13. 正则表达式

13.1. 认识正则

```java
/**
 * 没有使用正则表达式来检查字符串是否由数字组成
 */
@Test
public void test1() {
    String s = "5201314";
    char[] chars = s.toCharArray();
    boolean flag = true;
    for (int i = 0; i < chars.length; i++) {
        if (chars[i] < '0' || chars[i] > '9') {
            flag = false;
            break;
        }
    }
    if (flag) {
        System.out.println("是由数字组成");
    } else {
        System.out.println("不是由数字组成");
    }
}
```

13.2. 正则表达式（Regular Expression）

- 正则表达式使用单个字符串来描述、匹配一系列符合某个句法规则的字符串
- 正则表达式通常被用来检索、替换那些符合某个模式的文本
- java.util.regex包中提供以下两个类对正则表达式的支持：
  - Matcher类：通过解释Pattern对character sequence执行匹配操作的引擎
  - Pattern类：正则表达式的编译表示形式

13.3. Pattern类

```java
public final class Pattern extends Object implements Serializable
```

- 正则表达式的编译表示形式。指定为字符串的正则表达式必须首先被编译为此类的实例
- 典型的调用顺序：

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

13.4. Matcher类

```
public final class Matcher extends Object implements MatchResult
```

- Matcher类的主要功能是用于进行正则的匹配，通过Pattern类中定义完整的正则表达式，再使用Matcher类进行验证或替换
- 常用方法：

```
// 尝试将整个区域与模式匹配
boolean matches();
// 替换模式与给定替换字符串相匹配的输入序列的每个子序列
String replaceAll(String replacement);
// 替换模式与给定字符串匹配的输入序列的第一个子序列
String replaceFirst(String replacement);
```

13.5. String类对正则的支持

- 在JDK1.4之后加入了正则，随后又更新了String的操作类，因为在使用正则中，所有的内容通过字符串表示的比较多。在String类中有以下的方法可以完成对正则的支持：
- 常用方法：

```
// 告知此字符串是否匹配给定的正则表达式
boolean matches(String regex);
// 使用给定的replacement替换此字符串，所有匹配给定的正则表达式的子字符串
String replaceAll(String regex, String replacement);
// 使用给定的replacement替换此字符串，匹配给定的正则表达式的第一个子字符串
String replaceFirst(String regex, String replacement);
// 根据给定正则表达式的匹配拆分此字符串
String[] split(String regex);
```

13.6. 示例

- 验证电话号码（如：010-38389438）
- 验证手机号码
- 验证用户名，只能是字母开头的数字、字母和下划线的组合
- 验证IP地址（如：192.168.1.1）

- 验证网址（如：http://www.baidu.com）
- 验证年龄（100以内）
- 验证金额（可以有小数位）

```
import org.junit.Test;

/**
 * @author xiao儿
 * @date 2019/9/3 23:06
 * @Description RegexDemo2
 */
public class RegexDemo2 {
    @Test
    public void test() {
        // 匹配电话号码
        String phoneNumber = "010-38389438";
        boolean b = phoneNumber.matches("\\d{3,4}-\\d{7,8}");
        System.out.println("电话号码匹配: " + b);

        // 匹配手机号码
        String phone = "13895271234";
        boolean b1 = phone.matches("[1][3-9]\\d{9}");
        System.out.println("手机号码匹配: " + b1);

        // 匹配用户名: 字母开头, 数字字母下划线组合
        String username = "abc1234";
        boolean b2 = username.matches("[a-zA-Z]+[\\w|_]*");
        System.out.println("用户名匹配: " + b2);

        // 匹配IP地址
        String ip = "192.168.1.1";
        boolean b3 = ip.matches("\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3}");
        System.out.println("IP地址匹配: " + b3);

        // 匹配网址
        String addr = "http://www.baid.com";
        boolean b4 = addr.matches("http://\\w+.\\w+.\\s*");
        System.out.println("网址匹配: " + b4);

        // 匹配年龄
        String age = "18";
        boolean b5 = age.matches("\\d{1,3}");
        System.out.println("年龄匹配: " + b5);

        // 匹配金额
        String price = "19.8";
        boolean b6 = price.matches("\\d+.\\d+");
        System.out.println("金额匹配: " + b6);
    }
}
```

13.7. 注意

- 菜鸟教程正则表达式链接: https://www.runoob.com/regexp/regexp-tutorial.html