1、注解：Java代码提供元数据，描述数据的数据(标签)；

2、基本内置注解：

(1)@Override:对覆盖超类中方法的方法进行标记，如果被标记的方法并没有实际覆盖超类中的方法，则编译器会发出错误警告。

(2)@Deprecated:过时代码注解，若使用，则警告；

(3)@SuppressWarnings:去处警告,可传参；

3、元注解：注解到注解上的注解,主要是应用在注解声明；

(1)@Retention:保留期，注解存活时间；

- RetentionPolicy.SOURCE 注解只在源码阶段保留，在编译器进行编译时它将被丢弃忽视。

- RetentionPolicy.CLASS 注解只被保留到编译进行的时候，它并不会被加载到 JVM 中。

- RetentionPolicy.RUNTIME 注解可以保留到程序运行的时候，它会被加载进入到 JVM 中，所以在程序运行时可以获取到它们。

(2)@Documented 将注解中的元素包含到 Javadoc 中去;

(3)@Target 指定应用场景

参数为枚举ElementType。若应用错误，编译报错；

(4)@Inherited 继承 ，若超类的注解声明包含，则子类默认继承超类的的该注解；

(5)@Repeatable 可重复注解。jdk1.8开始使用 其余1.5后

4、注解的属性：

成员变量。只有变量，没有方法。通过实例来说明。

5、java预置注解

(1)@Deprecated 过时代码(方法、类)；

(2)@Override

(3)@SuppressWarnings

(4)@SafeVarargs

(5)@FunctionalInterface 函数式接口，jdk1.8后加入；

6、lambda表达式

通过表达式来代替功能接口。和方法一样,它提供了一个正常的参数列表和一个使用这些参数的主体(body,可以是一个表达式或一个代码块。

语法：

// 1. 不需要参数,返回值为 5

() -> 5

// 2. 接收一个参数(数字类型),返回其2倍的值

x -> 2 \* x

// 3. 接受2个参数(数字),并返回他们的差值

(x, y) -> x – y

// 4. 接收2个int型整数,返回他们的和

(int x, int y) -> x + y

// 5. 接受一个 string 对象,并在控制台打印,不返回任何值(看起来像是返回void)

(String s) -> System.out.print(s)

具体例子，详看：LambdasTest.java

7、函数式接口

Java 8对一类特殊类型的接口的称呼。 这类接口只定义了唯一的抽象方法的接口（除了隐含的Object对象的公共方法）。

用例详见FunctionInterfaceTest.java

java.lang.Runnable,

java.awt.event.ActionListener,

java.util.Comparator,

java.util.concurrent.Callable

(1)函数式接口声明，详见.java；