反射

- 连老师

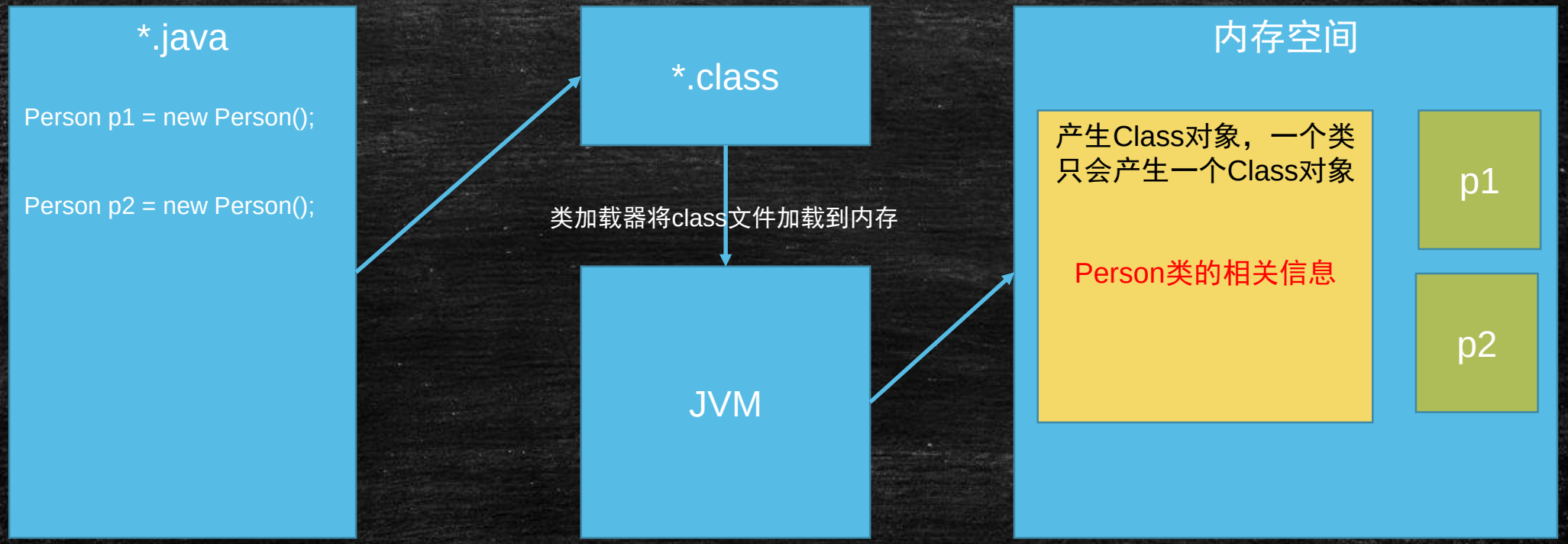
反射是什么

- Java中最强大的技术之一

反射概述

- **JAVA反射**机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为**java**语言的**反射**机制。
- 要想解剖一个类,必须先要获取到该类的字节码文件对象。而解剖使用的就是Class类中的方法.所以先要获取到每一个字节码文件对应的Class类型的对象.

类的加载过程



Class类的实例表示正在运行的Java应用程序中的类和接口，每个类只会产生一个Class对象，在类加载的时候自动创建

Class类

compact1, compact2, compact3
java.lang

Class Class<T>

java.lang.Object
    java.lang.Class<T>

参数类型

T - 由此类对象建模的类的类型。 例如， String.class的类型是Class<String> 。 如果正在建模的类是未知的，请使用Class<?> 。

All Implemented Interfaces:

Serializable , AnnotatedElement , GenericDeclaration , Type

```
public final class Class<T>
extends Object
implements Serializable, GenericDeclaration, Type, AnnotatedElement
```

Class类的类表示正在运行的Java应用程序中的类和接口。 枚举是一种类，一个注释是一种界面。 每个数组也属于一个反映为类对象的类，该对象由具有相同元素类型和维数的所有数组共享。 short， int， long， float和double），和关键字void也表示为类对象。

类没有公共构造函数。 相反， 类对象由Java虚拟机自动构建，因为加载了类，并且通过调用类加载器中的defineClass方法。

以下示例使用类对象来打印对象的类名称：

```
void printClassName(Object obj) {
    System.out.println("The class of " + obj +
                       " is " + obj.getClass().getName());
}
```

也可以使用类文字获取类对象作为命名类型（或为void）。 参见*The Java™ Language Specification*第15.8.2节。 例如：

获取Class类的三种方式

- 当用户想要获取任何一个Class类有三种方式：
  - Object→getClass()
  - 任何数据类型都有一个静态的class属性
  - 通过Class.forName()
- 三种方式的对比：
  - 第一种已经创建对象，就意味着已经产生了Class类
  - 第二种需要导入对应的包，依赖太强
  - 第三种常用，只需要传入一个类的完全限定名即可

反射的常用api

- 获取类的构造方法
- 获取类的成员变量
- 获取类的成员方法