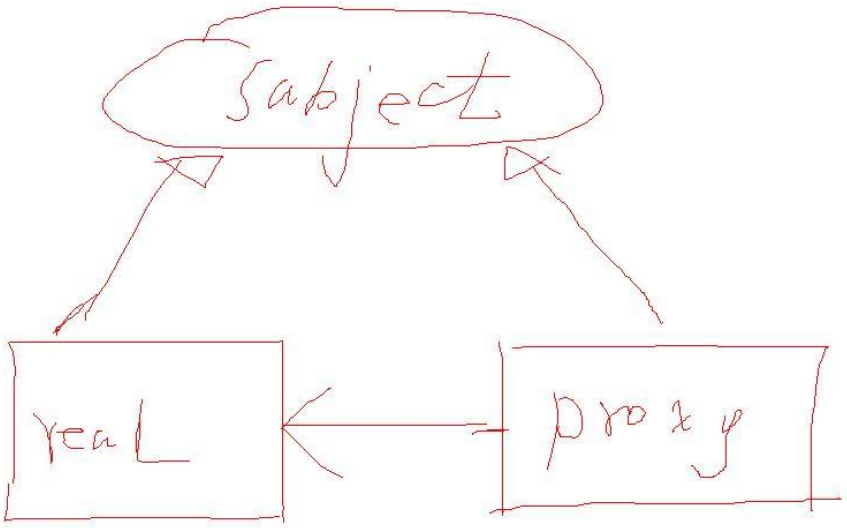
Java SE Lesson 10

1. 静态代理模式图示



2. Java 注解（Annotation）：

a) Override 注解表示子类要重写（override）父类的对应方法。

b) Deprecated 注解表示方法是不建议被使用的。

c) SuppressWarnings 注解表示抑制警告。

3. 自定义注解：当注解中的属性名为 value 时，在对其赋值时可以不指定属性的名称而直接写上属性值即可；除了 value 以外的其他值都需要使用 name=value 这种赋值方式，即明确指定给谁赋值。

4. 当我们使用 @interface 关键字定义一个注解时，该注解隐含地继承了 java.lang.annotation.Annotation 接口；如果我们定义了一个接口，并且让该接口继承自 Annotation，那么我们所定义的接口依然还是接口而不是注解；Annotation 本身是接口而不是注解。可以与 Enum 类比。

5. JUnit（3.8、4.x）：Keep the bar green to keep the code clean.

6. 我的名言：没有反射，很多框架就不存在了。（No Reflection，No most frameworks）。

7. JUnit4 的执行的一般流程：

a) 首先获得待测试类所对应的 Class 对象。

b) 然后通过该 Class 对象获得当前类中所有 public 方法所对应的 Method 数组。

c) 遍历该 Method 数组，取得每一个 Method 对象

d) 调用每个 Method 对象的 isAnnotationPresent(Test.class)方法，判断该方法是否被 Test 注解所修饰。

e) 如果该方法返回 true，那么调用 method.invoke()方法去执行该方法，否则不执行。

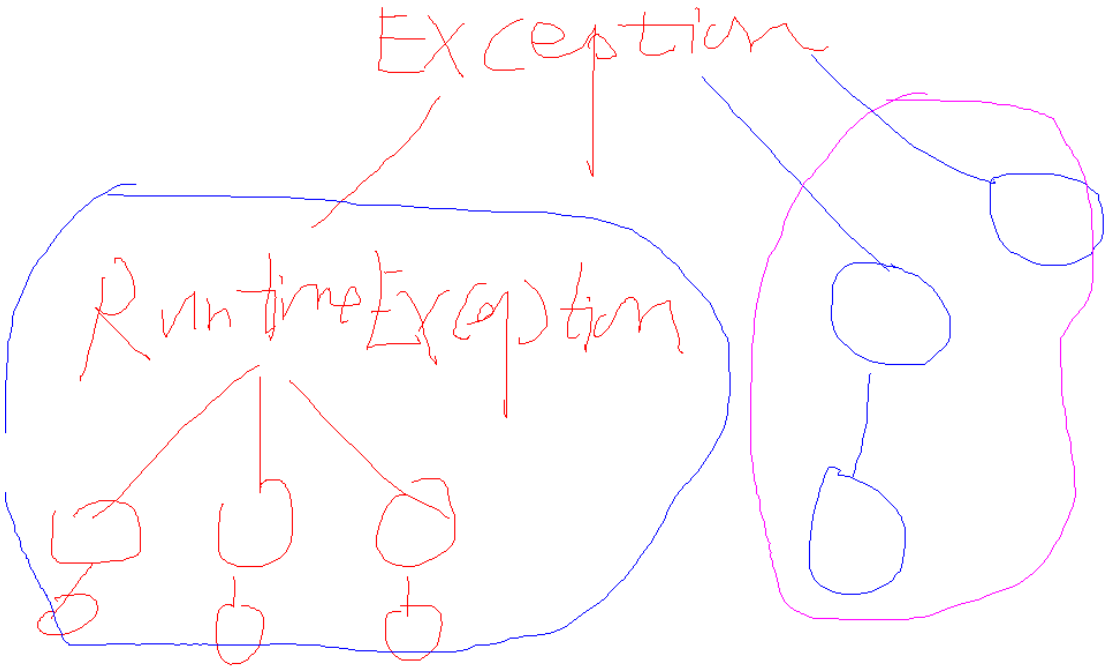
8. **单元测试不是为了证明你是对的，而是证明你没有错误。**

9. Writing Secure Code（编写安全的代码）：Input is evil。

10. 异常（Exception）。
11. Java 中的异常分为两大类：

a) Checked exception （非 Runtime Exception）
b) Unchecked exception（Runtime Exception）

12. Java 中所有的异常类都会直接或间接地继承自 Exception。
13. **RuntimeException 类也是直接继承自 Exception 类，它叫做运行时异常，Java 中所有的运行时异常都会直接或间接地继承自 RuntimeException。**
14. Java 中凡是继承自 Exception 而不是继承自 RuntimeException 的类都是非运行时异常。



15. 异常处理的一般结构是：

```
try
{
}
catch(Exception e)
{
}
finally
{
}
```

无论程序是否出现异常，finally 块中的代码都是会被执行的。

16. 对于非运行时异常（checked exception），必须要对其进行处理，处理方式有两种：第一种是使用 try.. catch…finally 进行捕获；第二种是在调用该会产生异常的方法所在的方法声明 throws Exception
17. 对于运行时异常（runtime exception），我们可以不对其进行处理，也可以对其进行处理。推荐不对其进行处理。
18. NullPointerException 是空指针异常，出现该异常的原因在于某个引用为 null，但你

却调用了它的某个方法。这时就会出现该异常。