正则表达式

用于校验字符串格式是否正确（密码格式， 电子邮箱），查找等

用于校验字符串是否符合指定的格式

```
//      Pattern pattern = Pattern.compile("."); //任意单个字符
//      Matcher matcher = pattern.matcher("4");  //校验字符串
//      System.out.println(matcher.matches());

//      Pattern pattern = Pattern.compile("[abc]"); //包含abc中任意一个字符
//      Matcher matcher = pattern.matcher("a");  //校验字符串
//      System.out.println(matcher.matches());

//      Pattern pattern = Pattern.compile("[^abc]"); //包含非abc中任意一个字符
//      Matcher matcher = pattern.matcher("f");  //校验字符串
//      System.out.println(matcher.matches());

//      Pattern pattern = Pattern.compile("[a-zA-Z]"); //大写和小写字母
//      Matcher matcher = pattern.matcher("C");  //校验字符串
//      System.out.println(matcher.matches());

//      Pattern pattern = Pattern.compile("\\d+"); //一个或多个数字
//      Matcher matcher = pattern.matcher("00001234500");  //校验字符串
//      System.out.println(matcher.matches());

//      Pattern pattern = Pattern.compile("[a-zA-Z]*"); //零个或多个字母
//      Matcher matcher = pattern.matcher("abcCGWE");  //校验字符串
//      System.out.println(matcher.matches());

        //校验Java标识符：只能以字母，数字，-,$开头，后面字母还可以有数字
//      Pattern pattern = Pattern.compile("^[a-zA-Z_$]+[\\w_$]*");
//      Matcher matcher = pattern.matcher("$9abCfd7");
//      System.out.println(matcher.matches());

        //校验数字
//      Pattern pattern = Pattern.compile("^[0-9]+$");
//      Matcher matcher = pattern.matcher("");
//      System.out.println(matcher.matches());

        //8位数字
//      Pattern pattern = Pattern.compile("^\\d{8}$");  //"^\\d{8,16}$"  8到16位；"^\\d{8,}$"   8位以上数字
//      Matcher matcher = pattern.matcher("12345678");
//      System.out.println(matcher.matches());

        //非0开头，一到两位小数
//      Pattern pattern = Pattern.compile("^[1-9][0-9]*(.[0-9]{1,2}){1}$");
//      Matcher matcher = pattern.matcher("18.24");
//      System.out.println(matcher.matches());


        //中文
//      Pattern pattern = Pattern.compile("^[\\u4E00-\\u9FA5]+$");
```

```
//      Matcher matcher = pattern.matcher("中文");
//      System.out.println(matcher.matches());

        //邮箱
//      Pattern pattern = Pattern.compile("^\\w+([-+.]\\w+)*@\\w+
([-.]\\w+)*\\.\\w+([-.]\\w+)*$");
//      Matcher matcher = pattern.matcher("abc23-5@qq.44.fdsO");
//      System.out.println(matcher.matches());

        //手机号码
//      Pattern pattern = Pattern.compile("^[1][3,4,5,7,8][0-9]{9}$");
//      Matcher matcher = pattern.matcher("13888888888");
//      System.out.println(matcher.matches());


        //日期
//      Pattern pattern = Pattern.compile("^\\d{4}-\\d{1,2}-\\d{1,2}");
//      Matcher matcher = pattern.matcher("2020-22-58");
//      System.out.println(matcher.matches());
```

查找，替换

```
        //**************用正则表达式查找，替换************************
        String text = "<html><body>姓名：${name};  学号：${num}</body></html>";
        Pattern pattern = Pattern.compile("\\$\\{\\w+\\}");
        Matcher matcher = pattern.matcher(text);
//      System.out.println(matcher.replaceAll("***"));  //替换
        while(matcher.find()) {
            System.out.println("开始位置：" + matcher.start());
            System.out.println("结束位置：" + matcher.end());
            System.out.println(matcher.group()); //返回匹配的字符串
        }
```

表达式中常用符号

字符类

下表列出了正则表达式中常用的字符类

示例	说明
.	匹配任意单个字符
[abc]	包含abc中任意的字符,等价于a
[^abc]	除了abc外的任意字符
[a-zA-Z]	从a到z或者从A到Z中的任意字符
[abd[1-9]]	abd中任意字符或者1-9中任意字符，取并集
[a-z&&[hij]]	任意h、i、j字符，取交集
\s	空白符(空格、tab、换行、换页和回车)
\S	非空白符(^\s)
\d	数字(0-9)
\D	非数字(^0-9)
\w	词字符[a-zA-Z0-9]
\W	非词字符[^\w]

贪婪型	勉强型	占有型	说明
X?	X??	X?+	一个或0个X
X*	X*?	X*+	0个或多个X
X+	X+?	X++	一个或多个X
X{n}	X{n}?	X{n}+	恰好n次X
X{n,}	X{n,}?	X{n,}+	至少n次X
X{n,m}	X{n,m}?	X{n,m}+	至少n次，至多m次X

边界匹配符

例	说明
^	一行的起始
$	一行的结束

异常（Exception）

```
try{
    //正常业务逻辑




}catch(){
    //处理异常的情况
}
```

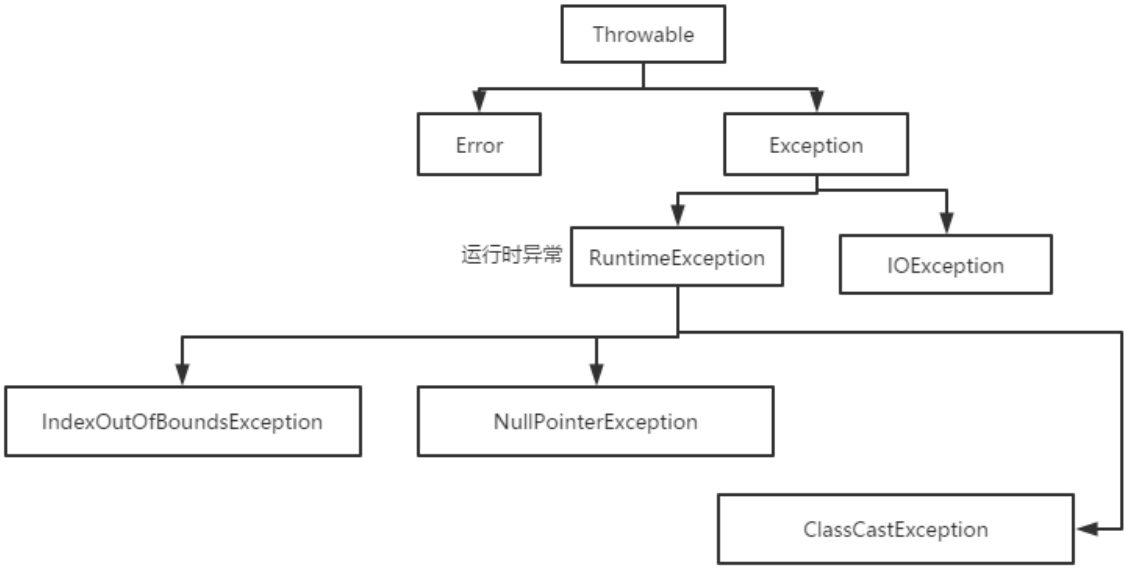
异常的类型

共同的跟类是Throwable,

Error程序运行中出现不可挽救的错误，比如硬件问题

Exception程序运行中出现例外，需要程序处理：给出提示，让用户重写操作

RuntimeException运行时异常，不强制要求处理异常：添加try catch， 添加trows声明



处理异常

直接给方法添加throws声明

```
public static void main(String[] args) throws FileNotFoundException, IOException
{
```

添加try cath

- 只用一个catch, exception类型用父类的类型

```
    try {
                properties.store(new FileOutputStream("x:\\a.txt"), "");
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
```

- 每一种异常类型一个catch

```
            try {
                properties.store(new FileOutputStream("x:\\a.txt"), "");
            } catch (FileNotFoundException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
```

从异常对象能获取的信息

```
e1.getMessage();  //返回异常信息
e1.getStackTrace(); //返回异常堆栈跟踪信息
e1.printStackTrace(); //打印异常堆栈跟踪信息
```

finally块的代码，可以有，也可以没有，特点不管异常是否发生，它一定会被执行

```
try {
                properties.store(new FileOutputStream("d:\\a.txt"), "");
                System.out.println("文件保存成功");
            } catch (FileNotFoundException e1) {
                e1.printStackTrace();
            } catch (IOException e1) {

            }finally {//不管异常是否发生，finally块的代码都会被执行。用于资源关闭
                System.out.println("执行finally");
            }
```

> 面试题：
>
> final, finally, finalize 有何异同?
>
> 相同：除了名字看起来相同，其实三者之间没有任何关系
>
> 不同：final是修饰符，用于修饰成员变量，方法，类， 添加final修饰的类和方法不能被继承，变量不能被重写赋值
>
> finally是异常中try catch 之后一个代码块，finally里面的代码不管异常是否发生，它都会被执行
>
> finalize它是Object的一个方法，当对象被垃圾回收时，这个方法会被调用

自定义异常

定义异常类

继承Exception或RuntimeException, 重写无参和有参构造器

```
package javaadvanceday04;

public class OrderException extends Exception {

    public OrderException() {
```

```
        super();
    }

    /**
     *
     * @param message 异常信息，打印出来看到的信息
     */
    public OrderException(String message) {
        super(message);
    }

}
```

在业务处理方法中抛出（throw）异常

throw 异常对象， 方法声明为throws 异常类型

```
package javaadvanceday04;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class OrderService {

    public void createOrder(String phone) throws OrderException  {
        Pattern pattern = Pattern.compile("^[1][3,4,5,7,8][0-9]{9}$");
        Matcher matcher = pattern.matcher(phone);
        if(!matcher.matches()) {
            //抛出异常
            throw new OrderException("联系电话格式不正确");
            //throw new Exception("错误消息");
        }
    }
}
```

调用业务方法时，捕获异常（try catch）

```
package javaadvanceday04;

public class OrderTest {

    public static void main(String[] args) {
        OrderService orderService = new OrderService();
        try {
            orderService.createOrder("138456987121");
        } catch (OrderException e) {
            System.out.println(e.getMessage()); //创建异常对象的构造方法的参数
            e.printStackTrace();
        }
    }

}
```

IO（Input，Output）

File

常用方法

```
package javaadvanceday04;

import java.io.File;
import java.io.IOException;
import java.util.Date;

public class FileDemo {

    public static void main(String[] args) throws IOException {
        File file = new File("C:\\Program Files\\Java\\jdk1.8.0_191");
        //文件名相关方法
        System.out.println(file.getName()); //返回文件名或目录名
        System.out.println(file.getAbsolutePath());  //返回绝对路径
        System.out.println(file.getParent()); //返回目录名字或上级目录名字

        //文件检测
        System.out.println(file.exists()); //返回文件或目录是否存在
        System.out.println(file.isFile()); //返回是否是文件
        System.out.println(file.isDirectory()); //返回是否是目录

        //文件信息
        System.out.println(file.length()); //文件长度
        System.out.println(new Date(file.lastModified())); //返回修改日期

        //操作文件
        File file2 = new File("d:\\java.txt");
        file2.createNewFile();  //创建新文件
        file2.delete();  //删除文件

        //操作目录
        File file3 = new File("d:\\aaa");
        file3.mkdir();  //创建目录
        String[] list = file.list(); //列出当前目录下的文件或子目录的文件名
        for(String s : list) {
            System.out.println(s);
        }
        System.out.println("================================");
        File[] listFiles = file.listFiles(); //列车当前目录下的文件和目录的File对象
        for(File f : listFiles) {
            System.out.println(f.getName());
        }


    }

}
```

今日作业

1. 用正则表达式提取如下字符串中的中文文字（见qq文件）
2. Throwable, Error, Exception, RuntimeException分别是什么?
3. 你见过哪些异常，这些是什么原因产生的?
4. final, finally, finalize有什么异同
5. 自定义登录异常类LoginException，写一个登录的业务方法login(String name, String password), 如果用户名和密码不正确，抛出LoginException. 写测试类调用该方法。
6. 写一个递归方法，遍历并打印一个目录的所有文件和目录（包含子目录），按照目录的层次进行缩进