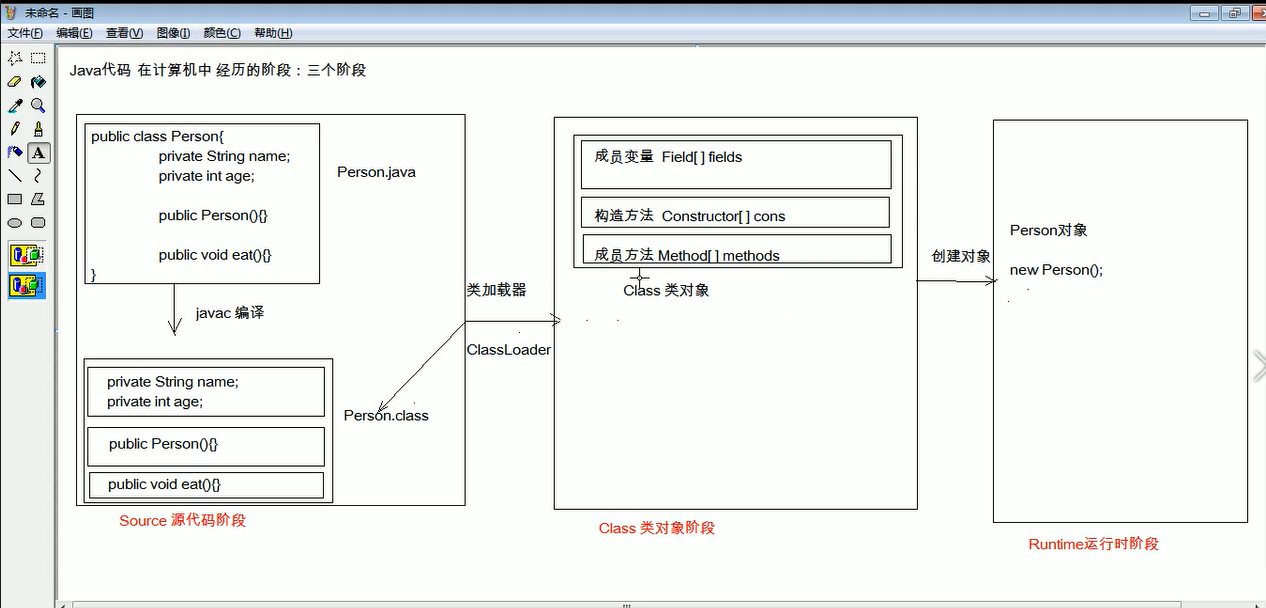
反射：框架设计的灵魂

框架：半成品软件，可以在框架的基础上进行软件开发，提高开发效率

反射：将类的各个组成部分封装为其他的对象，这就是反射

好处：

1. 在程序册运行过程中操作这些对象
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

\*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

1. 类名.class:通过类名的属性class获取

\*多用于参数传递

1. 对象.getClass():getClass方法在Object类中定义

\*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）在程序运行过程中，只会被加载一次，不论通过哪种方式获取的Class对象都是同一个

Class对象的功能

\*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

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

\*Filed getDeclaredField(String name) 获取指定名称的成员变量，不考虑修饰符

1. 获取构造方法们

\* Constructor<?>[] getConstructors()

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

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

\* Constructor<?>[] getDeclaredConstructors()

1. 获取成员方法们

\*\* Method[] getMethods()

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

\* Method[] getDeclaredMethods()

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

1. 获取全类名

\* String getName()

\*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"，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