JAVA DAY 14

1、异常概念

程序非正常的执行流程

在Java中，把异常信息封装成了一个类。

当出现问题时，就会创建异常类对象并抛出异常相关的信息（名称，位置，原因等）异常抛出、异常捕获、异常处理。

常见异常：

数组下标越界：ArrayIndexOutOfBoundsException

空指针（NPE）：NullPointerException

产异常对象，JVM将其产生的异常抛给调用者main()方法。

main方法接收了异常对象

main()方法没有处理异常的话，继续把异常抛给调用者JVM。JVM收到异常后，将异常的名称、内容、位置显示在控制台上，并将程序终止。

2、异常的处理

No.

Date

a) JVM的默认处理方式

把异常的名称、原因、位置等信息输出在控制台，同时会结束程序。

一旦有异常发生，其后来的代码不能继续执行。

b) 解决程序中异常的方式

编写处理代码 try...catch...finally

向上抛出 throws

JVM → main() → a() → b()

c) 语法

异常 try...catch...finally 基本使用

```
try{
    可能产生异常的代码
}catch(异常类型1){
    //
}catch(异常类型2){
    //
}finally{
    不管是否发生异常，都要执行
}
```

throw：抛出一个指定的异常对象

throw new xxxException(param...);

修饰符 返回值类型 方法(参数) throws 异常类1,2,{...}

d) 异常相关方法

java.lang.Exception 异常类

getMessage()：返回该异常的详细信息字符串，即异常提示信息。

toString()：返回该异常的名称与详细信息字符串

printStrackTrace()：在控制台输出该异常的名称与详细信息字符串、异常出现的代码位置。

e) 异常使用注意

1) 捕获异常要小后大（先子类后父类）

2) 可以用 catch(Exception e){} 匹配默认异常处理分支，放在最后

3) try...finally... 没有 catch 是可以的

4) 尽量不要在 try 中用 return 语句

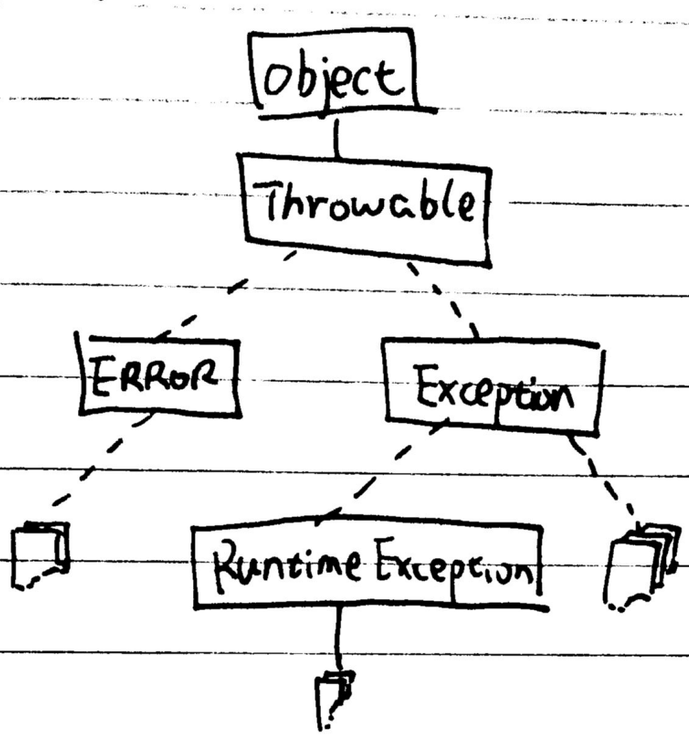
5) finally 中的代码总会被执行（除非遇到 System.exit();）经常用来释放资源

6) 程序带着某个返回条件进行 finally，如

果finally中修改了返回条件，以finally为准

7) 子类重写方法抛出的异常是父类方法抛出异常的交集

3. 异常的分类



Error：应用程序无法处理。

检查型异常：必须自己写代码进行处理

Exception子类，但是RuntimeException及其子类

1) try...catch...finally 在本方法中进行处理

2) throws 抛给调用者进行处理

非检查型异常(运行时异常)：无需写代码进行处理

要么是 RuntimeException，要么是其子类

No.

Date

4. 自定义异常

继承 java.lang.Exception，通过名字区别不同的异常，不需要重写任何方法；属于检查型异常

5. 题：

1. throw 与 throws 比较？

   1) ？

   2) ？

   3) ？

2. final，finally，finalize？

   1) ？

   2) ？

   3) ？

6. 断言 assert

测试时使用，当满足判断条件程序继续执行，否则报错退出

assert ~~except~~ expression(boolean): "errinfo"

需要通过 java -ea 开启断言功能.