第六课--字符串

任务目标

1、字符串 等价 字符数组

2、String类常见的方法

3、StringBuilder类常见的方法

相关知识

1、字符数组的操作-字符串操作类型

2、字符串

3、动态字符串

1、字符串

- [ ] 直接赋值
- [ ] 通过new String()创建字符串
- [ ] String类的length()方法
- [ ] String类的charAt()方法

```
public class Test1
{
    public static void main(String[] args)
    {
        String str1 = "Java is cool";
        String str2 = new String("Java is cool");
        System.out.print(str1.length()+"\n");
        System.out.print(str2.length()+"\n");
        System.out.print(str2.charAt(5)+"\n");

    }
}
```

2、判断回文字符串

1、取出字符串的第一个和最后一个比较，若不相同，程序结束，返回false。若相等，比较第二个字符和倒数第二字符，直到比较到字符串的中间字符为止，若都相等，则是回文，返回true。

```
import java.util.Scanner;
public class Test2
{
   public static boolean isPalindrome(String s)
   {
        boolean flag =true;
        for(int i=0;i<s.length()/2;i++)
        {
            if(s.charAt(i)!=s.charAt(s.length()-1-i))
            {
                flag =false;
                break;
            }
        }
        return flag;
    }

    public static void main(String[] args)
    {
        String str = "helleho";
        System.out.print(isPalindrome(str)+"\n");
    }
}
```

- [ ] int indexOf(int ch)
- [ ] int lastIndexOf(int ch)
- [ ] bytes[] getBytes()
- [ ] bytes[] getBytes(String charsetName)

```
import java.util.Scanner;
public class Test3
{
    public static void main(String[] args)
    {
        String str = "helleho";
        System.out.print(str.indexOf('o')+"\n");
        System.out.print(str.lastIndexOf('l')+"\n");
    }
}
```

3、将字符串转化为char数组

- [ ] char[] toCharArray()
- [ ] getChars(int begin,int end,char[] dst, int dstBegin)
- [ ] bytes[] getBytes()
- [ ] bytes[] getBytes(String charsetName)

```
import java.util.Scanner;
public class Test4
{
    public static void main(String[] args)
    {
        String str = "helleho";
        char[] array = str.toCharArray();
        for(int i=0;i<array.length;i++)
        {
        System.out.print(array[i]+"\t");
        }
    }
}
```

4、字符串的比较

- [ ] boolean equals(String str)
- [ ] ==
- [ ] int compareTo(String str)

```
import java.util.Scanner;
public class Test5
{
    public static void main(String[] args)
    {
        String str = "helleho";
        String str2 = "helleho";
        System.out.println(str.equals(str2));
        System.out.println(str==str2);
    }
}
```

5、字符串包含

- [ ] boolean startsWith(String prefix)
- [ ] boolean endsWith(String suffix)
- [ ] boolean contains(String str)

```
import java.util.Scanner;
public class Test6
{
    public static void main(String[] args)
    {
        String s1 = "ABC";
        String s2 = "ABE";
        System.out.println(s1.compareTo(s2));
        System.out.println(s1.startsWith("A"));
        System.out.println(s1.endsWith("C"));
        System.out.println(s1.contains("BC"));
    }
}
```

6、字符串的替换

- [ ] String replace('c','c')
- [ ] String substring(begin,end)
- [ ] String concat("String")
- [ ] String toUpperCase()
- [ ] String toLowerCase()

```
import java.util.Scanner;
public class Test7
{
    public static void main(String[] args)
    {
    String s = "Hello,world";
    s = s.replace('o','A');
    System.out.println(s);
    s = s.substring(0,6).concat("Java");
    System.out.println(s);
    s = s.toUpperCase();
    System.out.println(s);
    s = s.toLowerCase();
    System.out.println(s);
    }
}
```

7、字符串的拆分和组合

- [ ] String[] split(String regex)
- [ ] public boolean matches(String regex)
- [ ] public static String join()

```
//字符串的拆分
import java.util.Scanner;
public class Test2
{
    public static void main(String[] args)
    {
        String str = "1233333,john,information,23";
        String s[] = str.split(",");
        for(int i=0;i<s.length;i++)
        {
            System.out.print(s[i]+"\t");
        }
    }
}

//正则表达式
import java.util.Scanner;
public class Test2
{
    public static void main(String[] args)
    {
        // String str = "zhang@163.com";
        String str = "hello333@sina.com";
        if(str.matches("[a-z]+\\@[1-9]+\\.[a-z]+"))
        {
            System.out.print(str+"\t is email address");
        }
        else
        {
            System.out.print(str +"\t isn't email address");
        }


    }
}

//字符串重新组合
import java.util.Scanner;
public class Test2
{
    public static void main(String[] args)
    {
        String str = "1233333,john,information,23";
        String s[] = str.split(",");
        for(int i=0;i<s.length;i++)
        {
            System.out.print(s[i]+"\t");
        }
        str = String.join(":",s);
        {
            System.out.print(str);
        }
    }
}
```

8、字符串的格式化输出

- [ ] PrintStream printf(String format, Object... args)
- [ ] static String format(String format, Object ... args)

```
import java.util.Scanner;
public class Test8
{
    public static void main(String[] args)
    {
    String s = "Hello,world";
    System.out.printf("%s",s);
    System.out.format("%d",3);
    }
}
```

8、StringBuilder (可变字符串)

- [ ] length()、charAt()、indexOf()、substring()
- [ ] int capacity()
- [ ] void setCharAt(int index, char ch)
- [ ] StringBuilder append(int n)
- [ ] StringBuilder append(String str)
- [ ] StringBuilder insert(int offset, int n)
- [ ] StringBuilder insert(int offset, String str)

```
import java.util.Scanner;
public class Test9
{
    public static void main(String[] args)
    {
    StringBuilder str = new StringBuilder("Hello");
    System.out.print(str.capacity());
    String.format("%d",3);
    }
}
```

二、字符串数组

- [ ] 包含字符串的数组成为字符串数组。
- [ ] 使用冒泡排序对字符串进行排序

```
import java.util.Scanner;
public class Test1
{
    public static String[] bubble(String[] s)
    {
        String t;
        for(int i=0;i<s.length-1;i++)
        {
            for(int j=s.length-1;j>i;j--)
            {
                if(s[j].compareTo(s[j-1])<0)
                {
                    t = s[j];
                    s[j] =s[j-1];
                    s[j-1]=t;
                }
            }
        }
        return s;
    }
    public static void main(String[] args)
    {
        String[] str = {"C++","Java","C","Python","Ruby","Go","PHP"};
        str = bubble(str);
        for(int i=0;i<str.length;i++)
        {
            System.out.print(str[i]+"\t");
        }
    }
}
```

三、正则表达式

- [ ] String类的boolean matches(String regex)，字符串的匹配
- [ ] String类的String replaceFirst(String regex, String replacement)
- [ ] String类的String replaceAll(String regex, String replacement)
- [ ] String类的String[] split(String regex)
- [ ] String类的String[] split(String regex, int limit)
- [ ] Pattern类，是一个正则表达式的编译表示。
- [ ] Matcher类，是对输入字符串进行解释和匹配操作的引擎
- [ ] PatternSyntaxException，是一个非强制异常类

1、许多时候只需要临时使用某个正则表达式，而不需要重复使用。每次都生成Pattern对象和Matcher对象再操作显得很烦琐。这里的匹配指的并不是regex能否在String内找到匹配，而是指regex匹配整个String对象，非常适合用来做数据校验。字符串的类型匹配，字符串与Java的正则表达式匹配，其中\d、\D、\w、\W、\s、\S都是使用ASCII匹配规则。\d等价于[0-9]，\w等价于[0-9a-zA-Z]，\s无法匹配ASCII编码之外的空白字符。

2、boolean matches(String regex)

3、String replaceFirst(String regex, String replacement)

4、String replaceAll(String regex, String replacement)

5、String[] split(String regex)

6、String[] split(String regex,int limit)

```
public class Test61
{
    public static void main(String[] args)
    {
        if("1a".matches("\\d"))
        {
            System.out.print("True");
        }
        else
        {
            System.out.print("False");
        }
    }
}

public class Test61
{
    public static void main(String[] args)
    {
        if(" ".matches("\\s"))
        {
            System.out.print("True");

        }
        else
        {
            System.out.print("False");
        }
    }
}

public class Test61
{
    public static void main(String[] args)
    {
        String str  = "2020-10-30 2019-12-23".replaceFirst("\\d{4}-\\d{2}-
\\d{2}","Year");
        System.out.println(str);
        str  = "2020-10-30 2019-12-23".replaceFirst("(\\d{4})-(\\d{2})-
(\\d{2})","$2/$3/$1");
        System.out.print(str);
    }
}

public class Test61
{
    public static void main(String[] args)
    {
        String str  = "2020-10-30 2019-12-23".replaceAll("\\d{4}-\\d{2}-
\\d{2}","Year");
```

```
        System.out.println(str);
        str  = "2020-10-30 2019-12-23".replaceAll("(\\d{4})-(\\d{2})-
(\\d{2})","$2/$3/$1");
        System.out.print(str);
    }
}

public class Test61
{
    public static void main(String[] args)
    {
        String[] str  = "2020-10-30, 2019-12-23".split(",");
        for(String c: str)
        {
            System.out.println(c);
        }
    }
}

public class Test61
{
    public static void main(String[] args)
    {
        String[] str  = "2020-10-30,2019-12-23,2018-10-30,2017-12-23,2016-10-
30,2015-12-23".split(",",3);
        for(String c: str)
        {
            System.out.println(c);
        }
    }
}
```

2、使用Pattern、Matcher类进行字符串提取。

常量	**修饰符**	**说明**
CASE_INSENSITIVE	i	不区分ASCII字符的大小写
COMMENTS	x	允许正则表达式中出现的注释，表达式中的空白字符，以及#开始到行末的文本，都视为注释
MULTILINE	m	允许^和$不仅匹配字符串的起始和结束位置，还可以匹配字符串内部 文本行的起始和结束位置
DOTALL	s	允许点号.匹配任何字符，包括换行符
UNICODE_CASE	u	可以识别Unicode字符的不同形态，“不仅区分大小写”的范围不限于ASCII字符，但严重影响性能
UNIX_LINES	d	限定. ^ $能识别的行终结符只有换行符\n，忽略\r \n和Unicode行终结符等其他字符
CANON_EQ	无	匹配时采取Unicode “等价”规则，可以识别意义相等的复合字符（单个字符加上调号）与单个字符；但此选项会严重影响性能

```
import java.util.regex.*;
public class Test61
{
    public static void main(String[] args)
    {
        if("A".matches("a"))
        {
            System.out.println("True");
        }
        if("A".matches("(?i)a"))
        {
          System.out.println("True");
        }
        if(Pattern.compile("a",Pattern.CASE_INSENSITIVE).matcher("A").find())
        {
              System.out.println("True");
        }
        if("aBb".matches("a(?i)b(?-i)b"))
        {
              System.out.println("True");
        }
        if("aBB".matches("a(?i)b(?-i)b"))
        {
              System.out.println("True");
        }
        else
        {
             System.out.println("False");
        }
    }
}

import java.util.regex.*;
public class Test61
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\d{4}-\\d{2}-\\d{2}");
        String str = "2010-12-21 zhang wang 2019-09-20";
        Matcher matcher = p.matcher(str);
        while(matcher.find())
        {
            System.out.println(matcher.group(0));
        }
    }
}
```

3、字符串验证

```
import java.util.regex.*;
public class Test61
{
    public static void main(String[] args)
    {
        boolean f1 = "2020-10-30".matches("\\d{4}-\\d{2}-\\d{2}");
        boolean f2 = Pattern.matches("\\d{4}-\\d{2}-\\d{2}","2020-10-30");
        System.out.print(f1+"\t");
        System.out.print(f2+"\t");
    }
}

import java.util.regex.*;
public class Test61
{
    public static void main(String[] args)
    {
        String str = "2020-01-30,2020-02-30,2020-03-30,2020-04-30,2020-05-
30,hello java";
        Pattern p = Pattern.compile("\\d{4}-\\d{2}-\\d{2}");
        Matcher m = p.matcher(str);
        while(m.find())
        {
            System.out.println(m.group());
        }
       }
}

import java.util.regex.*;
public class Test61
{
    public static void main(String[] args)
    {
        String str = "2020-01-30,2020-02-30,2020-03-30,2020-04-30,2020-05-
30,hello java";
        Pattern p = Pattern.compile("(\\d{4})-(\\d{2})-(\\d{2})");
        Matcher m = p.matcher(str);
        while(m.find())
        {
            System.out.print(m.group(0)+"\t");
            System.out.print(m.group(1)+"\t");
            System.out.print(m.group(2)+"\t");
            System.out.println(m.group(3)+"\t");
        }
       }
}
```