

十、注解(Annotation)/元数据(Metadata)

一、介绍:

- Annotation使用时要在前面增加一个@符号,并把Annotation当成符号使用,用于修饰它支持的程序元素
- **三个基本元素Annotation:**
 1. @Override:该注解只能用于方法,比如重写父类方法
 2. @Deprecated:用于表示某些已经过失的程序元素(类,方法..)
 3. @SuppressWarnings:抑制编译器警告(Suppress镇压)

@Override:

```
package com.Annotation;

public class Override_ {
    public static void main(String[] args) {

    }
}

class Father{
    public void fly(){
        System.out.println("father fly..");
    }
}

class son extends Father{
    /*解析:
    * 1.@Override 注解放在fly()上方,表示子类的fly方法重写了父类的fly方法
    * 2.这里若没有写 @Override 还是重写了父类fly()
    * 3.若写了@Override注解,编译器会检查方法是否重写了父类的方法,
    * 若的确重写了则编译通过,若没有重新,则编译错误*/
    @Override
    public void fly() {
        System.out.println("son fly...");
    }
}
```

➤ 补充说明: @interface 的说明

@interface 不是interface, 是注解类 是jdk5.0之后加入的

➤ Override 使用说明

1. @Override 表示指定重写父类的方法 (从编译层面验证), 如果父类没有fly方法, 则会报错
2. 如果不写@Override 注解, 而父类仍有 public void fly(){}, 仍然构成重写
3. @Override 只能修饰方法, 不能修饰其它类, 包, 属性等等
4. 查看@Override注解源码为 @Target(ElementType.METHOD),说明只能修饰方法
5. @Target 是修饰注解的注解, 称为元注解

@Deprecated:

```
generation02.java x EnumMethod.java x Excer01.java x Override_.java x Deprecated_.java x
public class deprecated_ {
    public static void main(String[] args) {
        A a = new A(); //有个中划线, 表示过时了, 但可以用
        a.hi();
        System.out.println(a.n1);
    }
}

/*解析:
* 1. 修饰某个元素(类), 表示已经过时了
* 2. 不推荐使用, 但可以用
* 3. 可以修饰: 方法, 类, 字段, 包, 参数等等
* 4. 可以作为升级过度的使用: 比如:
*   jdk8有个A类---要升级成--->jdk11的B类
*   就要将jdk8的A类标注过时, 最好使用jdk11的B类
* */

@Deprecated
class A{
    @Deprecated
    public int n1 = 10;
    @Deprecated
    public void hi(){}
```

基本的 Annotation 应用案例

@Deprecated 的说明

1. 用于表示某个程序元素(类, 方法等)已过时
2. 可以修饰方法, 类, 字段, 包, 参数 等等
3. @Target(value={CONSTRUCTOR, FIELD, LOCAL_VARIABLE, METHOD, PACKAGE, PARAMETER, TYPE})
4. @Deprecated 的作用可以做到新旧版本的兼容和过渡

@SuppressWarnings:

● @SuppressWarnings 注解的案例

➤ 说明各种值

- 1) unchecked 是忽略没有检查的警告
- 2) rawtypes 是忽略没有指定泛型的警告(传参时没有指定泛型的警告错误)
- 3) unused 是忽略没有使用某个变量的警告错误
- 4) @SuppressWarnings 可以修饰的程序元素为, 查看@Target
- 5) 生成@SupperssWarnings 时, 不用背, 直接点击左侧的黄色提示, 就可以选择(注意可以指定生成的位置)

```
package com.Annotation;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.List;
public class SuppressWarnings_ {
    /*解析:
    * 1. 若不希望看到警告可以加@SuppressWarnings来抑制警告信息
    * 2. 在{"all"}中写入你希望抑制的警告
    * 3. @SuppressWarnings({"all"})的作用范围和其放置的位置有关*/
    @SuppressWarnings({"all"})
    public static void main(String[] args) {
        //内容为集合:后面会讲
        List list = new ArrayList();
        list.add("");
        list.add("");
        list.add("");
        int i;
        System.out.println(list.get(1));
    }
}
```

二、JDK元注解(了解)

● 元注解的基本介绍

JDK 的元 Annotation 用于修饰其他 Annotation

元注解：本身作用不大，讲这个原因希望同学们，看源码时，可以知道他是干什么。

● 元注解的种类 (使用不多，了解，不用深入研究)

- 1) Retention //指定注解的作用范围，三种 SOURCE, CLASS, RUNTIME
- 2) Target // 指定注解可以在哪些地方使用
- 3) Documented //指定该注解是否会在javadoc体现
- 4) Inherited //子类会继承父类注解

韩顺平
教育

> @Retention的三种值

- 1) RetentionPolicy.SOURCE: 编译器使用后，直接丢弃这种策略的注释
- 2) RetentionPolicy.CLASS: 编译器将把注解记录在 class 文件中。当运行 Java 程序时，JVM 不会保留注解。这是默认值
- 3) RetentionPolicy.RUNTIME: 编译器将把注解记录在 class 文件中。当运行 Java 程序时，JVM 会保留注解。程序可以通过反射获取该注解

