JAVA中字符串操作及正则表达式简介

面向对象编程简介

三大特性是：封装,继承,多态
五大基本原则：单一职责原则SRP,开放封闭原则OCP，替换原则，依赖原则，接口分离原则

文档

面向对象编程简介

面向对象与面向过程的区别：
设计一个方法，用于描述狼吃羊这个事情，某只狼吃了某只羊，你可以面向过程地吃，eat(狼A, 羊A)，也可以面向对象地吃，狼A.eat(羊A)
参考文章

Java简介

语言历程

1995年，SUN公司推出JAVA 是一款高级编程语言

2004年，JDK1.5发布（重要版本）

2009年，ORACLE公司收购SUN，取得了JAVA的版权

目前，JAVA是应用最为广泛的计算机语言

JAVA重要特性—跨平台—（一次编译，到处运行）

原因：在编译后的文件和操作系统之间多了一层虚拟机

虚拟机：解释编译后文件并通知系统要执行哪些操作，充当中介功能

字符串(String)简介

- 创建字符串
- 方式1： String str1 = “abc”
- 方式2： String str2 = new String(“abc”)
- 其中new String()中的参数形式可以有多种
- 如：String(bytes[] byte) 接受一个字节数组
  - String(char[] value) 接受一个字符数组
  - String(String str) 接受一个字符串对象
- 使用字节数组或者字符数组都可以构建字符串对象

字符串(String)简介

- 创建字符串

```
5   public static void main(String[] args) {
6       String str1 = "abc";
7       String str2 = new String("abc");
8       byte[] bt = {97,98,99};
9       String str3 = new String(bt);
10      char[] ch = {'a','b','c'};
11      String str4 = new String(ch);
12      System.out.println("Str1 = "+ str1 + " Str2 = "+ str2);
13      System.out.println("Str3 = "+ str3 + " Str4 = "+ str4);
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月20日

```
Str1 = abc Str2 = abc
Str3 = abc Str4 = abc
```

字符串(String)简介

- 基本操作
- 判断两个字符串内容是否一致
- str1 == str2 判别的是两个字符串的内存地址
- str1.equals(str2) 判别的是两个字符串的内容

```
6       String str1 = "Hello";
7       String str2 = new String("Hello");
8       System.out.println("str1 == str2?" + (str1==str2));
9       System.out.println("str1.equals(str2)?"+ (str1.equals(str2)));
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月20日 下午6:31:15)

```
str1 == str2?false
str1.equals(str2)?true
```

字符串(String)简介

- 基本操作
- a 获取方法
- 1. 获取字符串长度：int length() 长度即字符个数
  2. 获取特定位置的字符 char charAt(int index)
  3. 获取特定字符的位置 int indexOf(String str) 返回子串第一次出现的索引值,如果子串没出现在字符串中，则返回-1
  4. 获取最后一个字符位置 int lastIndexOf(String str) 返回子串最后一次出现的索引值，如果子串没出现在字符串中，则返回-1

字符串(String)简介

- 基本操作- 获取方法

```
5  ublic static void main(String[] args) {
6    String str = "abc中国ab中国";
7    System.out.println("字符串中字符个数："+str.length());
8    System.out.println("索引值获取对应字符："+str.charAt(3));
9    System.out.println("查找子串首次出现索引值："+str.indexOf("中国"));
10   System.out.println("查找子串末次出现索引值："+str.lastIndexOf("中国")
11
12
13
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月20日 下午9:17:15)

```
字符串中字符个数：9
索引值获取对应字符：中
查找子串首次出现索引值：3
查找子串末次出现索引值：7
```

字符串(String)简介

- 基本操作
- b判断方法
- 1. 是否以指定字符结束 boolean endsWith(String str)
- 2. 是否长度为0 boolean isEmpty()
- 3. 是否包含指定序列 boolean contains(String str)
- 4. 是否内容一致 boolean equals(Object obj)
- 5. 忽略大小写是否内容一致 boolean equalsIngoreCase(String str)

字符串(String)简介

- 基本操作-判断方法

```
6       String str = "Demo.java";
7       System.out.println("是否以指定字符结束："+str.endsWith("va"));
8       String str1 = "";
9       System.out.println("字符串是否为空："+str1.isEmpty());
10      System.out.println("字符串是否包含指定内容："+str.contains("Demo"));
11      System.out.println("字符串是否一致（忽略大小写）："+
12                                  str.equalsIgnoreCase("DEMO.JAVA"));
13    }
14 }
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月20日 下午9:35:36)

```
是否以指定字符结束：true
字符串是否为空：true
字符串是否包含指定内容：true
字符串是否一致（忽略大小写）：true
```

字符串(String)简介

- 基本操作
- c 转换方法（字符数组，字节数组和字符串可以相互转换）
- 1. 将字符串转换为字符数组 char[] toCharArray()
- 2. 将字符串转换为字节数组   byte[] getBytes()

字符串(String)简介

- 基本操作-转换方法

```
7   public static void main(String[] args) {
8       String str = "Demo.java";
9       char[] buf = str.toCharArray();
10      System.out.println("字符数组："+Arrays.toString(buf));
11      byte[] buf2 = str.getBytes();
12      System.out.println("字节数组："+Arrays.toString(buf2));
13  }
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月20日 下午9:5

字符数组：[D, e, m, o, ., j, a, v, a]

字节数组：[68, 101, 109, 111, 46, 106, 97, 118, 97]

字符串(String)简介

- 基本操作
- d 其它方法
- 1. 替换 String replace(String target, String replacement) 顺序替换
- 2. 切割 String[] split(String str)
- 3. 截取子串 String substring(int begin, int end) 取头舍尾
- 4. 转大写 String toUpperCase()
- 5. 转小写 String toLowerCase()
- 6. 去除空格 String trim() 去除字符串首尾的空格（包括\t之类字符，但全角空格去不掉）

字符串(String)简介

基本操作-其它方法

```
 8        String str = "abbaabbaaaaa";
 9        System.out.println("指定新内容替换旧内容：" + str.replace("aa", "cc"));
10        String str1 = "aa-Ee-cc-dd";
11        String str2 = "  sadas\t";
12        String[] arr = str1.split("-");
13        System.out.println("字符串数组的内容："+Arrays.toString(arr));
14        System.out.println("从索引值开始截取子串："+str1.substring(3,5));
15        System.out.println("转大写："+ str1.toUpperCase());
16        System.out.println("转小写："+ str1.toLowerCase());
17        System.out.println("去除首尾空格：" +str2.trim());
18    }
19 }
```

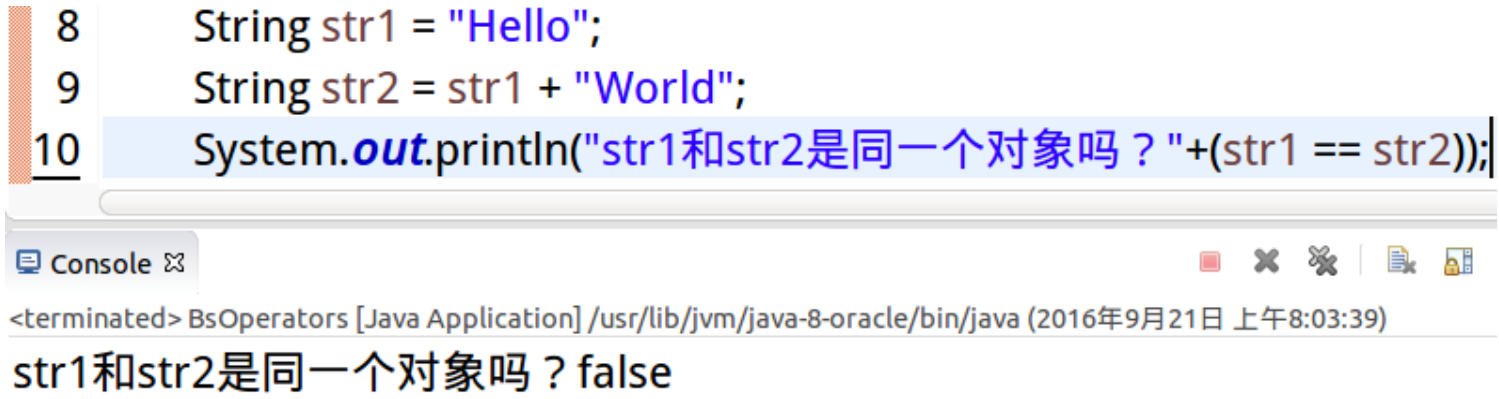
Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月20日 下午10:32:31)

```
指定新内容替换旧内容：abbccbbcccca
字符串数组的内容：[aa, Ee, cc, dd]
从索引值开始截取子串：Ee
转大写：AA-EE-CC-DD
转小写：aa-ee-cc-dd
去除首尾空格：sadas
```

字符串(String)简介

- 重要特性
- 字符串是常量，它们的值在创建后不能修改，字符串的内容一旦发生变化，那么马上会创建一个新的对象。
- 字符串的内容不适宜频繁修改，因为一旦修改马上就会创建一个新的对象。
- 如果需要频繁修改字符串的内容，建议使用字符串缓冲类（StringBuffer）。
- StringBuffer其实就是一个存储字符的容器



字符串(String)简介

- StringBuffer类
- 构造对象：
- StringBuffer sb = new StringBuffer()
- StringBuffer 底层是依赖了一个字符数组才能存储字符数据的，该字符数组默认的初始容量是16，如果字符数组的长度不够使用时，自动增长一倍。
- 基本操作：
- 增，删，查，改，判断

字符串(String)简介

- StringBuffer类基本操作
- 增加
- append(xxx) 可添加任意类型的数据
- Insert(int offset, xxx) 可插入任意类型的数据－String类没有该方法
- 删除
- delete(int start, int end) 根据指定的头尾索引值删除对应内容 －String类没有该方法
- deleteCharAt(int index) 根据指定索引值删除一个字符
- 修改
- replace(int start, int end, String str) 根据指定头尾索引值替换成指定内容
- reverse() 翻转字符串缓冲类的内容
- setCharAt(int index) 把指定索引值的字符替换指定字符
- substring(int start, int end) 根据指定索引值截取子串

字符串(String)简介

```
StringBuffer sb = new StringBuffer();
sb.append("abc");
sb.append(true);
sb.append(3.14);
System.out.println("字符缓冲类里的内容："+sb);
sb.insert(5, "INS");
System.out.println("字符缓冲类里的内容："+sb);
sb.delete(2, 4);
System.out.println("字符缓冲类里的内容："+sb);
sb.deleteCharAt(3);
System.out.println("字符缓冲类里的内容："+sb);
sb.replace(3, 5, "自然语言");
System.out.println("字符缓冲类里的内容："+sb);
sb.reverse();
System.out.println("字符缓冲类里的内容："+sb);
sb.reverse();
System.out.println("字符缓冲类里的内容："+sb.substring(3, 7));
```

```
字符缓冲类里的内容：abctrue3.14
字符缓冲类里的内容：abctrINSue3.14
字符缓冲类里的内容：abrINSue3.14
字符缓冲类里的内容：abrNSue3.14
字符缓冲类里的内容：abr自然语言ue3.14
字符缓冲类里的内容：41.3eu言语然自rba
字符缓冲类里的内容：自然语言
```

字符串(String)简介

- StringBuffer类基本操作
- 查找
- indexOf(String str, int index)
- capacity() 查看字符数组的长度（非已存储的字符串长度）
- length() 查看存储字符的个数
- charAt() 查看字符串指定位置的字符
- toString() 把字符串缓冲类的内容转换成字符串返回
- 若对字符串有较多增删操作，使stringbuffer类更好

字符串(String)简介

- StringBuffer类基本操作

```
 7   public static void main(String[] args) {
 8      StringBuffer sb = new StringBuffer();
 9      sb.append("abcdef");
10      int index = sb.indexOf("cde",2);
11      System.out.println("索引值为："+ index);
12      System.out.println("字符数组长度："+sb.capacity());
13      System.out.println("存储字符的个数："+sb.length());
14      System.out.println("索引值指定的字符："+sb.charAt(3));
15
16   }
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月21日 下午7:0

```
索引值为：2
字符数组长度：16
存储字符的个数：6
索引值指定的字符：d
```

字符串(String)简介

- 其它字符串缓冲类
- StringBuilder类
- 相同点 两个类都是字符串缓冲类，两个类方法都是一致的
- 不同点 StringBuffer是线程安全的，操作效率低 StringBuilder 是线程非安全的，操作效率高

正则表达式

- 正则表达式是用于操作字符串的一个规则，正则表达式的规则使用了特殊的符号表示
- 1. 预定义字符类
- . 任意字符
- \D 非数字:[^0-9]
- \d 数字:[0-9]
- \s 空白字符:[\t\n\x0B\f\r
- \S 非空白字符:[^\s]
- \w 单词字符:[a-zA-Z_0
- \W 非单词字符:[^\w]

```
7   public static void main(String[] args) {
8       System.out.println("Any:"+"a".matches("."));
9       System.out.println("Digit:"+"1".matches("\\d"));
10      System.out.println("Nondigit:"+"a".matches("\\D"));
11      System.out.println("Blank:"+"\t".matches("\\s"));
12      System.out.println("NonBlank:"+"s".matches("\\S"));
13      System.out.println("Word:"+"w".matches("\\w"));
14      System.out.println("Nonword:"+"#".matches("\\W"));
15  }
16 }
```

```
Console
<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月21日 下
Any:true
Digit:true
Nondigit:true
Blank:true
NonBlank:true
Word:true
Nonword:true
```

正则表达式

2 数量词

X?0 次或一次
X+至少出现一次
X* 0次或者多次
X{n} n 次。
X{n,} n 次。
X{n,m} n~m 次

```
8   System.out.println("<=1, ?:"+"a".matches("\\w?"));
9   System.out.println("0or>1, *:"+"".matches("\\d*")+"22".matches("\\d*"));
10  System.out.println(">=1, +:"+"a".matches("\\D+"));
11  System.out.println("n, {n}"+"\n\r\n".matches("\\s{3}"));
12  System.out.println("n~m, {n,m}:"+"1232132".matches("\\d{1,9}"));
13  System.out.println(">=n, {n,}:"+"www".matches("\\w{2,}"));
14
15
16
```

Console

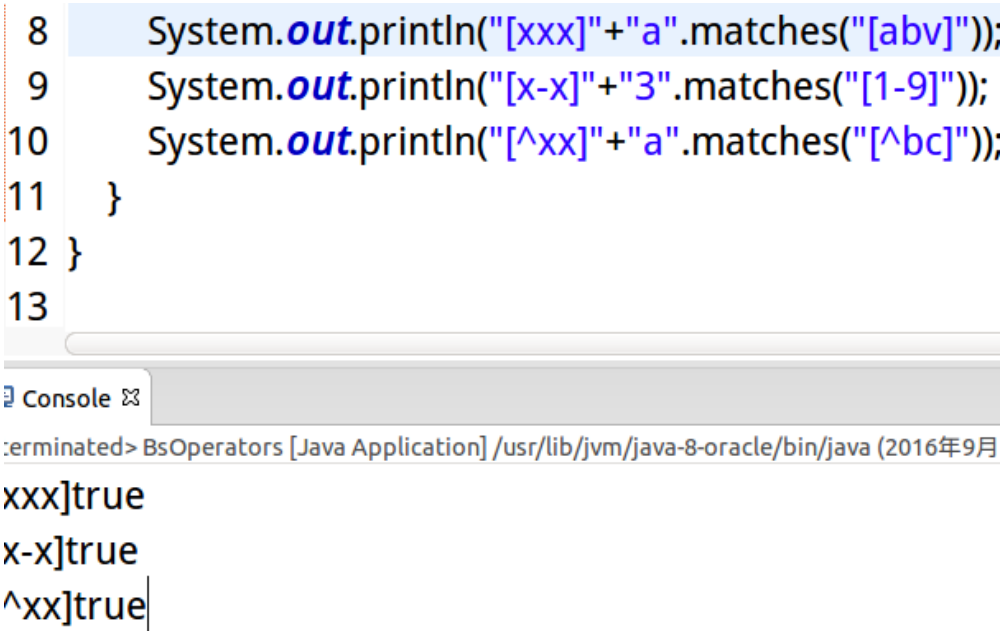
<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月21日 下午7:59:32)

```
<=1, ?:true
0or>1, *:truetrue
>=1, +:true
n, {n}true
n~m, {n,m}:true
>=n, {n,}:true
```

正则表达式

3.范围词

[abc] a,b或者c
[^abc] 除a,b,c任意字符
[a-zA-Z] a-z或A-Z的任字符
[a-d[m-p]] a到d或m到p
[a-z&&[def]] d,e或f(交集)
范围词内无论多长，
都只匹配一个字符而已



正则表达式

- 正则表达式对字符串操作的主要应用
- 匹配 matches()
- 切割 split()
- 替换 replaceAll(String regex, String replacement)
- 查找 查找需要使用的对象：Pattern(正则对象)Matcher(匹配器对象)

正则表达式

- 匹配
- 需求：编写正则表达式匹配手机号 第一位：只能1开头，第二位：3 4 5 7 8，长度:11位

```
 8     phoneChk("17717928609");
 9 }
10
11 public static void phoneChk(String phone) {
12     System.out.print(phone+":"+(phone.matches("1[3578]\\d{9}")?
13             "Valid":"Invalid"));
14 }
```

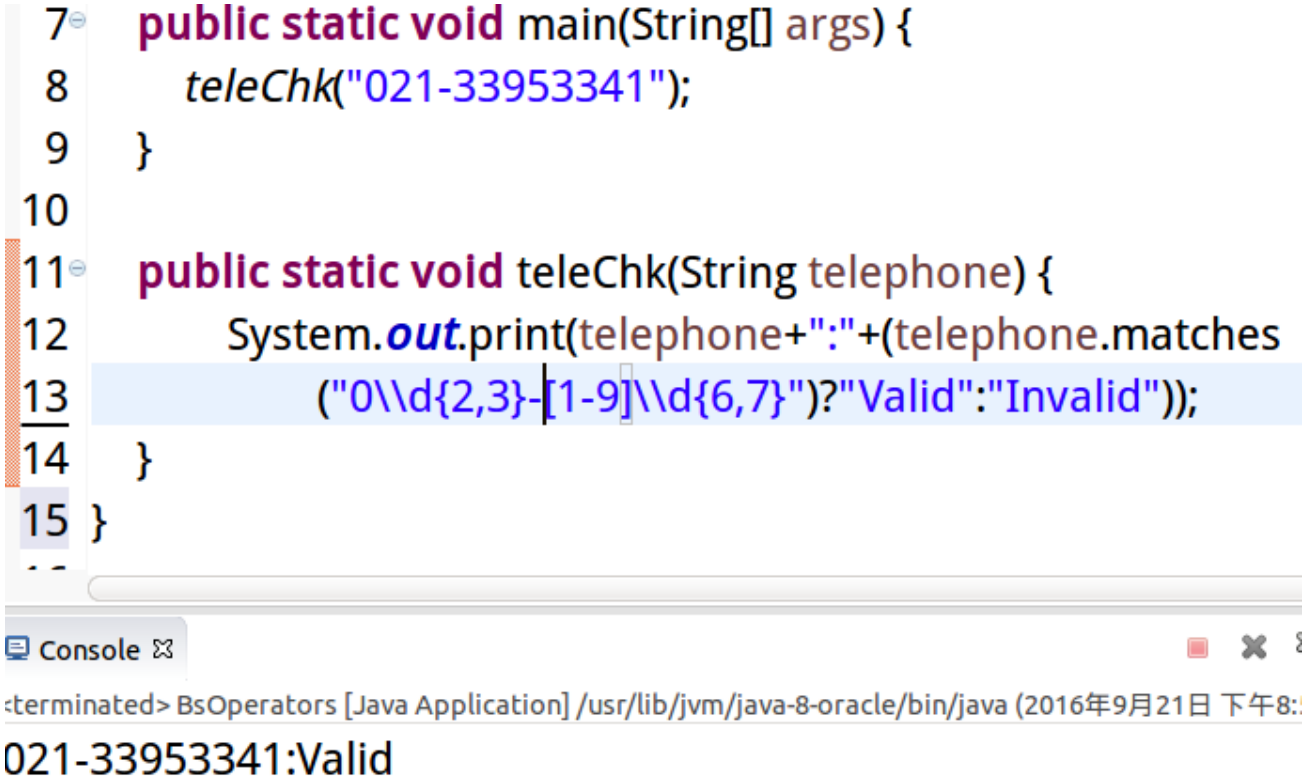
Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月21日 下午8:48:37)

17717928609:Valid

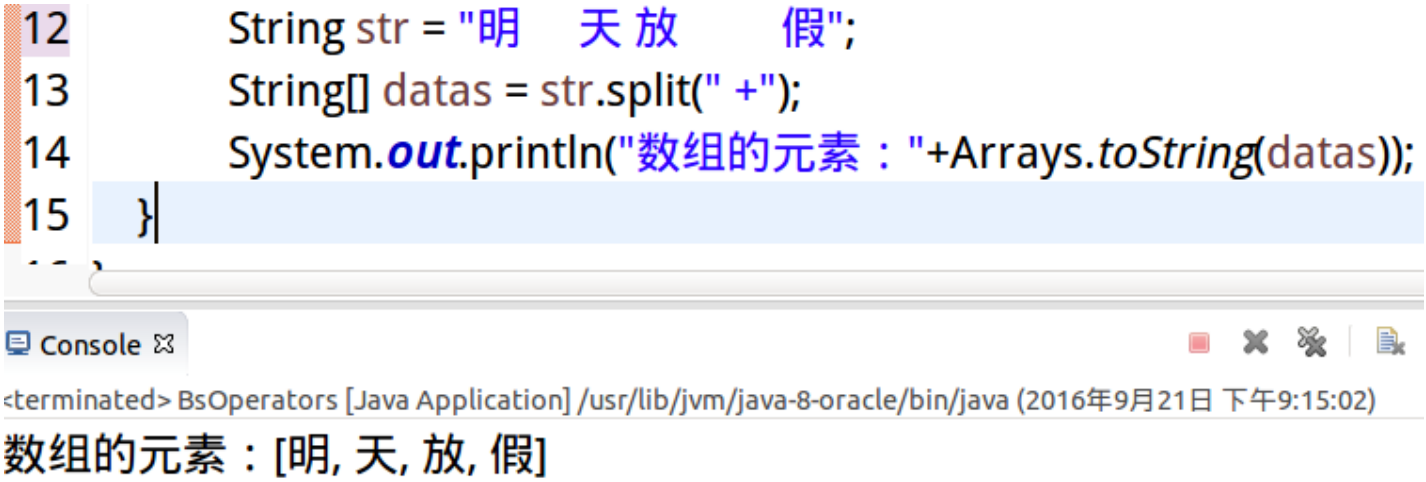
正则表达式

- 匹配
- 需求：编写正则表达式匹配固话 形式 区号-主机号 区号首位是0，长度3-4位 主机号：首位非0，长度7-8位



正则表达式

- 切割
- 需求：字符串各个字符之间多个空格，需分割字符



正则表达式

- 切割
- 需求：根据重叠词进行切割
- ()分组，如果正则的内容需要被复用，那么需要对正则的内容进行分组，分组的目的就是为了提高正则的复用性。组号不能指定，组号从1开始
- 例((A)(B(C))) 组1:((A)(B(C))) 组2 (A) 组3 (B(C)) 组4 (C)

```
12          String str = "大家家家天学学学学开心";
13          String[] datas = str.split("(.)\\1+");
14          System.out.println("数组的元素："+Arrays.toString(datas));
15      }
```

```
Console
erminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月21日 下午9:20:11)
数组的元素：[大, 天, 开心]
```

正则表达式

- 替换
- 需求：去除文本中的广告电话

```
12          String str = "如有需求请联系我：13773278832 如有需要请联系我："
13                  + "13818323212 如有需要请联系我：13818423212 "
14                  + "如有需要请联系我：13816323212";
15          str = str.replaceAll("1[34578]\\d{9}","*****");
16          System.out.println("替换后："+str);
17      }
18  }
19
20
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月21日 下午9:38:32)

替换后：如有需求请联系我：***** 如有需要请联系我：***** 如有需要请联系我：

正则表达式

- 替换
- 需求：还原重复输入的话
- 如需要在方法正则的外部引用组内容，那么使用“$组号”

```
12          String str = "我我要要要学学学学习";
13          str = str.replaceAll("(.)\\1+","$1");
14          System.out.println("替换后："+str);
15      }
16  }
```

Console

erminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/ja

替换后：我要学习

正则表达式

- 查找
- 典型调用顺序 Pattern p= Pattern.compile(“正则”)
- Matcher m = p.matcher(“aaaab”)

- 查找需要的对象
- 1 Pattern 2 Matcher
- 匹配器要使用的方法
- 1 find() 通知匹配器去查找，查找符合规则的字符串
- 2 group() 获取符合规则的字符串

正则表达式

\b代表单词边界，不匹配任何字符
用group方法时，一定先要调用find方法

```
13          String content = "some sds sdf dsfd jksja ss sss dsfjkl sss";
14          String reg = "\\b[a-zA-Z]{3}\\b";
15          Pattern p = Pattern.compile(reg);
16          Matcher m = p.matcher(content);
17          while(m.find()){
18              System.out.println(m.group());
19          }
```

Console

<terminated> BsOperators [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (2016年9月21日 下午10:09:08)

```
sds
sdf
sss
sss
```

谢　谢！