Java高级_反射

课程概要

反射概述

反射方式获取构造方法并使用

反射方式获取成员方法并使用

反射方式获取成员变量并使用

学习目标

理解反射的概念

能够使用反射技术获取Class对象

能够使用反射技术获取构造方法并使用

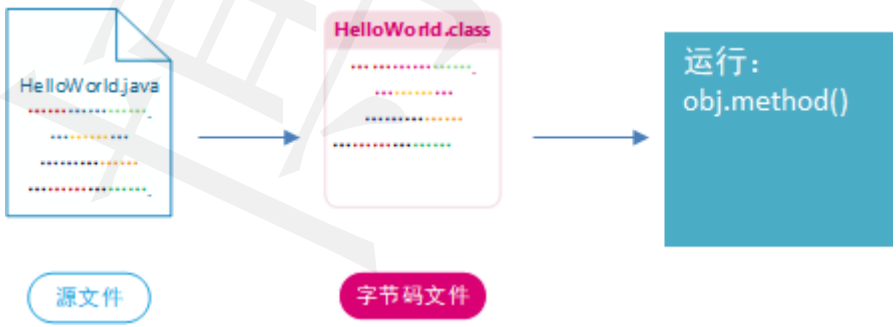
能够使用反射技术获取成员方法并使用

能够使用反射技术获取成员变量并使用

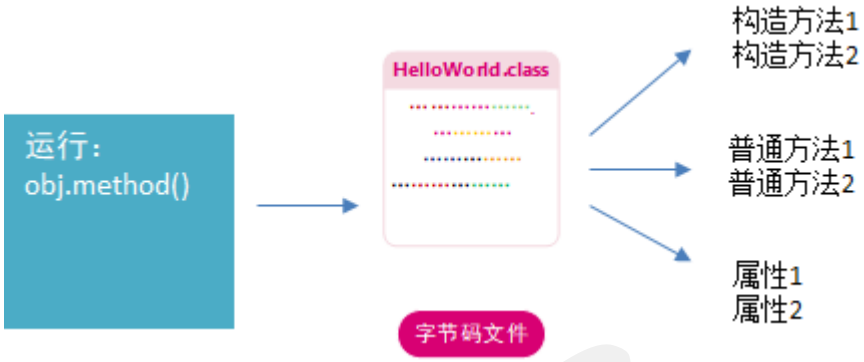
反射概述

原来的开发流程：程序代码是固定的，方法的执行顺序也是固定的

源代码 -> 字节码 -> 运行



使用反射：在需要的时候执行对应对象的方法，更改了程序执行的顺序，大大提高了程序的灵活性



反射技术致力于构建通用的底层代码，让所有的模块调用，提高程序的扩展性。

什么是反射?

在程序运行过程中分析类的一种能力

反射能做什么?

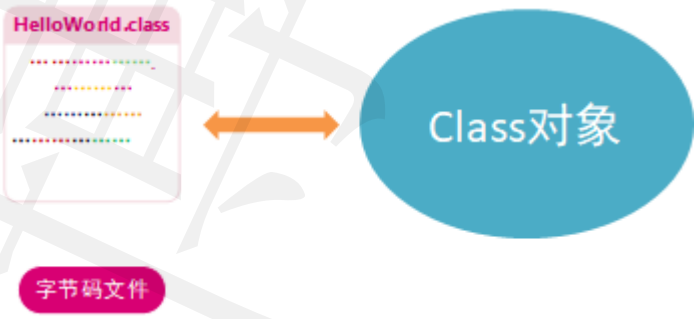
- 分析类
- 加载并初始化一个类
- 查看类的所有属性和方法
- 查看并使用对象
- 查看一个对象的所有属性和方法
- 使用对象的任意属性和方法

反射的应用场景

- 构建通用的工具
- 搭建具有高度灵活性和扩展性的系统框架

类加载器（ClassLoader）

负责将类的字节码文件（.class文件）加载到内存中，并生成对应的Class对象



Class对象

java.lang.Class类的对象，也叫字节码文件对象，每个Class对象对应一个字节码文件

类的加载时机

创建类的实例

```
Student stu = new Student();
```

访问类的静态成员

```
Calendar.getInstance();
```

初始化类的子类

```
class User extends Person {}
User user = new User(); // 先加载父类
```

反射方式创建类的Class对象

```
Class clazz = Class.forName("类的正名");
```

获取Class对象的三种方式

Object类的getClass()方法

```
Class clazz = 对象名.getClass();
```

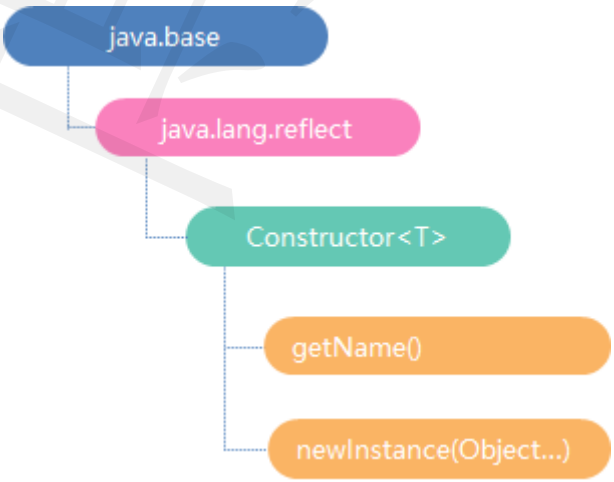
类的静态属性

```
Class clazz = 类名.class;
```

Class类的静态方法

```
Class clazz = Class.forName("类的正名"); // 正名：包类路径名，如：cn.itcast.bean.Student
```

反射方式获取构造方法并使用



Constructor对象

构造器对象，属于java.base模块，java.lang.reflect包

通过Class对象获取构造器对象

```
getConstructor(Class<?>… parameterTypes)
    // 返回一个Constructor对象，仅公共构造函数
    // Class<?>… ：可变参数，代表Class类型的数组
    // ? ：通配符，代表不确定的任意类型
```

```
getDeclaredConstructor(Class<?>… parameterTypes) // 返回一个Constructor对象，可获取私有构造函数
```

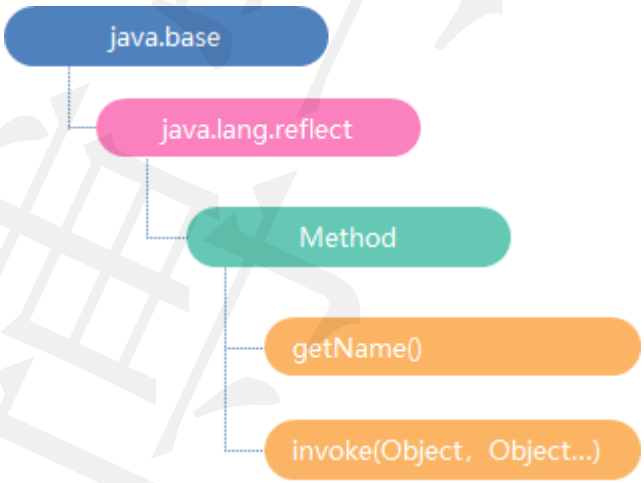
```
getConstructors() // 返回此类所有（不含私有）构造函数的数组
```

Constructor的常用方法

```
String getName() // 返回构造函数名
```

```
T newInstance(Object… initargs) // 使用此构造函数和指定参数创建并初始化对象
```

反射方式获取成员方法并使用



Method对象

方法对象，属于java.base模块，java.lang.reflect包

通过Class对象获取方法

```
getMethod(String name, Class<?>… parameterTypes)
    // 返回一个Method对象，仅公共成员方法
    // name ：方法名
        // parameterTypes ：方法的参数列表
```

```
getDeclaredMethod(String, Class<?>…) // 返回一个Method对象，可获取私有成员方法
```

```
getMethods() // 返回此类所有（不含私有）方法的数组
```

Method的常用方法

```
String getName() // 返回方法名
```

```
Object invoke(Object obj, Object… args) // 在指定对象上调用此方法，参数为args
```

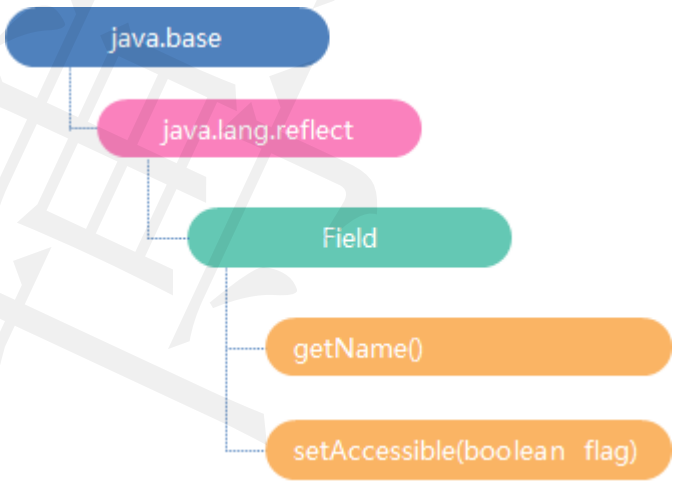
案例：通过反射获取方法并使用

需求：通过反射获取类的setter方法，使用该方法为属性赋值

分析：

1. setter方法的方法名由set和属性名（首字母大写）组成： 举例：setName, setAge
2. setter方法有且只有一个参数，参数类型为属性的类型： 举例：setName(String name), setAge(int age)
3. setter方法为public修饰的方法，反射获取该方法使用： getMethod(String , Class<?>...);
4. 根据上述分析分别为属性name、age赋值并使用

反射方式获取成员变量并使用



Field对象

域（属性、成员变量）对象，属于java.base模块，java.lang.reflect包

通过Class对象获取属性

```
getField(String name) // 返回一个Field对象，仅公共属性
        // name : 属性名
```

```
getDeclaredField(String name) // 返回一个Field对象，可获取私有属性
```

```
getDeclaredFields() // 返回此类所有（含私有）属性的数组
```

Field的常用方法

```
String getName() // 返回方法名
```

```
boolean setAccessible(boolean flag) // 将此属性的可访问性设置为指定布尔值
```