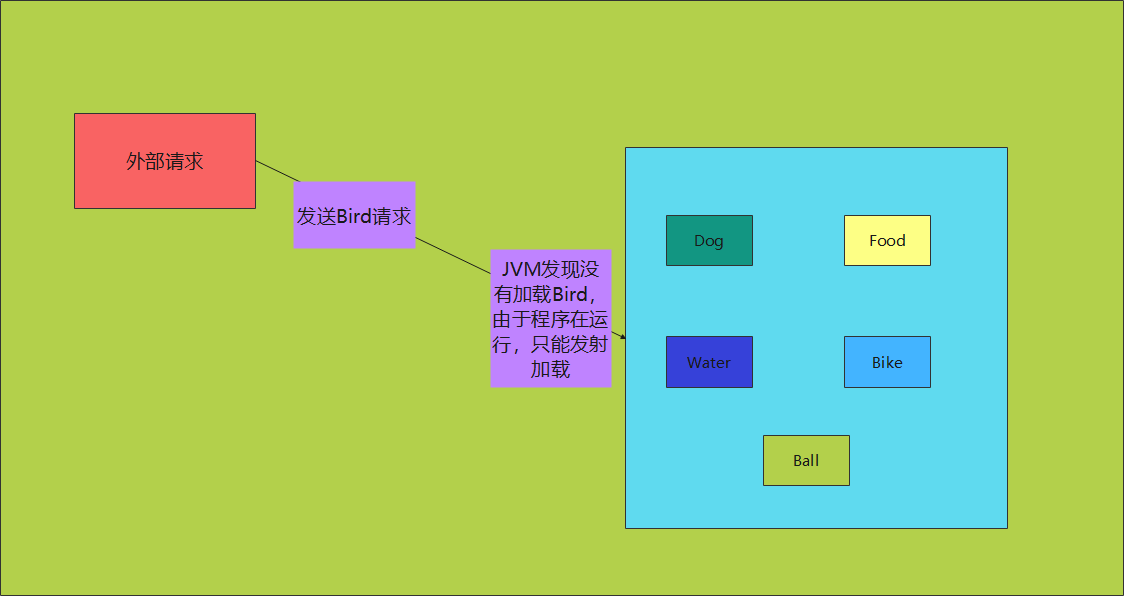
反射作用---->需要用到的时候才加载

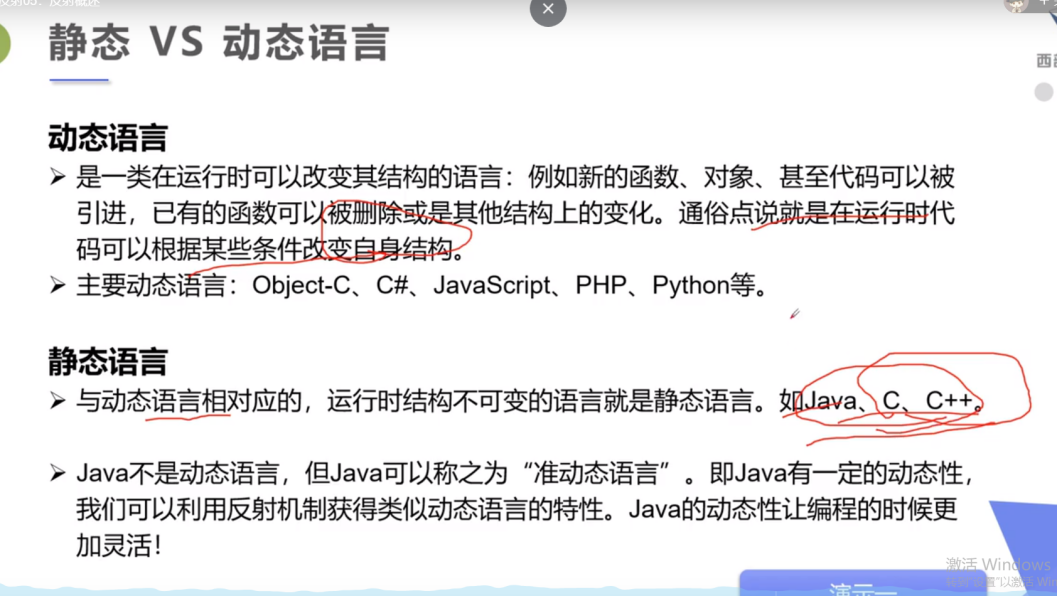
程序运行到某个阶段，发现某个类没有加载，由于程序在运行所以不可能new出一个对象（new是写死的操作），此时就可以通过反射获取该类的Class对象（加载类），然后创建一个实例对象。



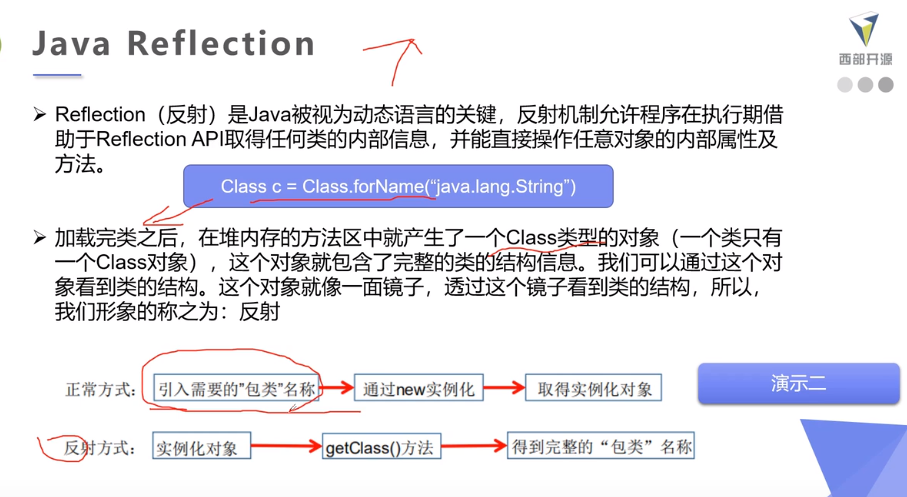
1. 语言类型

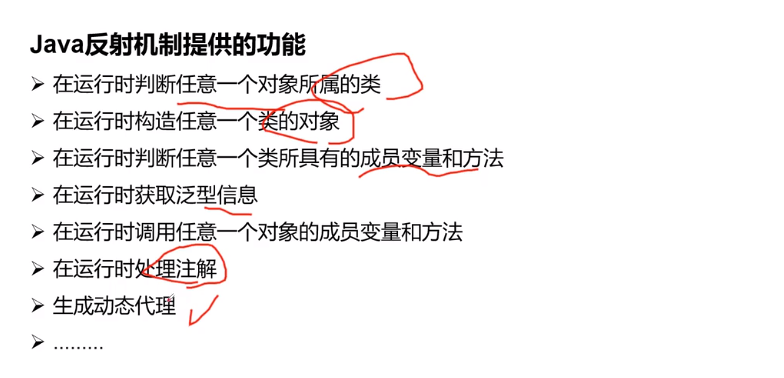
静态语言：运行时结构不可变

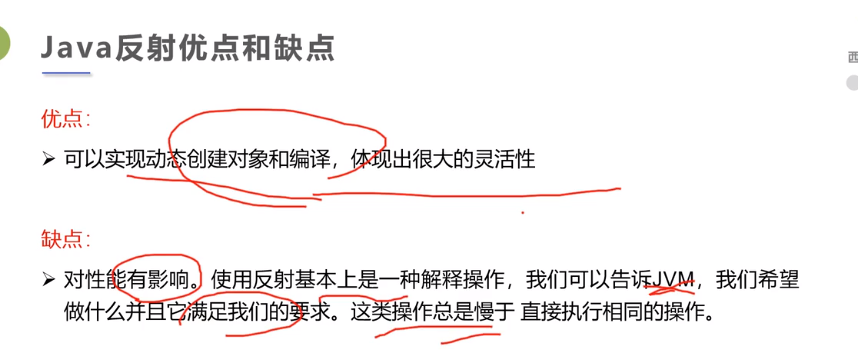
动态语言：运行时结构可变



1. 反射



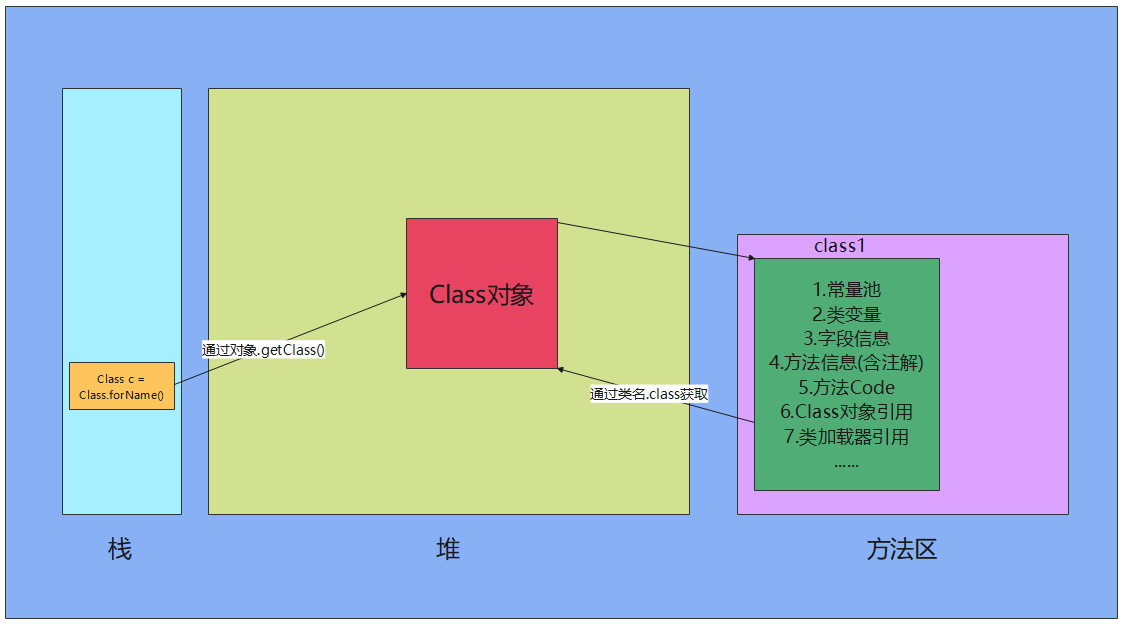






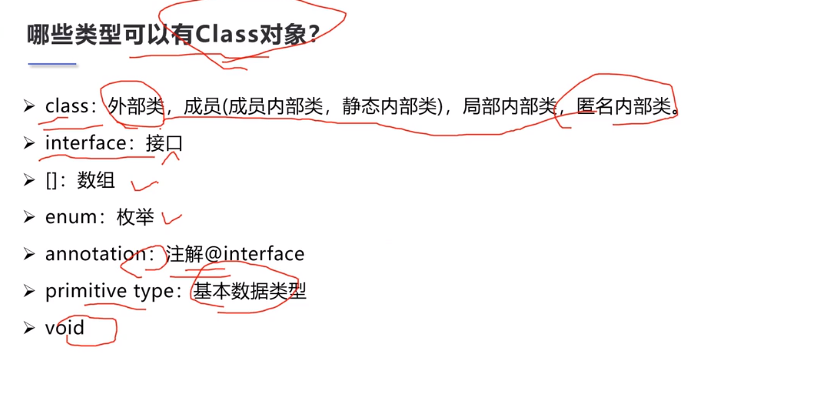
1. 获取Class对象

（a）获取类对应的Class对象（类加载阶段生成的，用于访问方法区类数据）

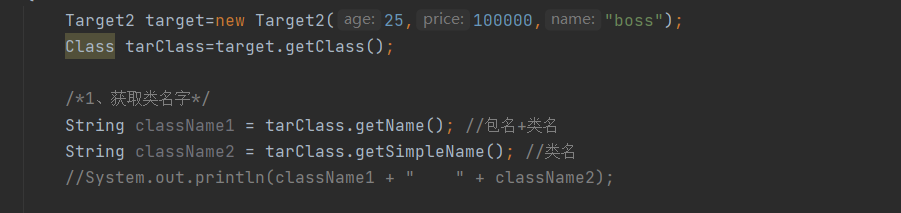


1. Class.forName()：直接加载，如果加载过了就返回已经加载的Class
2. 实例对象.getClass()：通过访问实例对象的对象头里面的类型信息
3. 类型.class：获取类型对应的Class对象

（b） 哪些类型有有class对象



1. 通过Class获取类的结构
2. 类名



1. 属性（本类使用getDeclaredField，父类public方法使用getField）



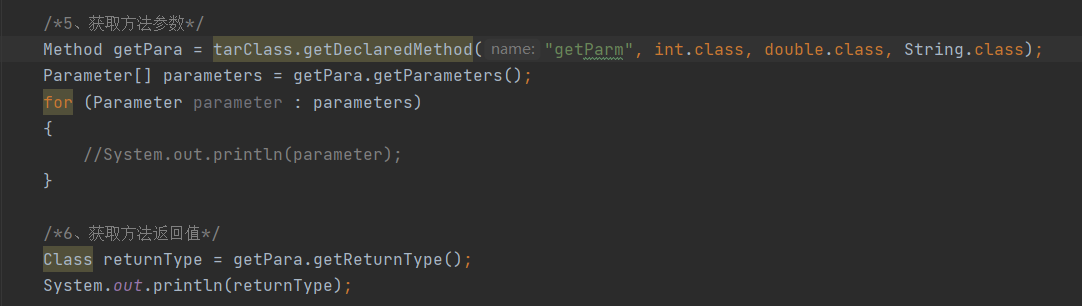
1. 方法（本类使用getDeclaredMethod，父类public方法使用getMethod）



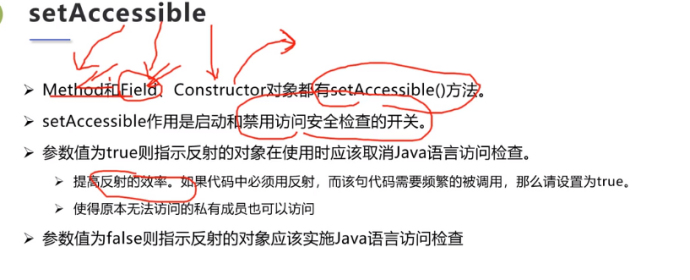
1. 构造器（使用getDeclareConstructor）



1. 方法参数、返回值



1. 使用类结构（需要先有对象，才能操作字段）



1. 实例对象（Class对象的无参构造创建；构造器实例创建）



1. 执行对象方法（如果方法为private，需要执行setAccessible）

method.invoke(对象,参数值)

invoke相当于可以动态变化（根据参数不同，执行不同对象的不同方法）以及可以执行private方法。这样程序不用写死，如果直接调用对象方法，相当于程序写死，扩展性不好。



1. 操作对象属性（如果属性为private，需要执行setAccessible）

field.set(对象,属性值)

field.get(对象)



1. 性能比较

