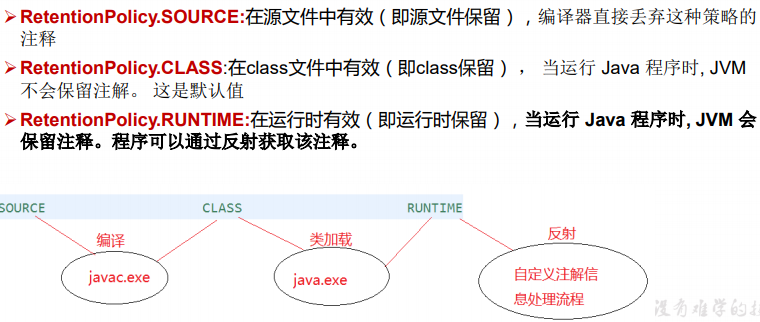
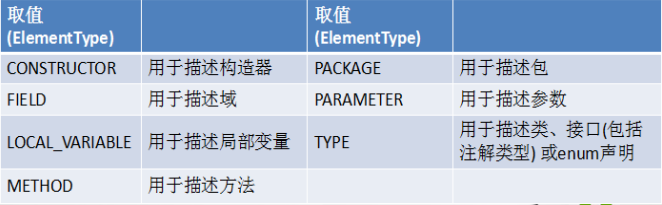
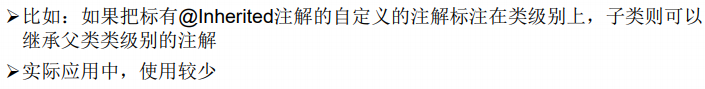
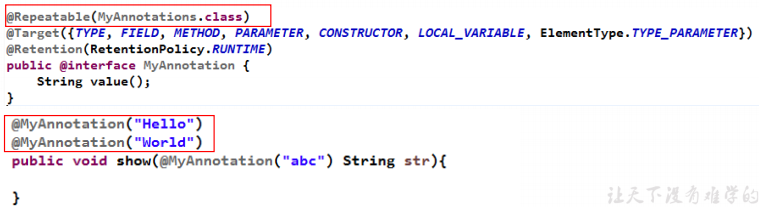
注解：

1. 代码里的特殊标记, 这些标记可以在编译, 类加 载, 运行时被读取, 并执行相应的处理
2. 可用于修饰包, 类, 构造器, 方法, 成员变量, 参数, 局部变量的声明，这些信息被保存在 Annotation 的 “name=value” 对中
3. 生成文档相关的注解：
   1. @author 标明开发该类模块的作者，多个作者之间使用,分割
   2. @version 标明该类模块的版本
   3. @see 参考转向，也就是相关主题
   4. @since 从哪个版本开始增加的
   5. @param 对方法中某参数的说明，如果没有参数就不能写
   6. @return 对方法返回值的说明，如果方法的返回值类型是void就不能写
   7. @exception 对方法可能抛出的异常进行说明 ，如果方法没有用throws显式抛出的异常就不能写
   8. 其中 @param @return 和 @exception 这三个标记都是只用于方法的。
   9. @param的格式要求：@param 形参名 形参类型 形参说明
   10. @return 的格式要求：@return 返回值类型 返回值说明
   11. @exception的格式要求：@exception 异常类型
   12. 异常说明 @param和@exception可以并列多个
4. 在编译时进行格式检查：
   1. @Override: 限定重写父类方法, 该注解只能用于方法
   2. @Deprecated: 用于表示所修饰的元素(类, 方法等)已过时。通常是因为所修饰的结构危险或存在更好的选择
   3. @SuppressWarnings: 抑制编译器警告
5. 自定义注解：
   1. 使用 @interface 关键字
   2. 自定义注解自动继承了java.lang.annotation.Annotation接口
   3. Annotation 的成员变量在 Annotation 定义中以无参数方法的形式来声明。其 方法名和返回值定义了该成员的名字和类型。我们称为配置参数。类型只能 是八种基本数据类型
   4. 指定成员变量的初始 值可使用 default 关键字
   5. 如果定义的注解含有配置参数，那么使用时必须指定参数值，除非它有默认值
   6. 自定义注解必须配上注解的信息处理流程才有意义。
6. 元注解：
   1. @Retention: 只能用于修饰一个 Annotation 定义, 用于指定该 Annotation 的生命 周期, @Rentention 包含一个 RetentionPolicy 类型的成员变量, 使用 @Rentention 时必须为该 value 成员变量指定值
   2. 
   3. @Target: 用于修饰 Annotation 定义, 用于指定被修饰的 Annotation 能用于 修饰哪些程序元素。 @Target 也包含一个名为 value 的成员变量。
   4. 
   5. @Documented: 用于指定被该元 Annotation 修饰的 Annotation 类将被 javadoc 工具提取成文档。默认情况下，javadoc是不包括注解的。
   6. 
   7. @Inherited: 被它修饰的 Annotation 将具有继承性。如果某个类使用了被 @Inherited 修饰的 Annotation, 则其子类将自动具有该注解。
   8. 
7. 可重复注解：
   1. 可重复的注解及可用于类型的注解
   2. 
   3. TYPE\_PARAMETER,TYPE\_USE：
      1. ElementType.TYPE\_PARAMETER 表示该注解能写在类型变量的声明语 句中（如：泛型声明）
      2. ElementType.TYPE\_USE 表示该注解能写在使用类型的任何语句中。
      3. 