回顾：String

·初始化

·一旦声明则无法改变

·掌握String类的基本的操作方法（构造方法和普通方法）

正则表达式：

·正则表达式的使用

·Pattern类和Matcher类

·String类对正则的支持

具体内容：

1. 正则表达式的使用

在java刚起步的时候，java中并没有正则表达式，而是JDK1.4之后才加入到java中，如果说在JDK1.4之前想要使用正则表达式的话，就需要到Apache的网站上下载正则表达式的开发包，正则表达式在各个语言中都可以使用，最早兴起于PHP并且得到了广泛的好评，主要作用是用于一些比较复杂的字符串的验证。

举例：判断一个字符串是否全部由数字组成

·不适用正则表达式：

|  |
| --- |
| **package** com.wanczy.regexDemo;  **public** **class** RegexDemo01 {  **public** **static** **void** main(String[] args) {  String s = "456786131aa23745564";//要验证此字符串是否全部由数字组成  **char** c[] = s.toCharArray();  **for** (**int** i = 0; i < c.length; i++) {  **if**(!(c[i]>='0' && c[i]<='9')){  System.*out*.println("此字符串不全部是由数字组成");  **return**;  }  }  System.*out*.println("此字符串全部由数字组成");  }  } |

·使用正则表达式：

|  |
| --- |
| **package** com.wanczy.regexDemo;  **public** **class** RegexDemo02 {  **public** **static** **void** main(String[] args) {  String s = "45678613123745564";//要验证此字符串是否全部由数字组成  **if**(s.matches("\\d+")){  System.*out*.println("此字符串全部由数字组成");  **return**;  }  System.*out*.println("此字符串不全部是由数字组成");  }  } |

通过上面的例子，发现使用正则表达式去验证字符串比不适用正则更加方便。

二、Pattern类和Matcher类

·Pattern：列出了所有正则的内容

常用的正则表达式：

字符匹配：

|  |  |  |
| --- | --- | --- |
| 编号 | 正则表达式 | 描述 |
| 1 | [abc] | 表示取值可能是a或b或c |
| 2 | [^abc] | 表示取值不是a或b或c |
| 3 | [a-zA-Z] | 表示全部是字母，不区分大小写 |

简短表达式：

|  |  |  |
| --- | --- | --- |
| 编号 | 正则表达式 | 描述 |
| 1 | \d | 表示由数字组成 |
| 2 | \D | 表示由非数字组成 |
| 3 | \w | 表示由数字字母下划线组成 |
| 4 | \W | 表示由非数字、非字母、非下划线组成 |
| 5 | \s | 表示由空格组成\n \t |
| 6 | \S | 表示由非空格组成 |

列出出现次数的表达式

|  |  |  |
| --- | --- | --- |
| 编号 | 正则表达式 | 描述 |
| 1 | 表达式+ | 出现一次或多次 |
| 2 | 表达式？ | 出现0次或一次 |
| 3 | 表达式\* | 表示出现0次或多次 |
| 4 | 表达式{n} | 表示正好出现n次 |
| 5 | 表达式{n,m} | 表示出现的次数在n到m之间 |
| 6 | 表达式{n,} | 表示至少出现n次 |

关系运算

|  |  |  |
| --- | --- | --- |
| 编号 | 正则表达式 | 描述 |
| 1 | （表达式1|表达式2） | 表示要么是表达式1的正则要么是表达式2的正则 |

对于以上的常用的正则表达式，我们来写一个简单的例子，验证一段字符是否是手机号码？

|  |
| --- |
| **package** com.wanczy.regexDemo;  **public** **class** RegexDemo03 {  **public** **static** **void** main(String[] args) {  //找出常用手机号码的规则，手机号码都是11位，手机号码要么是13开始，要么是14,15,18，后面是9位数字  String tel = "1351596753";  **if**(tel.matches("(13|14|15|18)\\d{9}")){//将字符串与正则进行匹配，如果匹配成功，则返回true  System.*out*.println("字符串是手机号码");  }**else**{  System.*out*.println("此字符串不是手机号码");  }  }  } |

Pattern类：是正则操作中最重要的一个类，所有的正则规范都需要在此类中定义。

要是用Pattern类，就必须取得此类的对象。通过API发现此类没有构造方法，那么怎么去实例化此类的对象呢？

Pattern类的主要方法：

·取得Pattern的对象：public static Pattern compile(String regex)，参数为正则表达式

·取得Matcher类的对象：public Matcher matcher(CharSequence input)，参数是需要匹配正则的字符串

·根据正则表达式进行字符串拆分：public String[] split(CharSequence input):参数是所需要拆分的字符串

·返回正在使用的正则表达式：public String pattern()

范例：使用Pattern类中的将字符串按照规定的正则进行拆分，有一串字符串，现在要根据字符串中的数字进行拆分。

|  |
| --- |
| **package** com.wanczy.regexDemo;  **import** java.util.regex.Pattern;  **public** **class** RegexDemo04 {  **public** **static** **void** main(String[] args) {  String str = "asd12sdfas345678lklcvzx34562323,mnuyta6778786skldfj";  Pattern pat = Pattern.*compile*("\\d+");//取得pattern类对象，并且定义正则规范  String s[] = pat.split(str);//进行字符串拆分  **for** (**int** i = 0; i < s.length; i++) {  System.*out*.println(s[i]);  }  }  } |

Pattern类的主要作用，用于定义正则规范，并进行字符串拆分。

·Matcher:主要作用是用于进行正则的匹配，通过Pattern定义的正则规范去匹配字符串，并且还能将字符串进行替换和验证。

主要方法如下：

·进行正则表达式与字符串的匹配：public boolean matches()

·替换全部：public String replaceAll(String replacement)，参数是替换后的字符串

·替换第一个匹配正则的字符串：public String replaceFirst(String replacement)，参数是替换后的字符串。

范例1：用正则表达式验证手机号码

|  |
| --- |
| **package** com.wanczy.regexDemo;  **import** java.util.regex.Matcher;  **import** java.util.regex.Pattern;  **public** **class** RegexDemo05 {  **public** **static** **void** main(String[] args) {  //找出常用手机号码的规则，手机号码都是11位，手机号码要么是13开始，要么是14,15,18，后面是9位数字  String str = "12515967537";  Pattern pat = Pattern.*compile*("(13|14|15|18)\\d{9}");//定义正则规范  Matcher mat = pat.21(str);//取得Matcher对象并且定义需要匹配正则的字符串  **if**(mat.matches()){//进行正则匹配  System.*out*.println("此字符串是手机号码");  }**else**{  System.*out*.println("此字符串不是手机号码");  }  }  } |

范例2：进行字符串替换，将字符串中的数字全部替换成X

|  |
| --- |
| **package** com.wanczy.regexDemo;  **import** java.util.regex.Matcher;  **import** java.util.regex.Pattern;  **public** **class** RegexDemo06 {  **public** **static** **void** main(String[] args) {  String str = "asd12sdfas345678lklcvzx34562323,mnuyta6778786skldfj";  Pattern pat = Pattern.*compile*("\\d+");//取得pattern类对象，并且定义正则规范  Matcher mat = pat.matcher(str);//取得Matcher对象并且定义需要匹配正则的字符串  String s = mat.replaceAll("X");//进行正则替换  System.*out*.println(s);    }  } |

范例：进行字符串替换，将字符串中的第一个匹配到正则的数字替换成X

|  |
| --- |
| package com.wanczy.regexDemo;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class RegexDemo07 {  public static void main(String[] args) {  String str = "asd12sdfas345678lklcvzx34562323,mnuyta6778786skldfj";  Pattern pat = Pattern.compile("\\d+");//取得pattern类对象，并且定义正则规范  Matcher mat = pat.matcher(str);//取得Matcher对象并且定义需要匹配正则的字符串  String s = mat.replaceFirst("X");//进行正则替换  System.out.println(s);    }  } |

三、String对正则的支持

在JDK1.4之后java加入了正则表达式，随后更新了String类的操作，使得在某些操作时不需要对Pattern和Matcher类进行实例化，而是直接使用String类中的方法即可。

在String类中支持正则的方法：

·public boolean matches(String regex)：进行字符串匹配，支持使用正则表达式进行匹配  
 ·public String replaceAll(String regex, String replacement)：支持正则的字符串替换

·public String replaceFirst(String regex, String replacement)：支持正则的字符串第一个匹配到的进行替换

·public String[] split(String regex)：支持正则的字符串拆分方法

范例1：进行字符串替换

|  |
| --- |
| **package** com.wanczy.regexDemo;  **public** **class** RegexDemo08 {  **public** **static** **void** main(String[] args) {  String str = "asd12sdfas345678lklcvzx34562323,mnuyta6778786skldfj";  // Pattern pat = Pattern.compile("\\d+");//取得pattern类对象，并且定义正则规范  // Matcher mat = pat.matcher(str);//取得Matcher对象并且定义需要匹配正则的字符串  String s = str.replaceAll("\\d+","X");//进行正则替换  System.*out*.println(s);    }  } |

范例2：进行字符串拆分

|  |
| --- |
| **package** com.wanczy.regexDemo;  **public** **class** RegexDemo09 {  **public** **static** **void** main(String[] args) {  String str = "asd12sdfas345678lklcvzx34562323,mnuyta6778786skldfj";  String s[] = str.split("\\d+");//进行正则替换  **for** (**int** i = 0; i < s.length; i++) {  System.*out*.println(s[i]);  }  }  } |

总结：

·正则中Pattern和Matcher的作用（JDK1.4之前在java中使用正则的必备步骤）

·掌握String类对正则的支持：字符串匹配、字符串替换、字符串分割的方法。