(String[] args)
    People p=null;
    p.setName("张三");
    System.out.println(p.getName());
}
```

分析：声明一个 People 对象，并打印出该对象的中的 Name 值。

说明：这个时候你的 p 就出现空指针异常，因为你只是声明了这个 People 类型的对象并没有创建对象，所以它的堆里面没有地址引用，切忌你要用对象调用方法的时候一定要先创建对象。

b) 在方法体中调用参数的方法。

这种情况下，如果调用方法的时候传递进来的值是 null，也要产生 NullPointerException 异常。要解决这种异常，只需要检查异常出现在第几行（通常在集成开发环境中会提示用户错误发生在第几行），然后查看调用了哪个对象的方法，然后检查这个对象为什么没有赋值成功即可。

要避免程序产生这种异常，比较好的解决方法是在调用某个对象的方法时候判断这个对象是否可能为空，如果可能，则增加判断的语句，例如上面的代码可以写成：

```
if(str!=null){
    System.out.println(str.length());
}else{
    System.out.println(0);
}
```

❷ ClassCastException 异常

从字面上看，是类型转换错误，通常是进行强制类型转换时候出的错误。下面对产生 ClassCastException 异常的原因进行分析，然后给出这种异常的解决方法。

这种异常是如何产生的呢？举一个比较形象的例子。

Animal 表示动物，Dog 表示狗，是动物的子类，Cat 表示猫，是动物的子类。看下面的代码：

```
Animal a1 = new Dog();
Animal a2 = new Cat();
Dog d1 = (Dog)a1;
Dog d2 = (Dog)a2;
```

第 3 行代码和第 4 行代码基本相同，从字面意思看都是把动物（Animal）强制转换为狗（dog）。但是第 4 行代码将产生 java.lang.ClassCastException。原因是你要把一个猫（a2 这只动物是猫）转换成狗不可以，而第 3 行中是把狗转换成狗，所以可以。

从上面的例子看，java.lang.ClassCastException 是进行强制类型转换的时候产生的异常，

强制类型转换的前提是父类引用指向的对象的类型是子类的时候才可以进行强制类型转换，如果父类引用指向的对象的类型不是子类的时候将产生 java.lang.ClassCastException

异常。

遇到这样的异常的时候如何解决呢？如果你知道要访问的的对象的具体类型，直接转换成该类型即可。如果不能确定类型可以通过下面的两种方式进行处理（假设对象为 o）：

- 通过 o.getClass().getName()得到具体的类型，可以通过输出语句输出这个类型，然后根据类型进行进行具体的处理。
- 通过 if(o instanceof 类型) 的语句来判断 o 的类型是什么。