

Java 元注解@Retention 规则

@Retention 是 java 当中的一个元注解，该元注解通常都是用于对软件的测试

1、适用方式:

```
@Retention(RetentionPolicy.RUNTIME)
```

```
@interface Task{.....}
```

参数 RetentionPolicy.RUNTIME 就说明了，@Task 注解在程序运行时是可见的。

RetentionPolicy 的枚举类型还有 SOURCE、CLASS 分别指定注解对于那个级别是可见的，但是我们一般都是用 RUNTIME，因为这是在程序运行时可以对注解进行读取，从而易于软件的测试。

2、接下来我们要先介绍一下 java 程序的内省和反射机制，之后在讨论 @Retention 的具体用法实例。

在 java 虚拟机 JVM 在运行时，就会将类进行加载，这时，每个类都会生成一个 Class 数据类型的对象（Class 类在 java.lang.Class 中），这个对象就是对应类的“运行时对象”，通过这个运行时对象，就能够获取对应类的许多信息，也就是说，运行时对象实际就是对应类的一个映射。

这就是 java 的内省反射机制。

3、接下来我们讨论一下，这个 Class 运行时对象的使用

① 获取对应类的 Class 数据类型的运行时对象的引用—getClass()

```
public class Point{.....} //声明一个类
Point pt = new Point(); //创建对应类的实例对象
Class cls = pt.getClass(); //则 cls 就指向了 Point 类的运行时对象
```

② 运行时对象 cls 的成员函数

<1> public String getName() 返回对应类的类名
<2> public boolean isAnnotationPresent(注解名.class)

判定指定的“注解”是否在运行时注解了 cls 的对应类

<3> public boolean isAnnotation();

判定 cls 是否在运行时被任何注解 注解过

<4> public A getAnnotation(注解名.class)

A 指的是一个注解的类型，具体用法如下：

```
@Retention(RetentionPolicy.RUNTIME) //指定@Task 运行时可见
```

```
@interface Task{String description(); }
```

```
@Task(description="NotFinished") //为 computer 作注
```

```
class Computer{.....}
```

则 Computer my = new Computer() ;

```
Class cls = my.getClass() ;
```

```
Task tk = (Task) cls.getAnnotation(Task.class);  
//这时 tk 就指向了标注 Computer 的注解@Task  
tk.description(); //调用@Task 中的 description(),输出  
"NotFinished"
```

<5> public Method[] getMethods()

返回由对应类中的所有的方法形成的Method数组，每个Method对象都唯一对应一个对应类中的方法，通过 Method[i]就可以获得对应方法的信息（Method 类在 java.lang.reflect.Method 中）。

这个 Method 类也有很多成员方法，用来获取对应的方法的信息。 比如有：

```
public boolean isAnnotationPresent(注解名.class)
```

判定对应的方法是否被指定的注解所注解

```
public A getAnnotation(注解名.class)
```

用法和上面的讲述的一样，之不过创建的注解型的引用变量指向的是 "标记对应方法的注解"

4、上面将所有的成员方法只有在注解运行时可见的情况下才能够发挥作用，所以@Retention 变得很有用。

@Retention 的用法实例：

```
package mypackage1;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.reflect.Method;  
import java.lang.SuppressWarnings;  
  
@Retention(RetentionPolicy.RUNTIME)  
@interface WorkInProgress{ }  
  
@Retention(RetentionPolicy.RUNTIME)  
@interface Task  
{  
    String description();  
}  
  
@SuppressWarnings({"unchecked","rawtypes"})  
@WorkInProgress  
public class Count  
{  
    @WorkInProgress  
    @Task(description="Implement tax computations")  
    public static float ComputeTax(){ return 0 ;}  
  
    public static void main(String [] args)  
    {
```

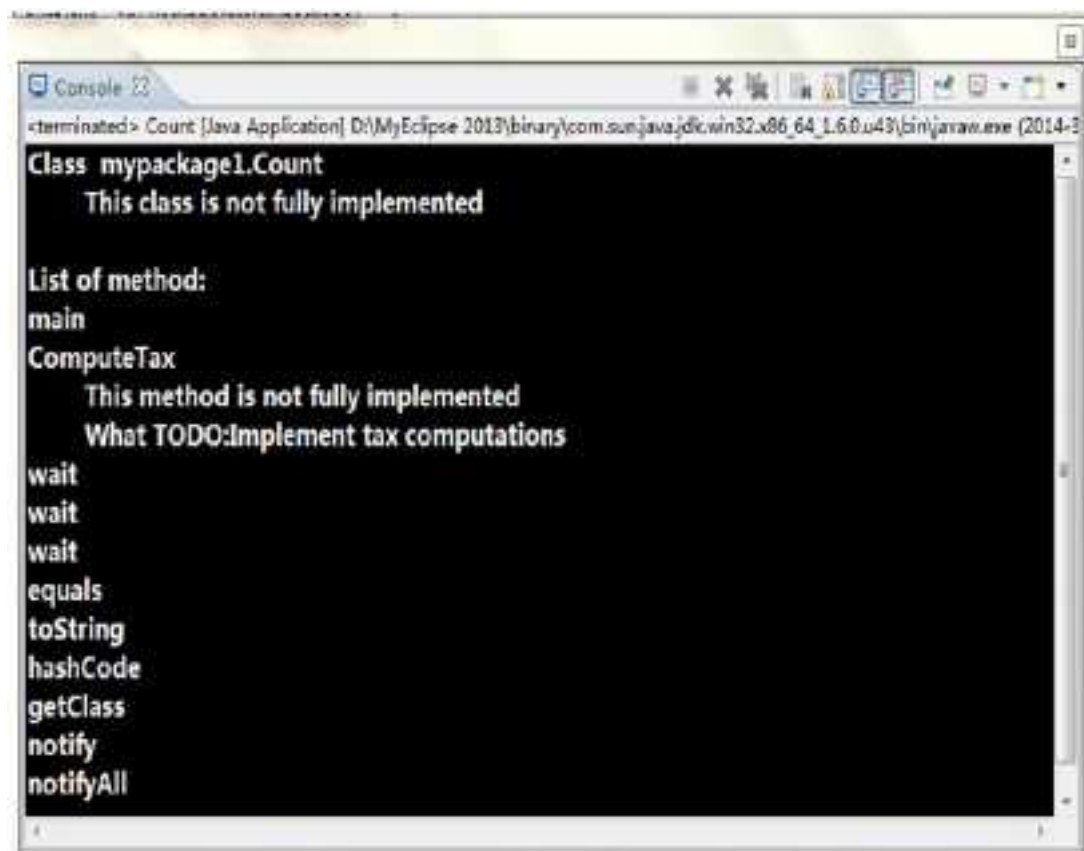
```

public static void main(String [] args)
{
    try{
        Count obj = new Count();
        Class cls = obj.getClass();
        System.out.println("Class " + cls.getName());
        if(cls.isAnnotationPresent(WorkInProgress.class))
        {
            System.out.println("\tThis class is not fully implemented");
        }
        System.out.println("\nList of method:");
        Method[] methods = cls.getMethods();
        for(Method method : methods){
            System.out.println(method.getName());
            if(method.isAnnotationPresent(WorkInProgress.class))
            {
                System.out.println("\tThis method is not fully implemented");

                if(method.isAnnotationPresent(Task.class))
                {
                    Task annotationTask =
                        (Task)method.getAnnotation(Task.class);
                    System.out.println("\tWhat TODO:"
                        +annotationTask.description());
                }
            }
        }
    }
    catch(Exception e){
        System.out.println(e.getMessage());
    }
}
}

```

运行结果：



```
<terminated> Count [Java Application] D:\MyEclipse 2013\binary\com.sun.java.jdk.win32.x86_64_1.6.0.u43\bin\java.exe (2014-3-11 14:58:58)
Class mypackage1.Count
    This class is not fully implemented

List of method:
main
ComputeTax
    This method is not fully implemented
    What TODO:Implement tax computations
wait
wait
wait
equals
toString
hashCode
getClass
notify
notifyAll
```