Java 注解：

1. 注解是一个特殊的接口 public intaface @face {}
2. 注解中的方法，在实际用时，是作为一个参数的

注解的创建，注解的使用

@Target({ElementType.TYPE})

@Retention(RetentionPolicy.SOURCE)

public intaface @Face{

public string name();

public string age();

Boolean sex() default true; 可以设置默认值

Class<?>[] application(); 数组类型的注解

}

@face(name=”zhu”,age=”11”,application={String.class,Integer.class})

Public class student {}

1. 注解在反射时的获取：

Face face=student.class.getAnnotation()

face.name();

face.age();

可插拔式的注解处理方式：

相关的处理包：java .annotation.processing javax.lang.model包

Processing负责正真的处理，在处理的过程中，会需要程序的原信息，这是要用到model包

处理分多伦进行，本次的输出是下一次的输入

简单实现：



这两个注解的作用

@SupportedSourceVersion(SourceVersion.***RELEASE\_7***)

充当可以处理的最高的版本号（原本是要写在方法中的）getSupportedSourceVersion()方法

@SupportedAnnotationTypes("com.java7book.chapter8.annotation.Author")

同上 充当：get SupportedAnnotationTypes() 可以处理的直接类型

注意：

如果获得的直接类型是class 通过element.getAnnotationMirrors()方法获取的class可以使用，因为已经经过了类加载等处理，而在通过element.getAnnotation()获得的class无法使用，没有经过类加载。

注解处理器的使用，编译器：

（1）javac –processo 来制定处理器类。

（2）注解处理器的自动发现机制：将注解处理器配置在METAINF/services/javax.annotation.processing.Processor中（文件中的每一行表示一个注解处理器）

通过 –processorpath进行上述路径的配置 可以通过 –proc：none禁用注解处理

注解处理器创建新文件：

（编译器级别的）

创建新文件的方法由processingEnvironment接口得到的javax.annotation.processing.Filer接口的实现完成。

对已有的源代码进行修改则有编译器的API完成。

如何生成一个Java源文件：



生成的方式类似于编译器api中的生成方式（JavaFileObject对象）

对已有的代码进行修改：

通过OPENJDK的编译器API进行修改（将所有的域声明变为public）

（原理同上，都是通过注解处理器实现的-------编译器级别的）



通过反射API来进行annotation的处理：

反射API用来获得注解的相关信息，注解用来规定具体的行为，而动态代理则用来实现具体的细节

