

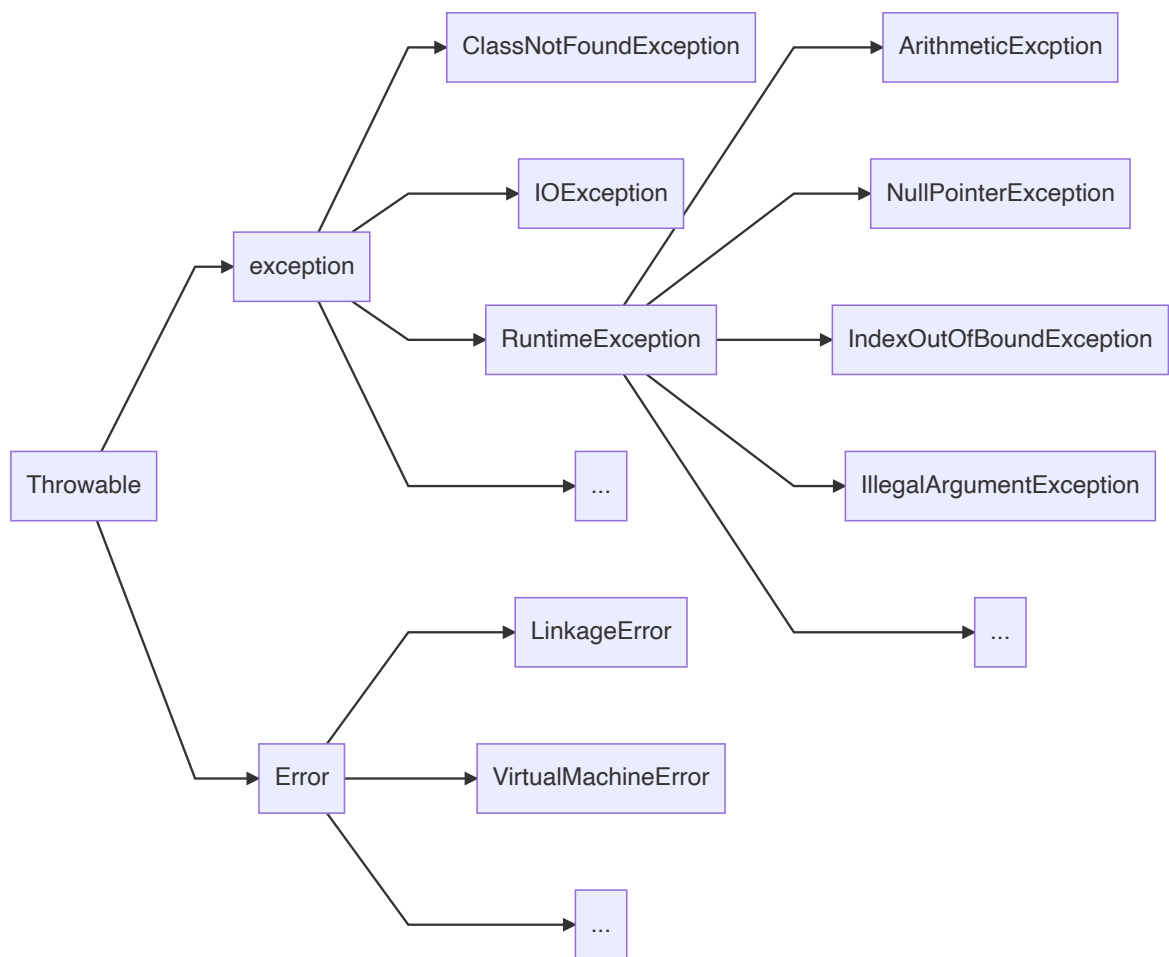
0617技术整理

Title: 0617-技术知识整理

author: 刘星烨

content: Java异常、IO

Java异常



基本认识

- Throwable下有Error和Exception。Error一般无法解决，由JVM引发，只能写提示信息。
- Error/RuntimeException是unchecked exception，不能用try-catch块处理。
- 对于exception，需要先declare再throw两步，被catch后进行处理，也可以继续throw。
- finally是一定会执行的，如果在catch中return，会先将返回值存入程序栈；rethrow也会在执

行finally后进行。

使用规范

- 异常catch后要进行处理，如果无法处理就不要catch。直接在函数定义中使用throws将抛出该异常。
- 不要catch Exception类一次捕获所有异常。
- finally块需要释放程序资源，但不能抛出异常。在try-catch块中抛出异常后会转至finally执行，如果此时抛出异常会导致前面的异常不能正确抛出。
- 抛出自定义异常时保留原始异常信息。
- 打印异常信息时带上异常堆栈。
- 对于多种异常情况应定义多种异常，不要使用异常返回值处理。
- 守护线程需要catch runtime exception，防止偶然错误导致线程结束。

使用实例

1. 在方法中抛出异常

```
public void method1() throws MyException{
    ...
    throw new MyException("Exception Information");
}
```

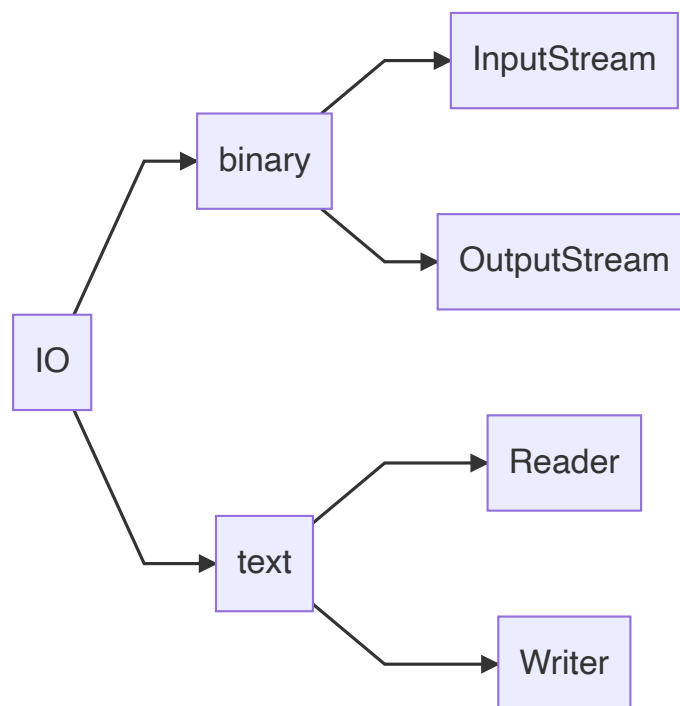
2. 对于exception的二次检测，关闭资源

```
Connection con;
try{
    con = dataSource.getConnection();
}
catch(SQLException e){
    throw e;
}
finally{
    try{
        if(con!=null) con.close();
        //处理本可能出现的NullPointerException
        ...
    }catch(SQLException e){
        ...
    }
}
```

3. catch顺序检测，本例子中只能catch到A

```
try{
    throw new B();
}catch(A a){...}
catch(B b){...}
class A extends Exception{}
class B extends A{}
```

IO



InputStream

- 抽象类，主要实现read()方法，读取字节流
- 使用 `try(resource)` 调用资源实现自动关闭

```
try (InputStream input = new FileInputStream("src/readme.txt")) {
    int n;
    while ((n = input.read()) != -1) {
        System.out.println(n);
    }
}
```

- `ByteArrayInputStream` 可以用来模拟字节流输入（测试用）

OutputStream

- 注意 `flush()` 方法的使用，需要立刻发送信息时手动flush
- 同样可以用 `try(resource)` 来实现资源自动关闭

Filter(detector)

主要是叠加包装的想法实现多种功能。

```
InputStream file = new FileInputStream("test.gz");
InputStream buffered = new BufferedInputStream(file);
InputStream gzip = new GZIPInputStream(buffered);
```

使用一个InputStream对象可以完成多种格式操作。

FilterInputStream/FilterOutputStream：对输入的字节流数据进行filter得到不同类型的数据。

Serialize

通过序列化后将对象存入文件或通过socket传输。

序列化：

```
try (ObjectOutputStream output = new ObjectOutputStream(buffer)) {
    // 写入int:
    output.writeInt(12345);
    // 写入String:
    output.writeUTF("Hello");
    // 写入Object:
    output.writeObject(Double.valueOf(123.456));
}
```

反序列化：

`readObject()`返回一个Java对象。

可能出现的异常：`ClassNotFoundException` `InvalidClassException`

解决方法 `serialVersionUID`：

```
public class Person implements Serializable {  
    private static final long serialVersionUID = 2709425275741743919L;  
}
```

Reader

读取时以char作为基本的读取单元。

- `FileReader` 实现文件字符流输入，需指定编码
- `CharArrayReader` 和 `StringReader` 可以在内存模拟字符流输入

Reader基于InputStream构造（本质上读入char等价于两个byte）

Writer

类似Reader

PrintStream

继承自OutputStream接口，是一种FilterOutputStream。

`System.out` 即位系统默认提供的PrintStream。`System.err` 为标准错误输出