**反射**

参考：[https://blog.csdn.net/ljphhj/article/details/12858767](https://blog.csdn.net/ljphhj/article/details/12858767、)

<https://www.sczyh30.com/posts/Java/java-reflection-1/>

在程序的运行期间，对于任意一个类或任意一个对象，都可以获取和操作它们内部成员（属性、方法、构造方法）。这种动态获取和调用的方式叫反射机制。

其实反射的本质是必须要获取加载到方法区内的Class对象，然后通过这个Class对象来操作它内部的属性和方法。

在类加载机制第一个“加载”阶段，把一个编译好的Class文件内容以二进制流的方式加载到方法区，并创建一个相应class对象，作为这个类型在堆内存中实例对象的调用入口。其中反射机制就是通过这个Class对象来进行各种操作，假如方法区内还没有加载这个Class类型，则需要先加载这个Class类型。

**Class对象的获取**



Class对象的获取有如上三种方式，这三种方式获取的都是同一个Class对象，就是说，class1==class2==class3。

**通过反射获取一个类的实例对象**

这里的反射获取实例对象的话，指的是这个类中没有公共的构造函数，只有私有的构造函数，这时候可以通过反射调用这个类的私有构造方法构造一个响应的对象。相反，如果这个类中有公共的构造函数，可以通过这个公共构造函数直接new出一个对象来，也就不用使用反射来创建一个对象了。如下，其中MathDemo 是一个只有私有构造函数的类：

  /\*

    \* 使用反射创建构造方法私有化的类对象

    \* 类：MathDemo

    \* 构造方法：private MathDemo()

    \*/

    //1.获取类的无参构造方法

   Constructor<MathDemo> constructor = MathDemo.class.getDeclaredConstructor;

   //2.设置取消访问检查，是访问私有构造方法的关键

   constructor.setAccessible(true);

   //3.调用该构造方法，获得对象

   MathDemo mathDemo = constructor.newInstance();

**通过反射获取一个类中的属性（可以是基本类型也可以是引用类型对象）、方法（可以调用）**

参考：<https://blog.csdn.net/a997208868/article/details/48133129>

在上面链接中没有没有通过反射获取一个引用类型对象的例子，可以参考如下：

public class MathDemo {

    private int m = 10;

    private TextView tv;

    private MathDemo(){

        tv = new TextView();

    }

    private int add(int a,int b){

        return a+b;

    }

    public int getM() {

        return m;

    }

}

**如下，获取MathDemo类中的tv对象：**

**1、首先获取到MathDemo 对象**

Constructor<MathDemo> constructor = MathDemo.class.getDeclaredConstructor;

    constructor.setAccessible(true);

   MathDemo mathDemo = constructor.newInstance();

**2、然后获取到mathDemo的引用类型属性**

  Field field = MathDemo.class.getDeclaredField("tv");

  field.setAccessible(true);

  TextView mTv = (TextView) field .get(mathDemo);//获得该属性对应的对象

  mTv.setText(“修改成功”);