注解

1. 注解分类：
2. 源码注解：注解只在源码中存在，编译成.class文件就不存在了。
3. 编译时注解：在源码和.class文件中都存在。
4. 运行时注解：在运行阶段还会起作用，甚至会影响运行逻辑的注解，可以通过反射读取。
5. 自定义注解
6. 成员类型的写法

String author();//成员类型是受限制的，合法的类型包括原始类型及//String,Class,Annotation,Enumeration

Int age() default 18;

1. 如果只有一个成员，成员名必须为value(),在使用的时候可以忽略成员名和赋值号(=)。
2. 注解类可以没有成员，称为标识注解。

三. 元注解

a) @Target({EleementType.METHOD,ElementType.TYPE})

作用域：构造方法声明、字段声明、局部变量声明、方法声明、包声明、参数声明、类、接口

b) @Reteention(RetentionPolicy.RUNTIME) 声明周期注解

说明：标识注解是源码注解、编译注解、运行时注解。

1. @Inherited 标识型注解，允许子注解继承它。
2. @Documented 生成javadoc时会包含注解。

四. 解析注解

概念：通过反射获取类、函数或成员上的运行时注解信息，从而实现动态控制程序运行的逻辑。

1. 使用类加载器Class c=Class.forName(“com.xxx.xxx”)
2. 判断是否是类注解boolean isExit=c.isAnnotationPresent(Description.class)
3. 拿到注解实例 Description d=( Description)c.getAnnotation(Description.class);
4. 找到方法上的注解 Method[] ms=c.getMethods();
5. 判断方法是否使用该注解Boolean isExist=ms[0].isAnnotationPresent(Description.class);
6. 拿到注解实例 Description d=( Description) ms[0].getAnnotation(Description.class);

第二种解析方法：(拿到一个方法上的所有注解)

Annotation[] as=ms[0].getAnnotations();