·Class类的使用

·在面向对象的世界里，万事万物皆对象。

·java语言中，静态的成员、普通数据类型不是对象。

·类是对象，类是java.lang.Class类的对象（There is a class named Class）

·Class类的实例对象的三种表示方式

· Foo的实例对象如何表示

Foo foo1 = new Foo();//foo1就表示出来了。

//Foo这个类也是一个实例对象，Class类的实例对象，如何表示呢？

//任何一个类都是Class的实例对象，这个实例对象有三种表示方式

·第一种表示方式--->实际在告诉我们任何一个类都有一个隐含的静态成员变量class

Class c1= Foo.class;

·第二种表示方式--->已经知道该类的对象通过getClass方法

Class c2=foo1.getClass();

/\*官网c1,c2表示了Foo类的类类型（class type）

\* 万事万物皆对象，

\* 类也是对象，是Class类的实例对象

\* 这个对象我们称为该类的类类型

\* \*/

//不管c1 or c2都代表了Foo类的类类型，一个类类型，一个只可能是Class类的一个实例对象

System.out.println(c1==c2);

·第三种表达方式

Class c3=Class.forName("com.imooc.reflect.Foo");

//我们完全可以通过类的类类型创建该类的对象实例--->通过c1 or c2 or c3创建Foo的实例

Foo foo = (Foo)c1.newInstance();

class Foo{

}

·Class.forName("类的全称")

·不仅表示了类的类类型，还代表了动态加载类

·请大家区分编译、运行

·编译时刻加载类是静态加载类，运行时刻加载类是动态加载类

·Class.forName();是动态加载类，在运行时刻加载类

·new创建对象是静态加载类，在编译时刻就需要加载所有的可能使用到的类，只要有 一个不存在，那么其他的类也都不可用。

·基本的数据类型，void关键字，都存在类类型。

·Class类的基本API操作

·方法的反射

·如何获取某个方法

方法的名称和方法的参数列表才能唯一决定某个方法

·方法反射的操作

method.invoke(对象，参数列表)

·成员变量的反射

·构造函数的反射

·Java类加载机制