【第十一章 异常和断言】

1、异常就是在程序运行的过程中所发生的不正常的事件，它会中断指令的正常执行。

2、异常分为两大类：

①Error:描述了java运行系统中的内部错误以及资源耗尽错误

唯一的解决办法：尽力使程序安全地终止

1. Exception：程序中需关注
2. 运行时异常（RuntimeException）：在java虚拟机正常运行期间抛出的异常，由程序错误导致。Java编译器允许程序中不对这类异常做出处理。

* 错误的类型转换
* 数组下标越界
* 访问空指针

1. 其他异常（如：IO异常、SQL异常）：一般是外部错误，Java编译器要求在程序中必须处理这类异常。

3、异常机制提供了程序退出的安全通道。当出现异常后，程序执行的流程发生改变，执行相应的异常处理代码。

4、Java中两种常见的异常处理机制：

①捕获异常（try-catch-finally）

若当前方法有能力处理异常，就捕获并处理它

②抛出异常（throw、throws）

若当前方法没有能力处理异常，则只需抛出异常，交由方法的调用者来处理

5、捕获异常：

1）使用try和catch捕获异常

try {

//接受监视的程序块,在此区域内发生的异常,

//由catch中指定的程序处理;

}catch(要处理的异常种类和标识符) {

//处理异常;

}

* try语句块中没有抛出任何异常
  + 程序会跳过catch子句
* try语句块中抛出了catch子句中说明的异常
  + 程序跳过try语句块中的其余代码
  + 程序执行catch子句中的异常处理代码

2）多重异常：

①

* 一段代码可能会生成多个异常。
* 当引发异常时，会按顺序来查看每个 catch 语句，并执行第一个类型与异常类型匹配的语句。
* 执行其中的一条 catch 语句之后，其他的 catch 语句将被忽略。
* 使用多重 catch 语句时，异常子类一定要位于异常父类之前。

例：try{

…….

} catch(ArrayIndexOutOfBoundsException e) {

……

} catch(Exception e) {

……

}

②

* 一个catch可以捕获多个异常。
* 多个异常类型之间用“|”分隔开。
* 只有1个异常类型的标识符。
* 多个异常类型之间不存在父子继承关系。

例：try {

i = a / b;

System.out.println("try block");

} catch (IndexOutOfBoundsException | ArithmeticException e1) {

System.out.println("发生异常，请处理该异常！");

}

3）try、catch使用注意事项：

①try语句块只能有一个，而catch语句块可以有任意多个

②catch语句块紧跟在try语句块之后

③建议对捕获的异常做适当的处理，而不仅仅是打印异常信息

1. finally语句块

① finally语句定义一个总是被执行的代码块，而不考虑是否出现异常

* 无论try、catch是否执行，finally必定执行

② 不执行finally语句块的特殊情况

* 在执行finally之前首先执行了“System.exit(0);”

③ finally语句块典型应用

* 回收资源

5）语法结构

try{

……

} catch(Exception e){

……

} finally{

//资源回收

}

6）try-catch-finally使用注意事项：

①无catch时finally必须紧跟try

②catch与finally不能同时省略

③try、catch和finally语句块之间不能插入任何代码（注释除外）

1. 异常处理的优势：

1）java采用异常类表示异常

- 把异常情况表示成异常类，可以充分发挥类的可扩展和可重用的优势。

2）异常流程的代码和正常流程的代码分离，提高了程序的可读性，简化了程序的结构

3）Java可以灵活的处理异常，如果当前方法有能力处理异常，就捕获并处理它，否则只需抛出异常，交由方法调用者来处理。

7、抛出异常：一个方法不处理它产生的异常,而是沿着调用层次向上传递，由调用它的方法来处理这些异常，叫抛出异常。

1）声明异常使用throws关键字

①throws是方法可能抛出异常的声明。

用在声明方法时，表示该方法可能要抛出异常。

②语法：[(修饰符)](返回值类型)(方法名)([参数列表])[throws(异常类型)]{......}

例：public void doA(int a) throws Exception1,Exception3{......}

2）抛出异常:

语法：throw (异常对象);

例：throw new ArithmeticException();

3）如何抛出异常？

- 确定要抛出的异常类

系统提供的异常类

自定义的异常类

- 创建这个类的对象

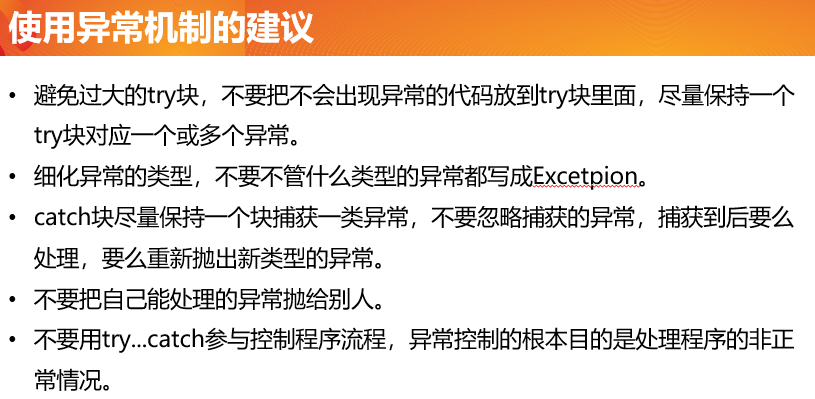
- 将该对象抛出

8、throw和throws的区别：

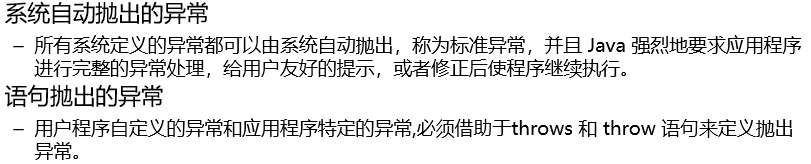
①throws语句用在方法声明后面，表示该方法会抛出哪些异常，使它的调用者知道要捕获这些异常。

②throw语句用在方法体内，表示抛出异常，是具体向外抛异常的动作，它抛出一个异常实例。

③throws表示出现异常的一种可能性，并不一定会发生这些异常；执行throw则一定抛出了某种异常。

9、

10、



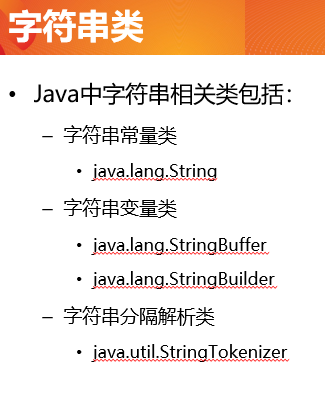
【第十二章 字符串】

1. 字符串：n个字符组成的序列

字符串常量：一对双引号“ ”定界起来的字符序列

空串：若两个双引号之间没有任何字符，则为空串

（null表示不存在，“”表示值为空）



1. String类：

1）String是字符串常量类

①String对象的值一经赋值，其值不可变

②指的是所指向的内存值不可修改，但可以改变指向

1. String类型变量的初始化
   1. 构造方法初始化

例：String name= new String(“zhangxiao”);

②字符串常量初始化

例：String sex = “男”;

1. String类是final的，无法被继承
2. String中常用的字符串处理方法：

字符串连接：①concat（String str）②“+”运算符

字符串查找：

indexOf (String str)返回第一次出现的索引值;

lastIndexOf(String str)返回最后一次出现的索引值[正向的索引值]

charAt(int indexOf)查找特定索引值的字符，返回字符

startsWith(String prefix)

字符串分割：

split(String regex)：字符串分割（特殊字符需要用"\\"进行转义）

compareTo(String str)：字符串比较 (比较ASCII码)

//compareTo()返回值为-1表示后者比前者大;

//返回值大于0表示前者比后者大;返回值为0表示两者一样大

equalslgnoreCase(String str)：忽略大小写

字符串替换：

replace(char oldChar, char newChar)

字符串求子串：

substring(int beginIndex, int endIndex)

//包含beginIndex，不包含endIndex

字符串大小写替换：

toUpperCase() 小写转大写

toLowerCase()大写转小写

1. StringBuffer类

1）StringBuffer类是一个字符串变量类

①StringBuffer对象的值可以修改

②主要用于对字符串做大量修改的操作时

2）StringBuffer变量的初始化

构造方法初始化：

例：StringBuffer stringBuffer = new StringBuffer(“Hello”);

1. StringBuffer类中常用的方法：

字符串连接：

append(Object obj)

字符串删除：

delete(int start,int end)

// 删除索引值为start至end的字符，包含start，不包含end

字符串插入：

insert(int offset,Object obj)

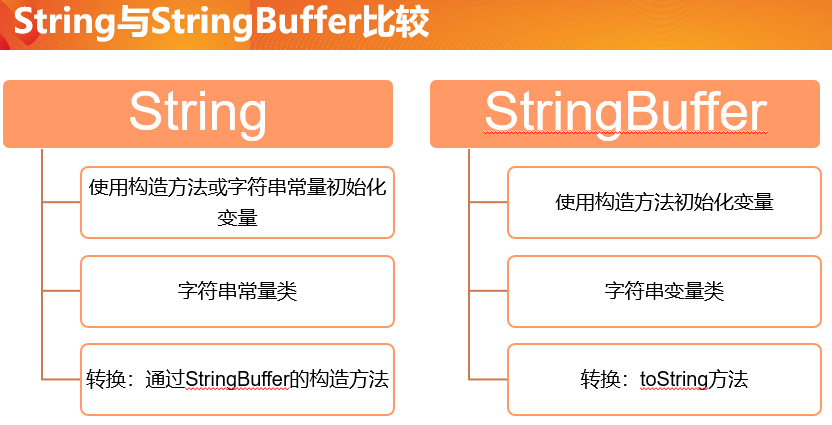
//在给定的索引值之处插入新的字符串

字符串逆序：

reverse()

转换为String对象：

toString()

7、

8、StringBuilder类与StringBuffer类的区别：

StringBuffer是线程安全的

StringBuilder是非线程安全的

9、