java反射机制

java反射机制的API，位于：java.lang.Reflect

一、字面意思

什么叫反射？

光的反射。最常见的：照镜子。

照镜子，是为了自我审查，得到自己的信息，发现不合心意的地方，自我整理（自己修改自己的信息）。

二、java对类的执行原理

1、开发步骤：

1）、在我们开发java程序的时候，需要自己编写代码，后缀是：.java。这是java的源文件。

所谓的源文件，是给人看的，不是给计算机看的。

2）、编译：

将人看的java源文件，转换成.class文件。

.class，是一个字节码文件，并不是原生的二进制字节码。

class文件是对计算机认识的二进制的描述文件。

底层才机器码。（二进制）

3）、运行

java虚拟机（JVM）首先通过IO读取.class文件。

类加载器（Classloader）加载这个类的信息，转成二进制的机器码，放到内存。

当我们第一次创建这个类的对象的时候：

a、java虚拟机创建这个类的Class对象。（java.lang.Class类）

b、通过这个Class对象，来实例化出一个实例（对象）

```
package reflect;

public class Main {
    public static void main(String[] args) {
        //创建出了Student的实例
        //Student是一个类，什么是类？
        //类是具有相同属性和行为的群体的一个模板。
```

```
        //类是创建实例的“模具”。平时看到的类：Student.java。
        //在java虚拟机中，会将类加载去进去后，创建出这个类的Class对象。这才是真正的模具。
        Student stu = new Student();
        stu.test();
    }
}
```

2、java.lang.Class对象

Class对象是一个特殊的对象，是用来创建其它对象的对象（这里的其他对象就是指：java类的实例）。其实Class对象就是java类编译后生成的.class文件，它包含了与类有关的信息。

每当第一次使用一个类时，JVM必须使用“类加载器”子系统加载该类对象的Class对象。一旦这个类的Class对象被载入内存，它就被用来创建这个类的所有对象。当我们使用new关键字创建一个类的第一个对象的时候，JVM会帮助我们加载该类Class对象，但是当我们想自己加载这个类的Class对象怎么办呢？实际上有3种方法：

1. Class.forName("类名字符串")（注意：类名字符串必须是全称，包名+类名）
2. 类字面常量法：类名.class
3. 实例对象.getClass()

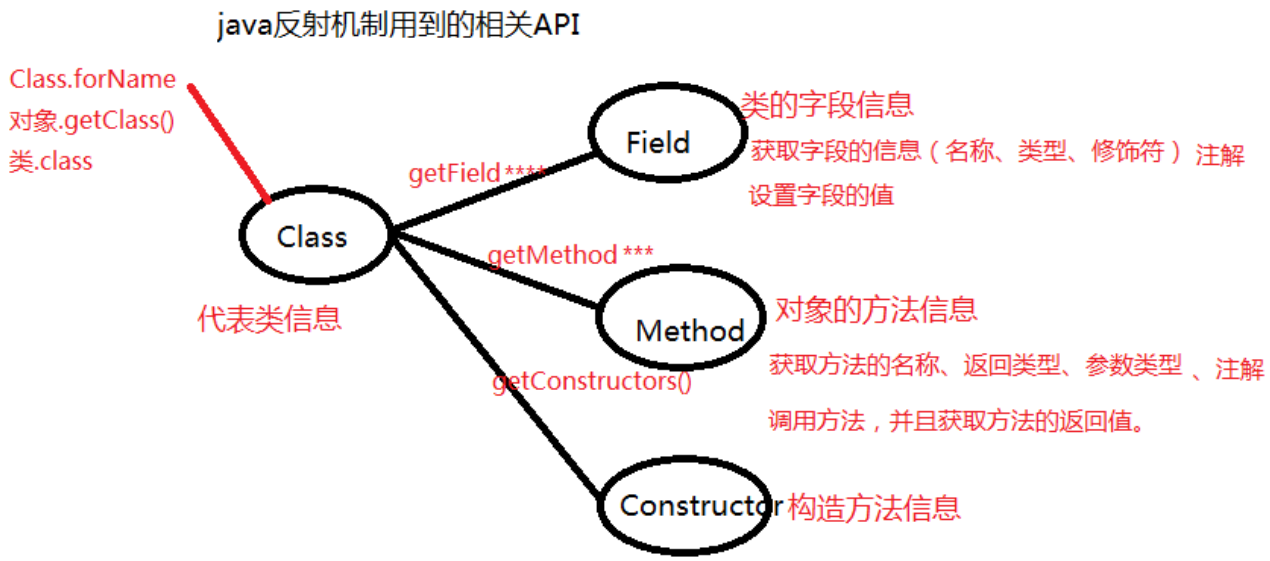
```
package reflect;

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;

public class Main {
    public static void main(String[] args) throws NoSuchMethodException, SecurityException,
IllegalAccessException, IllegalArgumentException, InvocationTargetException {
        // 创建出了Student的实例
        // Student是一个类，什么是类？
        // 类是具有相同属性和行为的群体的一个模板。
        // 类是创建实例的“模具”。平时看到的类：Student.java。
        // 在java虚拟机中，会将类加载去进去后，创建出这个类的Class对象。这才是真正的模具。
        Student stu = new Student();
        //使用：对象.getClass（）来获取这个类的class对象
        Class<? extends Student> clazz = stu.getClass();

        // class对象，获取类的属性
        Field[] declaredFields = clazz.getDeclaredFields();
        for (Field field : declaredFields) {
            System.out.println(String.format("字段名：%s，字段类型：%s", field.getName(),
field.getType()));
        }

    }
}
```



3、Filed、Method、Constructor

都位于java.lang.reflect包下。

通过Class对象，可以得到该类中所有的信息。

包含：字段（Filed）、方法（Method）、构造方法（Constructor）、Annotation（注解）

通过这几个类，可以获取字段信息、方法信息、构造方法的信息、注解信息。

还可以：在程序运行过程中，设置字段的值、动态的调用方法、创建对象的实例（调用构造方法）。

```
//常用的方法：
Class类：
forName(String className)     整个反射机制中，最重要的一个方法。
        动态的加载一个类。得到该类的Class对象。
//获取字段
Field getDeclaredField(String name)根据字段名获取指定的字段，包括public、private，封装Field对象。
Field[] getDeclaredFields()  获取该类的所有字段，包括public、private，封装成Filed数组。
Field getField(String name)  根据字段名获取指定的字段获取指定的public的字段，封装字段对象。
Field[] getFields()  获取所有的public字段。

//获取方法
Method getMethod(String name, Class<?>... parameterTypes)
     获取public方法。方法名和参数，与给定的保持一致。
     为什么要带上参数列表？因为java中存在方法重载（方法名相同，参数列表（参数的类型、个数、顺序）不同）
Method[] getMethods()   获取所有的public方法
Method getDeclaredMethod(String name, Class<?>... parameterTypes)
    获取指定名称和参数列表的方法，包括public、private
Method[] getDeclaredMethods()
    获取所有的方法，包括public、private

//获取构造方法
Constructor<T> getConstructor(Class<?>... parameterTypes)
```

```
        获取指定参数列表的public的构造方法
Constructor<?>[] getConstructors()
        获取所有的public构造方法

//获取注解（Class、Field、Method、Constructor都有）
<A extends Annotation> A getAnnotation(Class<A> annotationClass)
    获取指定的注解
Annotation[] getAnnotations()
    获取所有的注解
```