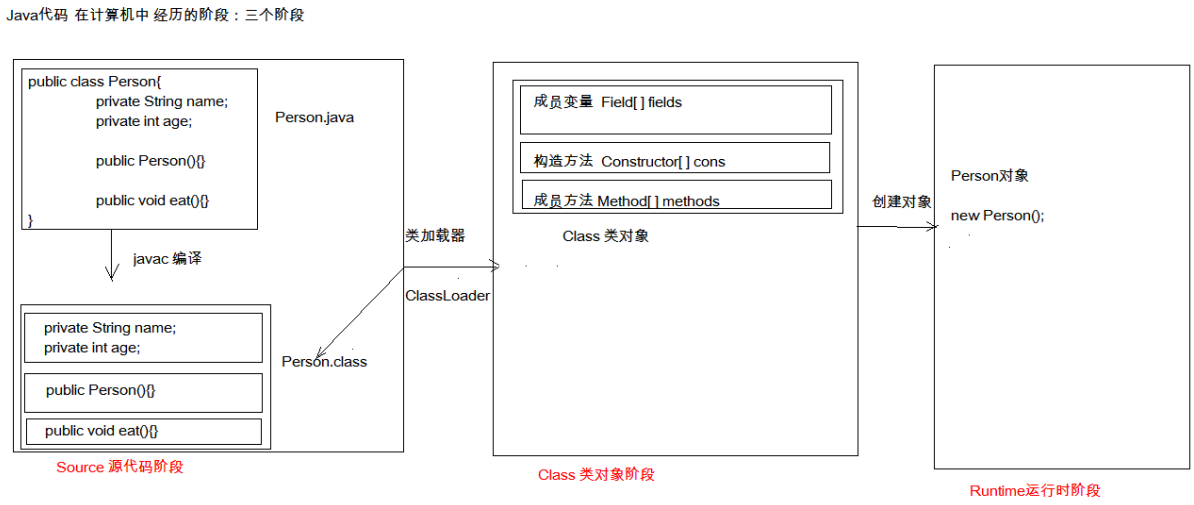
反射：框架设计的灵魂



第一个阶段Source源代码文件阶段里.java文件与.class文件都在硬盘上，此时都还没有进入硬盘上面；

第三个阶段Runtime阶段里new创建的对象是在内存里面，字节码要想变成对象，即需要把字节码文件加载进内存之后才能够创建这个对象；

第二个阶段Class类对象阶段里通过类加载器把字节码文件加载到内存里面去，在内存里面也需要用一个对象去描述这个Person.class文件，在内存里面会有一个对象来描述这个字节码文件，这个对象叫Class类对象（这个类的名字就叫做class，这个类是用来描述所有字节码文件的共同的特征与行为，不管是什么字节码文件，里面都有一些共同的，即中间的框里的三个（当然除了这三个还有其他的共同的，但没有画，只是这三个比较重要））；与可能有好多个成员变量，所以要用一个Field数组

```
* 框架：框架是半成品软件。可以在框架的基础上进行软件开发，简化编码
* 反射：将类的各个组成部分封装为其他对象，这就是反射机制（概念比较抽象）
        注：在上图中看，将类的成员变量封装成成员变量Field对象；将Person对象封装成一个Construct对象；将成员方法封装成Method对象，这就是反射机制
    * 好处：
        1. 可以在程序运行过程中，操作这些对象。(即IDEA的自动方法提醒)
        2. 可以解耦，提高程序的可扩展性。
* 获取Class对象的方式：
    1. Class.forName("全类名")：将字节码文件加载进内存，返回Class对象
        * 前提是字节码文件还未加载到内存
        * 多用于配置文件，将类名定义在配置文件中。读取文件，加载类
        * 全类名:包名和类名都得有
    2. 类名.class：通过类名的属性class获取
        * 前提是字节码文件已经加载到了内存
        * 多用于参数的传递
```

3. 对象.getClass()：getClass()方法在Object类中定义着。

* 前提是已经创建对象成功了,任何一个对象都有getclass这个方法,这个方法是封装在Object类里面的,被所有的对象都继承下来了

* 多用于对象的获取字节码的方式

* 结论：

同一个字节码文件(*.class)在一次程序运行过程中，只会被加载一次，不论通过哪一种方式获取的Class对象都是同一个。

* Class对象功能：

* 获取功能：

1. 获取成员变量们

* Field[] getFields() ：获取所有public修饰的成员变量

* Field getField(String name) 获取指定名称的 public修饰的成员变量

* Field[] getDeclaredFields() 获取所有的成员变量，不考虑修饰符

* Field getDeclaredField(String name) 获取指定的成员变量，不考虑修饰符

2. 获取构造方法们

* Constructor<?>[] getConstructors()

* Constructor<T> getConstructor(类<?>... parameterTypes)

* Constructor<T> getDeclaredConstructor(类<?>... parameterTypes)

* Constructor<?>[] getDeclaredConstructors()

3. 获取成员方法们：

* Method[] getMethods()

* Method getMethod(String name, 类<?>... parameterTypes)

* Method[] getDeclaredMethods()

* Method getDeclaredMethod(String name, 类<?>... parameterTypes)

4. 获取全类名

* String getName()

5.Class对象的newInstance方法

* 使用空参数构造方法创建对象

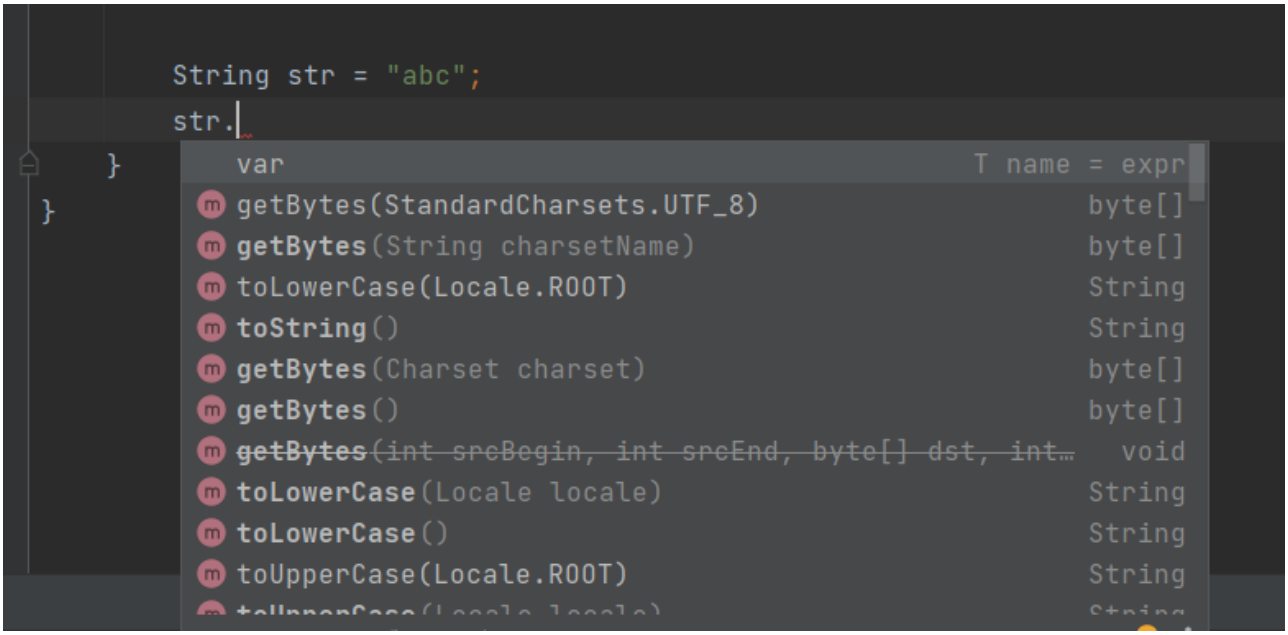
* Field：成员变量

* 操作：

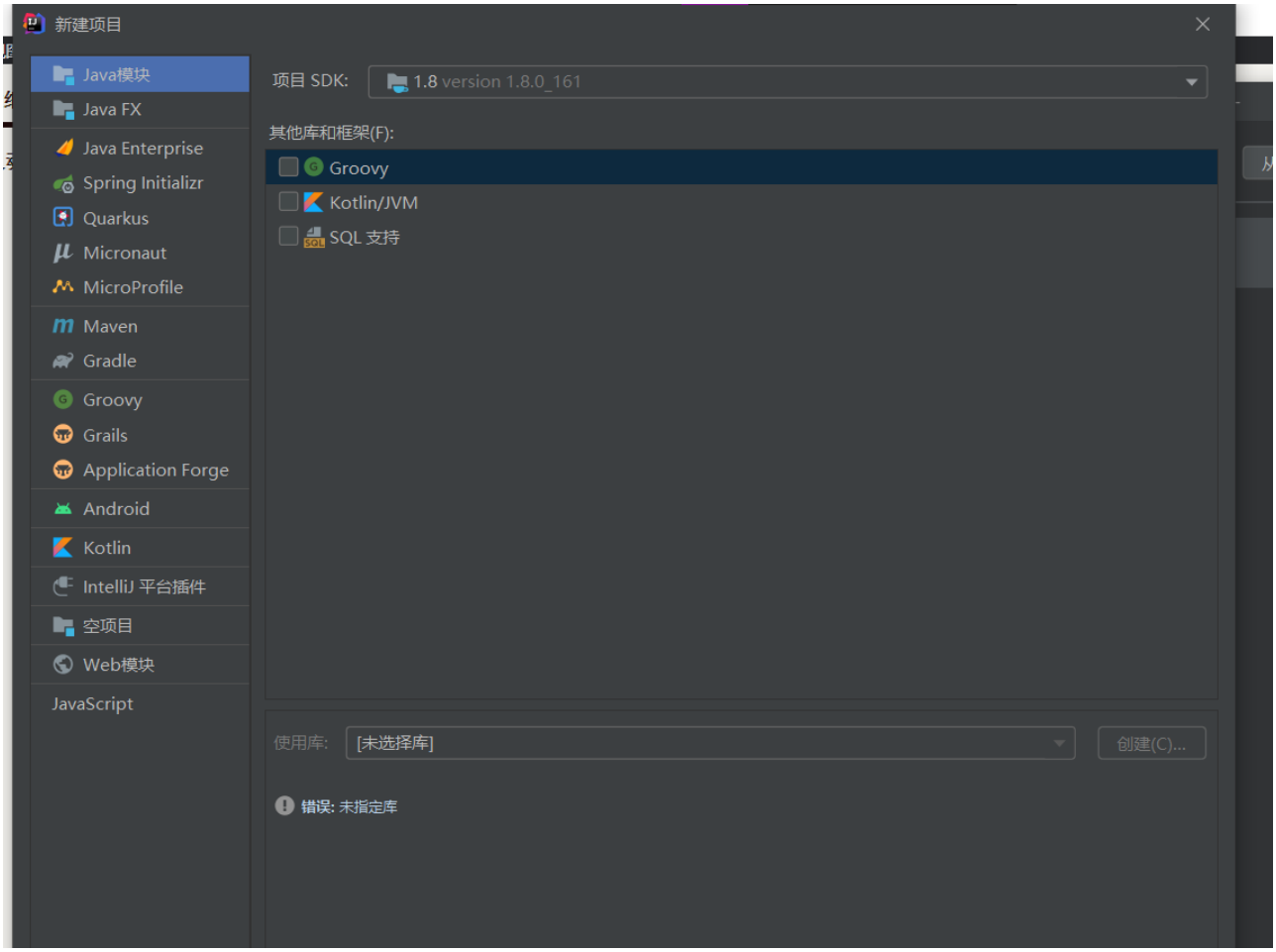
```
        1. 设置值
            * void set(Object obj, Object value)
        2. 获取值
            * get(Object obj)

        3. 忽略访问权限修饰符的安全检查
            * setAccessible(true):暴力反射
* Constructor:构造方法
    * 创建对象：
        * T newInstance(Object... initargs)

        * 如果使用空参数构造方法创建对象，操作可以简化：Class对象的
newInstance方法
* Method：方法对象
    * 执行方法：
        * Object invoke(Object obj, Object... args)

    * 获取方法名称：
        * String getName:获取方法名
* 案例：
    * 需求：写一个"框架"(加引号指不是真正的框架)，不能改变该类的任何代码的前提
下，可以帮我们创建任意类的对象，并且执行其中任意方法
        * 实现：
            1. 配置文件
            2. 反射
        * 步骤：
            1. 将需要创建的对象的全类名和需要执行的方法定义在配置文件中
            2. 在程序中加载读取配置文件
            3. 使用反射技术来加载类文件进内存
            4. 创建对象
            5. 执行方法
```

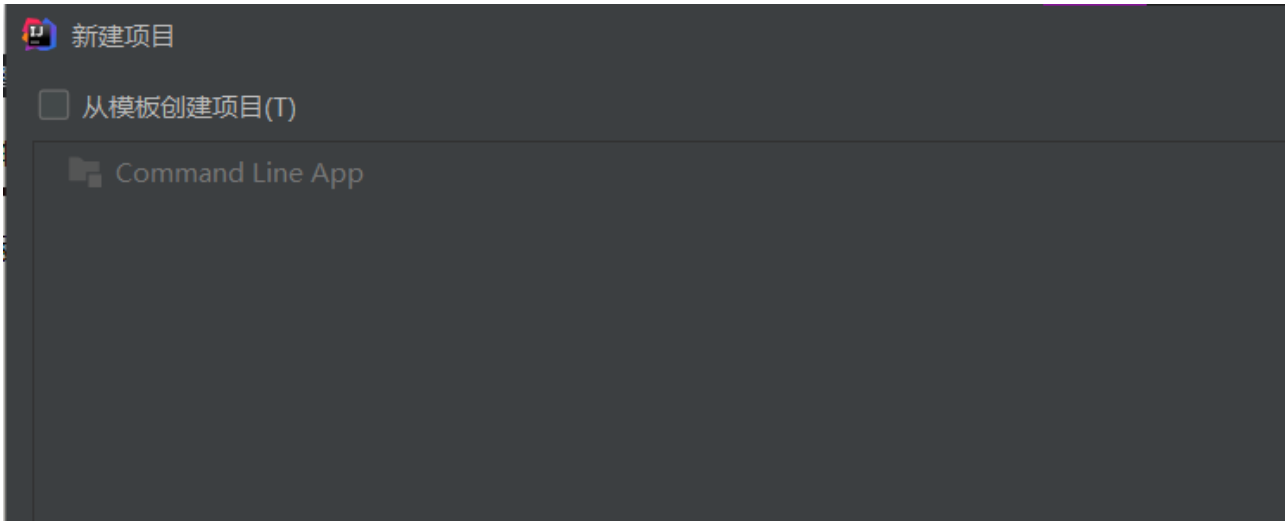
- 反射的好处,可以在程序运行过程中，操作这些对象:

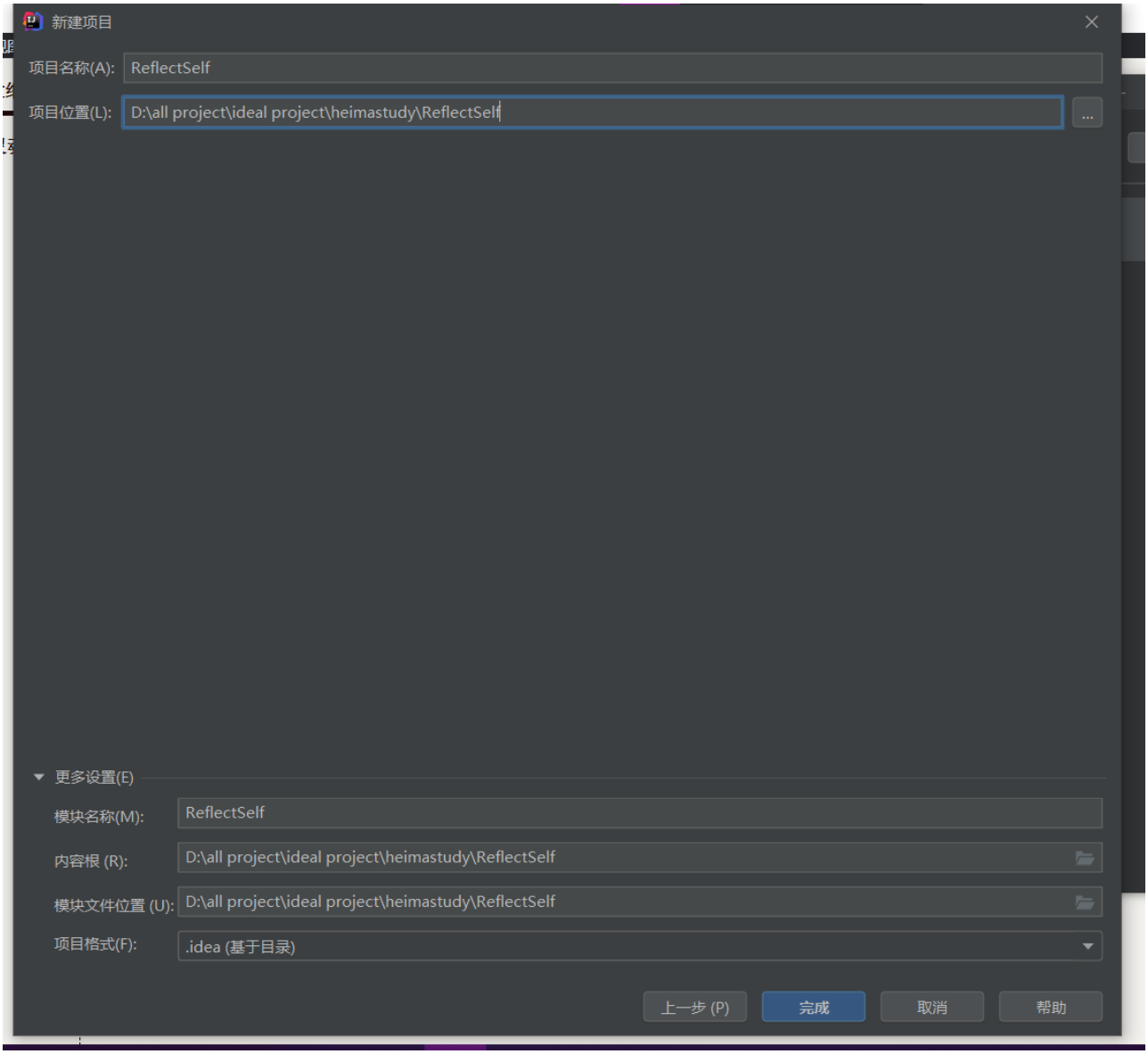
  比如我们定义了一个字符串，写上str.后会提示出来很多方法,那么这些方法是从哪来的呢?其实内部用的就是反射机制,我们在定义了一个字符串之后,它会把这个字节码文件加载进内存,在内存中有一个class类对象,class类对象已经把所有的方法都抽取出来封装成为了Method[s这个对象,把str所有的方法都放进去都放进去了一个Method()里面,IDEA一直在运行当中,所以写上str.就会产生方法提醒

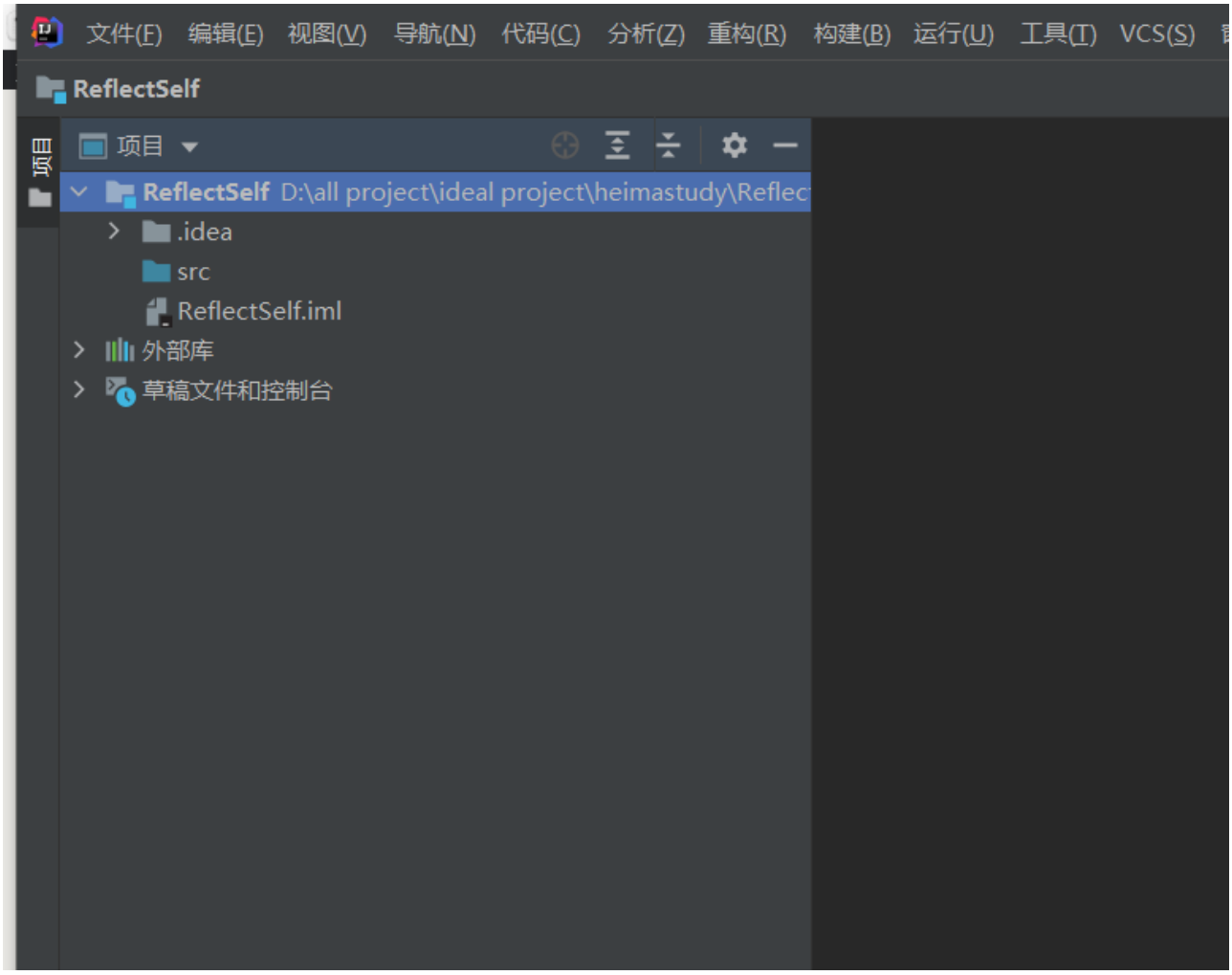


- 创建,进去后不需要增添框架




新建项目
从模板创建项目(T)
Command Line App


新建项目
项目名称(A): ReflectSelf
项目位置(L): D:\all project\ideal project\heimastudy\ReflectSelf
更多设置(E)
模块名称(M): ReflectSelf
内容根 (R): D:\all project\ideal project\heimastudy\ReflectSelf
模块文件位置 (U): D:\all project\ideal project\heimastudy\ReflectSelf
项目格式(F): .idea (基于目录)
上一步 (P)
完成
取消
帮助


文件(F) 编辑(E) 视图(V) 导航(N) 代码(C) 分析(Z) 重构(R) 构建(B) 运行(U) 工具(T) VCS(S)
ReflectSelf
项目
项目
ReflectSelf D:\all project\ideal project\heimastudy\Reflec
.idea
src
ReflectSelf.iml
外部库
草稿文件和控制台