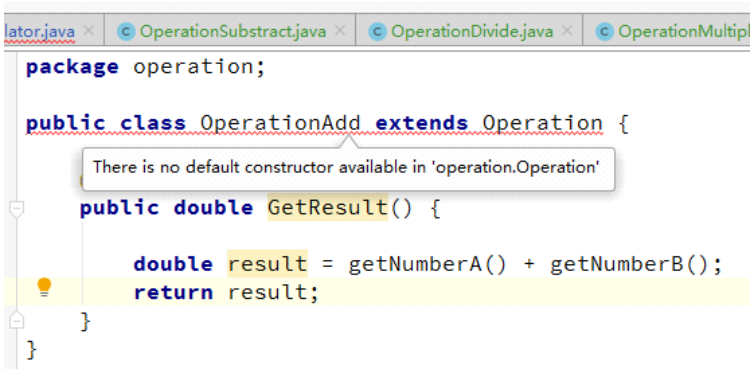
DP

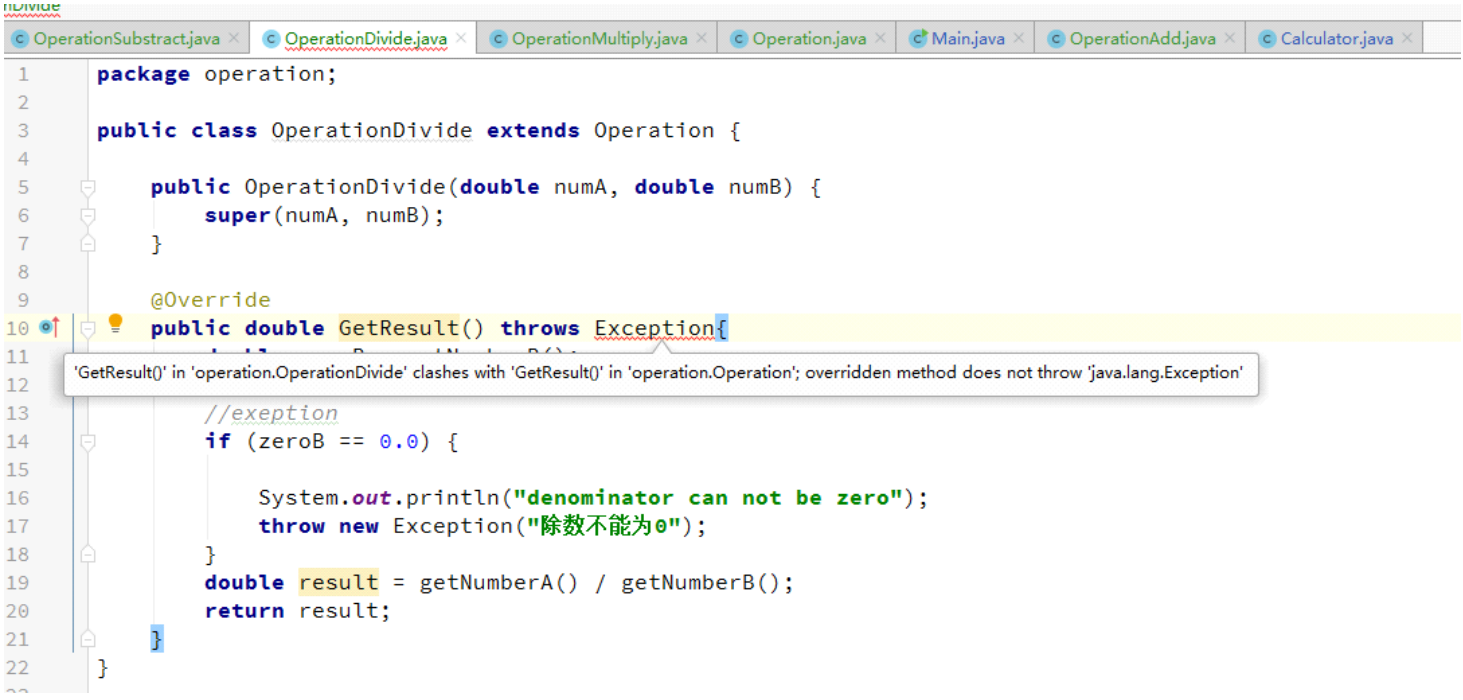
2019年11月26日 23:08

1.There is no default constructor available in xxx



若父类显示声明了带参数的构造函数，则默认构造函数自动失效。
解决办法是在父类中显示声明默认构造函数，或者在子类中显示声明默认构造函数。

2.overridden method does not throw exception



https://stackoverflow.com/questions/23526362/overridden-method-does-not-throw-exception

推荐： https://www.geeksforgeeks.org/exception-handling-with-method-overriding-in-java/

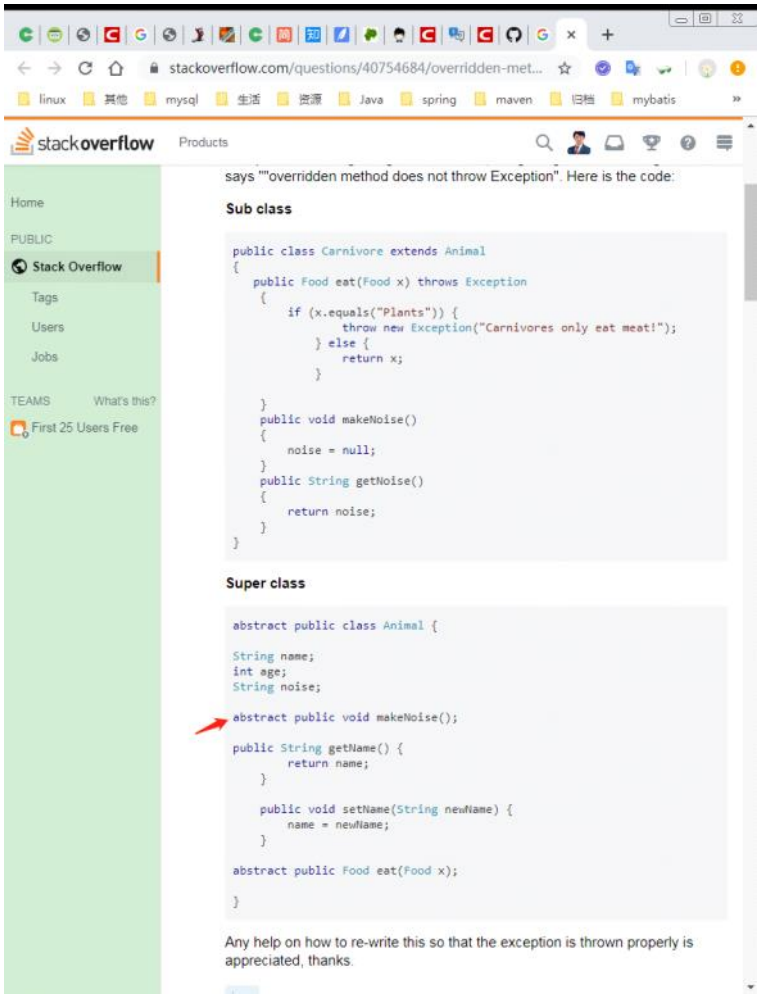
Conclusion for Handling such Exceptions: Hence, following conclusions can be derived from the above examples:

1. If SuperClass does not declare an exception, then the SubClass can only declare unchecked exceptions, but not the checked exceptions.
2. If SuperClass declares an exception, then the SubClass can only declare the child exceptions of the exception declared by the SuperClass, but not any other exception.
3. If SuperClass declares an exception, then the SubClass can declare without exception.

结论：
1.如果父类未声明异常，则子类不能声明检查异常，只能声明未检查异常。
2.如果父类声明了异常，则子类只能声明父类异常的子异常。
3.如果父类声明了异常，则子类可以不声明异常。

3.abstract

https://www.geeksforgeeks.org/abstraction-in-java-2/



1）用abstract关键字修饰的类是抽象类

2）抽象方法没有方法体

3）抽象类中的方法可以是抽象方法，也可以是普通方法。

4）父类中的抽象方法必须在子类中重写并实现方法体，或者将子类也定义为抽象类。

5）有抽象方法的类必须以abstract关键字修饰。

6）抽象类不能被实现。

7）抽象类可以有带参数的构造函数，并且默认构造函数。。。（and default constructor is always present in an abstract class）

（没懂）默认构造函数在抽象类中无法被隐藏？

封装 VS 抽象

1）封装是对数据的隐藏，而抽象是对实现的隐藏。

2）封装将数据和依赖这些数据的方法聚合在一起，而抽象是将接口暴露给用户并隐藏实现的细节。

抽象的好处

1）降低事物的复杂性。

2）降低代码重复，提高重用性。

3）通过仅向开发人员提供重要细节，来帮助提高应用或程序的安全性。

4.接口interface

https://www.geeksforgeeks.org/interfaces-in-java/

1）接口指明类什么必须做，而不是怎么做。接口是类的蓝图。

2）接口是关于能力的。比如一个叫Player的接口，任何实现Player接口的类都必须实现move()方法。

即，接口指定了一些类必须实现的方法。

3）如果一个类实现了一个接口，但没有给出接口中所有方法的方法体，则该类必须声明为抽象类。

4）一个Java的例子：比较器接口。如果一个类实现了该接口，则该类可以给集合排序。

接口中的所有方法都没有方法体且为public，且所有域默认为public,static,final。

实现接口的类必须实现接口中声明的所有方法。

为什么要用接口：
1）接口常常用来实现完全的抽象。
2）java的类不支持多继承，但是可以通过接口来实现多继承。
3）可以实现松耦合。
4）接口常常用来实现抽象，那么既然我们有抽象类，为什么还要用接口呢？
原因是，抽象类可以包含非final的变量，而接口中的变量是final,public和static的。

JDK8 中的新特性：
1）假设我们要在接口中添加一个新功能，很显然旧代码就不能用了，因为实现这个接口的类
没有实现这个新功能。
在JDK8 中，可以在接口的方法中添加默认方法体，可以解决上述问题。
2）可以在接口中定义静态方法了。
接口中的静态方法可以通过“类名.方法名”来调用，而不用先实例化。
这种方法不可遗传，不可被继承（？）

接口总结：
1）接口不能被实例化，但可以实例化实现该接口的类。
2）类可以实现多个接口。
3）一个接口可以继承其他的一个或多个接口。
4）一个类实现了一个接口，那么该类要实现该接口的所有方法。
5）接口中的方法都是pubcli和abstract的，且所有的域都是public，static和final的。
6）接口常用来实现多继承。
7）接口常用来实现松耦合。

5.抽象类和接口的区别？
https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/
抽象类和接口都是用来达到抽象的目的的。
1）方法的类型
接口只能有abstract方法。抽象类可以有abstract和非abstract方法，Java 8以后可以有默认和static的方法。
2）final类型的变量
接口中的变量默认为final，抽象类中的变量可以是非final的。
3）变量的类型
抽象类可以包含final,non-final,static,non-static的变量，接口只能有final,static的变量。
4）实现
抽象类可以提供接口的实现。接口无法提供抽象类的实现。（不懂）
5）继承和抽象
实现接口时用implements关键字，继承抽象类时用extends关键字。
6）多继承
接口只能继承其他接口，抽象类可以继承其他类和多个接口。
7）访问权限
接口中的成员默认为pubcli，抽象类则可以包含private,protected等类型。

什么时候用什么？

以下情况考虑用抽象类
1）当一些关联的类有类似的代码时，可以把类似的代码封装成抽象类，
并让这些类来继承这个抽象类
2）在抽象类中，可以定义non-static或者non-final的域，这样可以通过方法更改对象的状态
3）当一些类有很多共同之处，并且需要修改protected,private等除了public之外的域

以下情况考虑用接口
1）完全抽象，实现接口的类必须实现接口的所有方法
2）一个类需要多继承时
3）你想指定特定数据类型的表现，而不关心谁去实现。