【济南中心】JAVA 编程阶梯：基础篇之第十九章

- **异常的概述**

异常就是 Java 程序在运行过程中出现的错误。

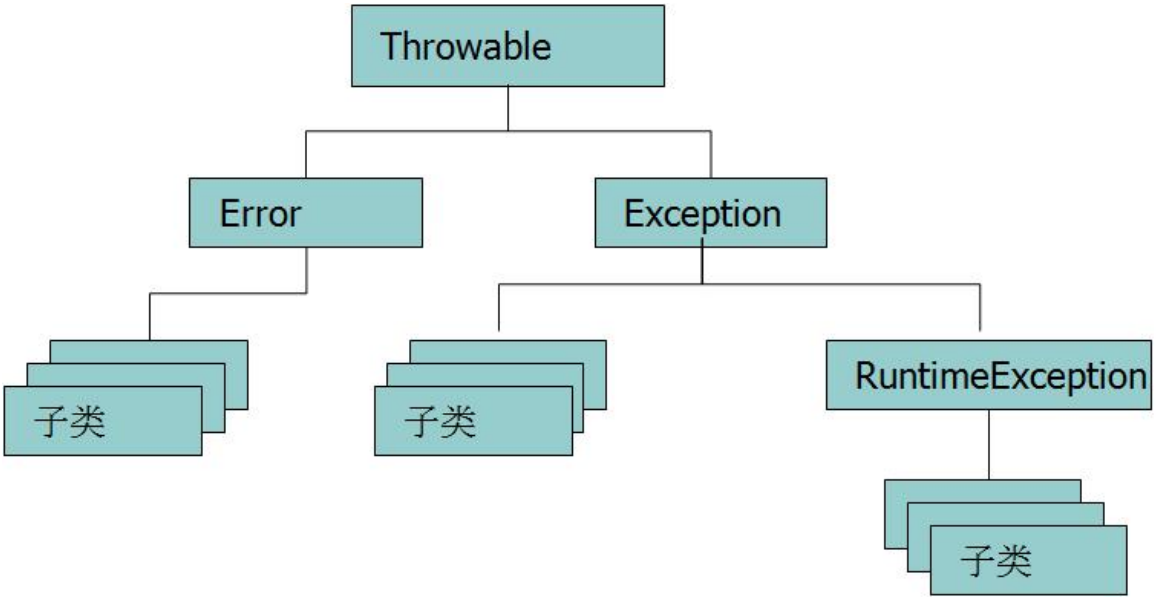
异常的分类

Java 中运行时发生的除了异常 Exception 还有错误 Error。

异常(Exception)：通常发生可以有针对性的处理方式的。

错误(Error) :通常发生后不会有针对性的处理方式。Error 的发生往往都是系统级别的问题，都是 jvm 所在系统发生的并反馈给 jvm 的。

异常的继承体系



JVM 默认处理异常的方式

main 函数收到这个问题时,有两种处理方式:

a:自己将该问题处理,然后继续运行

b:自己没有针对的处理方式,只有交给调用 main 的 jvm 来处理

jvm 有一个默认的异常处理机制,就将该异常进行处理.

并将该异常的名称,异常的信息.异常出现的位置打印在了控制台上,同时将程序停止运行

异常处理的两种方式

第一种 try catch 捕获异常：

```
try {
                        // 需要被检测的语句。
        } catch (异常类 变量)// 参数。
        {
                        // 异常的处理语句。
        } finally {
                        // 一定会被执行的语句,并且在 return 之前运行。
        }
```

try catch：对代码进行异常检测，并对检测的异常传递给 catch 处理。

try finally：对代码进行异常检测，检测到异常后因为没有 catch，所以一样会被默认 jvm 抛出，异常是没有捕获处理的。但是功能所开启资源需要进行关闭，所以 finally 只为关闭资源。

try catch finally:检测异常，并传递给 catch 处理，并定义资源释放。

第二种 throws 的方式处理异常：

定义功能方法时，需要把出现的问题暴露出来让调用者去处理，那么就通过 throws 在方法上标识。

```
public static void method(int age) throws IllegalArgumentException {
}
```

编译期异常和运行期异常的区别

Java 中的异常被分为两大类：编译时异常和运行时异常。

所有的 RuntimeException 类及其子类的实例被称为运行时异常，其他的异常就是编译时异常

编译时异常：Java 程序必须显示处理，否则程序就会发生错误，无法通过编译

运行时异常：无需显示处理，也可以和编译时异常一样处理

Throwable 的几个常见方法

getMessage()：获取异常信息，返回字符串。

toString()：获取异常类名和异常信息，返回字符串。

printStackTrace()：获取异常类名和异常信息，以及异常出现在程序中的位置。返回值 void。

throw 的概述

在功能方法内部出现某种情况，程序不能继续运行，需要进行跳转时，就用 throw 把异常对象抛出。

throws 和 throw 的区别

throws：1、用在方法声明后面，跟的是异常类名　2、可以跟多个异常类名，用逗号隔开 3、表示抛出异常，由该方法的调用者来处理

throw：　1、用在方法体内，跟的是异常对象名　　2、只能抛出一个异常对象名 3、表示抛出异常，由方法体内的语句处理

```
if(arr==null)    //空指针抛出异常
        {
```

```
                throw new NullPointerException("arr 指向的数组不存在");
        }
if(index<0 || index>=arr.length)      // 角标越界抛出异常
        {
                throw new ArrayIndexOutOfBoundsException("错误的角标，"+index+"索引在数组中不
存在");
        }
if(age<0 || age>200)    //无效参数抛出异常
        {
                throw new IllegalArgumentException(age+",年龄数值非法");
        }
```

自定义异常

1、继承自 Exception

2、继承自 RuntimeException

例：

```
class NoAgeException extends RuntimeException
        {
                /*
                为什么要定义构造函数，因为看到 Java 中的异常描述类中有提供对问题对象的初
始化方法。
                */
                NoAgeException()
                {
                        super();
                }
                NoAgeException(String message)
                {
                        super(message);// 如果自定义异常需要异常信息，可以通过调用父类
的带有字符串参数的构造函数即可。
                }
        }
```

继承 Exception 和继承 RuntimeExcpetion 的区别：

继承 Exception，必须要 throws 声明，一声明就告知调用者进行捕获，一旦问题处理了调用者的程序会继续执行。

但是如果使用到了对象的数据，导致都失败的。

继承 RuntimeExcpetion,不需要 throws 声明的，这时调用是不可能编写捕获代

码的，因为调用根本就不知道有问题。

一旦发生异常，调用者程序会停掉，并有 jvm 将信息显示到屏幕，让调用者看到问题，修正代码。在用 throws 抛出一个的时候，如果这个异常是属于 RuntimeException 的体系的时候，我们在调用的地方可以不用处理。

(RuntimeException 和 RuntimeException 的子类)

异常注意事项

子类重写父类方法时，子类的方法必须抛出相同的异常或父类异常的子类。

如果父类抛出了多个异常,子类重写父类时,只能抛出相同的异常或者是他的子集,子类不能抛出父类没有的异常

如果被重写的方法没有异常抛出,那么子类的方法绝对不可以抛出异常,如果子类方法内有异常发生,那么子类只能 try,不能 throws

如何使用异常处理

原则:如果该功能内部可以将问题处理,用 try,如果处理不了,交由调用者处理,这是用 throws

区别:

后续程序需要继续运行就 try

后续程序不需要继续运行就 throws

如果 JDK 没有提供对应的异常，需要自定义异常。

```
public class Demo {
		public static void main(String[] args) {
			try {
				Util.method(-1);
			} catch (IllegalArgumentException e) {
				Util.method(18);
```

```
                }
        }
}

class Util {

        public static void method(int age) throws IllegalArgumentException {
                if (age < 0 || age > 200) // 无效参数抛出异常
                {
                        throw new IllegalArgumentException(age + ",
年龄数值非法");
                }
                System.out.println("年龄为：" + age);

        }
}
```

finally 的特点

被 finally 控制的语句体一定会执行

特殊情况：在执行到 finally 之前 jvm 退出了(比如 System.exit(0))

finally 的作用

用于释放资源，在 IO 流操作和数据库操作中会见到

final,finally 和 finalize 的区别

final 是最终的意思。它可以用于修饰类，成员变量，成员方法。它修饰的类不能被继承，它修饰的变量时常量，它修饰的方法不能被重写。

finally:是异常处理里面的关键字。它其中的代码永远被执行。特殊情况：在执行它之前 jvm 退出。System.exit(0);

finalize:是 Object 类中的一个方法。它是于垃圾回收器调用的方式。

面试题

```
public class Demo {
		public static void main(String args[]) {
				Demo t = new Demo();
				int b = t.get();
				System.out.println(b);
		}

		public int get() {
				try {
						return 1;
				} catch (Exception e) {
						return 2;
				} finally {
						return 3;
				}
		}
}
```

try 中的 return 语句调用的函数先于 finally 中调用的函数执行 ,也就是说 return 语句先执行，finally 语句后执行，所以，返回的结果是 2。Return 并不是让函数马上返回，而是 return 语句执行后，将把返回结果放置进函数栈中，此时函数并不是马上返回，它要执行 finally 语句后才真正开始返回。

File 类的概述

File 可以称为文件路径或者文件夹路径

路径分为绝对路径和相对路径

绝对路径是一个固定的路径,从盘符开始

相对路径相对于某个位置,在 eclipse 下是指当前项目下,在 dos 下

构造方法

File(String pathname)：根据一个路径得到 File 对象

File(String parent, String child):根据一个目录和一个子文件/目录得到 File 对

象

File(File parent, String child):根据一个父 File 对象和一个子文件/目录得到 File 对象

File 类的创建功能

创建功能

public boolean createNewFile():创建文件 如果存在这样的文件，就不创建了

public boolean mkdir():创建文件夹 如果存在这样的文件夹，就不创建了

public boolean mkdirs():创建文件夹,如果父文件夹不存在，会帮你创建出来

File 类的重命名和删除功能

public boolean renameTo(File dest):把文件重命名为指定的文件路径

public boolean delete():删除文件或者文件夹

重命名注意事项

如果路径名相同，就是改名。

如果路径名不同，就是改名并剪切。

删除注意事项：

Java 中的删除不走回收站。

要删除一个文件夹，请注意该文件夹内不能包含文件或者文件夹

File 类的判断功能

public boolean isDirectory():判断是否是目录

public boolean isFile():判断是否是文件

public boolean exists():判断是否存在

public boolean canRead():判断是否可读

public boolean canWrite():判断是否可写

public boolean isHidden():判断是否隐藏

File 类的获取功能

public String getAbsolutePath()：获取绝对路径

public String getPath():获取路径

public String getName():获取名称

public long length():获取长度。字节数

public long lastModified():获取最后一次的修改时间，毫秒值

public String[] list():获取指定目录下的所有文件或者文件夹的名称数组

public File[] listFiles():获取指定目录下的所有文件或者文件夹的 File 数组

举个栗子

```
/**
  * 请删除一个带有内容的目录。 删除一个带有内容的目录,必须从里往外删。
  *
  * @author Somnus
  */

public class Demo {
                public static void main(String[] args) {
                                File dir = new File("E:\\test");
                                removeDir(dir);
                }
```

```
    /**
     * 删除一个目录。
     */
    public static void removeDir(File dir) {

        File[] files = dir.listFiles();// 如果目录是系统级文件夹，java没有访问权限，那么会返回 null 数组。最好加入判断。
        if (files != null) {

            for (File file : files) {

                if (file.isDirectory()) {
                    removeDir(file);
                } else {
                    System.out.println(file + ":" + file.delete());// 删除文件。用打印语句验证是否删除成功，是否出现了误删操作。
                }
            }
        }
        System.out.println(dir + ":" + dir.delete());

    }
}
```

文件名称过滤器

文件名称过滤器的概述

public String[] list(FilenameFilter filter)

public File[] listFiles(FileFilter filter)

文件名称过滤器的使用

需求：判断 E 盘目录下是否有后缀名为.jpg 的文件，如果有，就输出该文件名称

```
/**
 * 根据指定的过滤器在指定目录下获取所有的符合过滤条件的文件，并存储到 list 集合中。
 * @param dir
 * @param filter
```

```
     * @param list
     */
    public static void getFileList(File dir,FileFilter filter,List<File> list) {

                    File[] files = dir.listFiles();

                    for(File file : files){

                                    if(file.isDirectory()){
                                                    getFileList(file,filter,list);
                                    }else{
                                                    //如果是文件，传递到过滤器中去过滤。将满足条
件存储起来。
                                                    if(filter.accept(file)){
                                                                    list.add(file);
                                                    }
                                    }
                    }

    }
```