本章内容

- 正则表达式

01_正则表达式的概述和简单使用

- A:正则表达式
  - 是指一个用来描述或者匹配一系列符合某个语法规则的字符串的单个字符串。其实就是一种规则。有自己特殊的应用。
- B:作用
  - 比如注册邮箱,邮箱有用户名和密码,一般会对其限制长度,这个限制长度的事情就是正则表达式做的
- C:案例演示
  - 需求：校验qq号码.
    - 1:要求必须是5-15位数字
    - 2:不能以0开头
    - 3:必须都是数字
  - a:非正则表达式实现
  - b:正则表达式实现 [1-9]\d{4,14}

```
Scanner scanner = new Scanner(System.in);
System.out.println("请输入您的QQ号码:");
String qqStr = scanner.nextLine();
if(qqStr.length() >= 5 && qqStr.length() <= 15) {

    if(!qqStr.startsWith("0")) {
        boolean flag = true;//记录是否是数字
        for (int i = 0; i < qqStr.length(); i++) {
            char chr = qqStr.charAt(i);
            if(!Character.isDigit(chr)) {
                flag = false;
                break;
            }
        }
        if(flag) {
            System.out.println("恭喜你，qq号不错呦~~");
        } else {
            System.out.println("必须都是数字!");
        }
    } else {
        System.out.println("不能以0开头!");
    }

} else {
```

```
        System.out.println("长度不对!");
    }
```

02_字符类演示

- A:字符类
  - [abc] a、b 或 c（简单类）
  - [^abc] 任何字符，除了 a、b 或 c（否定）
  - [a-zA-Z] a到 z 或 A到 Z，两头的字母包括在内（范围）
  - [0-9] 0到9的字符都包括
- B:注意事项
  - 如果没有规定长度，那么默认长度是1.

03_预定义字符类演示

- A:预定义字符类
  - . 任何字符。
  - \d 数字：[0-9]
  - \D 非数字:[^0-9]
  - \w 单词字符：[a-zA-Z_0-9]
  - \W 非单词字符:[^\w]
  - \s 空白字符:[\t\n\x0B\f\r]
  - \S 非空白字符:[^\s]

04_数量词

- A:Greedy 数量词
  - X? :X，一次或一次也没有
  - X* :X，零次到多次
  - X+ :X，一次到 多次
  - X{n} :X，恰好 n 次
  - X{n,} :X，至少 n 次
  - X{n,m} :X，至少 n 次，但是不超过 m 次

05_正则表达式的分割功能

- A:正则表达式的分割功能
  - public String[] split(String regex)
- B:案例演示
  - "a,b,c,d"
  - "a,,b,,c,,d"
  - "a,b,,c,,,d"

- "a,,b,,,c,d,,,,,e"
- 将以上四个字符串按照逗号进行分割。

06_Pattern和Matcher的概述

- A:Pattern和Matcher的概述
- B:模式和匹配器的典型调用顺序
  - 典型的调用顺序是

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

07_正则表达式的获取功能

- A:正则表达式的获取功能
  - Pattern和Matcher的结合使用
- B:案例演示
  - 需求：把一个字符串中的手机号码获取出来

```
String msg = "18627775385...fadsfds18627775383..dsfjdsa18627775384";
String reg = "[1]{1}[356789]{1}[0-9]{9}";
//1,将正则字符串转换为正则对象
Pattern pattern = Pattern.compile(reg);
//2,获取手机号
//2.1,获取到对应的匹配对象
Matcher matcher = pattern.matcher(msg);
//2.2,find方法：在此字符串中匹配是否满足正则表达式的子字符串，有匹配上返回true,否则false
//2.3,group方法，返回匹配上正则表达式的子字符串.
while(matcher.find()){
    String phone = matcher.group();
    System.out.println(phone);
}
```

08_正则表达式综合案例

- A.需求:
- 创建一个User对象，有三个字段：账户、密码、邮箱，有如下要求：
  - a.账户：
    - 由数字和字母组成,第一个位置上必须是大写字母，长度范围为6~10
    - [A-Z]{1}[a-zA-Z0-9]{5,9}
  - b.密码:
    - 由数字、字母和_组成，第一位置上必须是字母，长度为6
    - [a-zA-Z]{1}\w{5}
  - c.邮箱：
    - 由字母、@和.组成，长度范围为8~10,必须是qq邮箱

- [a-zA-Z]{1,3}@qq\.com