day06【String和StringBuilder】

今日内容

- String
- StringBuilder

教学目标

- [ ] 能够使用String类的构造方法创建字符串对象
- [ ] 能够明确String类的构造方法创建对象,和直接赋值创建字符串对象的区别
- [ ] 能够说出String类常用方法的功能及使用
- [ ] 能够知道String和StringBuilder的区别
- [ ] 能够完成String和StringBuilder的相互转换
- [ ] 能够使用StringBuilder完成字符串的拼接
- [ ] 能够使用StringBuilder完成字符串的反转

第一章 String类

1.1 String类概述

`java.lang.String` 类代表字符串。Java程序中所有的字符串文字（例如 `"abc"` ）都可以被看作是实现此类的实例。类 `String` 中包括用于检查各个字符串的方法，比如用于**比较**字符串，**搜索**字符串，**提取**子字符串以及创建具有翻译为**大写**或**小写**的所有字符的字符串的副本。

1.2 使用步骤

- 查看类
  - `java.lang.String` ：此类不需要导入。
- 查看构造方法
  - String() 创建一个空的字符串对象
  - String(String original) 根据字符串来创建一个字符串对象
  - String(char[] value) 通过字符数组来创建字符串对象
  - String(byte[] bytes) 通过字节数组来构造新的字符串对象
  - String(byte[] bytes, int offset, int length) 通过字节数组一部分来构造新的字符串对象
- 构造举例，代码如下：

```
public static void main(String[] args) {
    // 1.String() 创建一个空的字符串对象
    String str1 = new String();
    System.out.println("str1:" + str1);

    // 2.String(String original) 根据字符串来创建一个字符串对象
    String str2 = new String("abc");
    System.out.println("str2:" + str2);
```

```
    // 3.String(char[] value) 通过字符数组来创建字符串对象
    char[] chs = {'A', 'B', 'C', 'D', 'E'};
    String str5 = new String(chs);
    System.out.println("str5:" + str5);

    // 4.String(byte[] bytes) 通过字节数组来构造新的字符串对象
    byte[] bytes = {97, 98, 99, 100, 101}; // a, b, c, d, e
    String str3 = new String(bytes);
    System.out.println("str3:" + str3);

    // 5.String(byte[] bytes, int offset, int length) 通过字节数组一部分来构造新的字符串对象
    String str4 = new String(bytes, 1, 2);
    System.out.println("str4:" + str4);

    String str6 = "abcde";
    System.out.println("str6:" + str6);
}
```

1.3 创建字符串对象两种方式的区别

- 字符串不变：字符串的值在创建后不能被更改。

```
String s1 = "abc";
s1 += "d";
System.out.println(s1); // "abcd"
// 内存中有"abc"，"abcd"两个对象，s1从指向"abc"，改变指向，指向了"abcd"。
```

- 因为String对象是不可变的，所以它们可以被共享。

```
String s1 = "abc";
String s2 = "abc";
// 内存中只有一个"abc"对象被创建，同时被s1和s2共享。
```

- 通过构造方法创建

  通过 new 创建的字符串对象，每一次 new 都会申请一个内存空间，虽然内容相同，但是地址值不同

- 直接赋值方式创建

  以“”方式给出的字符串，只要字符序列相同(顺序和大小写)，无论在程序代码中出现几次，JVM 都只会建立一个 String 对象，并在字符串池中维护

1.4 常用方法

1.4.1 判断功能的方法

- `public boolean equals (Object anObject)`：将此字符串与指定对象进行比较。
- `public boolean equalsIgnoreCase (String anotherString)`：将此字符串与指定对象进行比较，忽略大小写。
- `public boolean contains (CharSequence s)`：判断参数字符串在当前字符串中是否存在(区分大小写)。存在，返回true，否则，返回false。

- `public boolean endsWith(String suffix)`：测试此字符串是否以指定的后缀结尾(区分大小写)。
- `public boolean startsWith(String prefix)`：测试此字符串是否以指定的前缀开始(区分大小写)

  方法演示，代码如下：

```
public class String_Demo01 {
    public static void main(String[] args) {
        // 创建字符串对象
        String s1 = "hello";
        String s2 = "hello";
        String s3 = "HELLO";

        // boolean equals(Object obj):比较字符串的内容是否相同
        System.out.println(s1.equals(s2)); // true
        System.out.println(s1.equals(s3)); // false
        System.out.println("-----------");

        //boolean equalsIgnoreCase(String str):比较字符串的内容是否相同,忽略大小
写
        System.out.println(s1.equalsIgnoreCase(s2)); // true
        System.out.println(s1.equalsIgnoreCase(s3)); // true
        System.out.println("-----------");

        String s = "我爱Java，我爱学习！";
        System.out.println("字符串中是否包含Java：" +
s.contains("Java"));//true
        System.out.println("字符串中是否包含java：" +
s.contains("java"))//false

        String name = "Test.java";
        System.out.println("判断name是否以java结尾：" +
name.endsWith("java"));//true
        System.out.println("判断name是否以Java结尾：" +
name.endsWith("Java"));//false

        String name = "我爱Java";
        System.out.println("字符串是否以‘我’开头：" +
name.startsWith("我"));//true
    }
}
```

> Object 是” 对象”的意思，也是一种引用类型。作为参数类型，表示任意对 象都可以传递到方法中。

1.4.2 获取功能的方法

- `public int length ()`：返回此字符串的长度。
- `public String concat (String str)`：将指定的字符串连接到该字符串的末尾。
- `public char charAt (int index)`：返回指定索引处的 char值。
- `public int indexOf (String str)`：返回指定子字符串第一次出现在该字符串内的索引。
- `public int lastIndexOf(String str)`：返回指定子字符串最后一次出现的字符串中的索引。 如果不包含，则返回-1。

- `public String substring (int beginIndex)`：返回一个子字符串，从beginIndex开始截取字符串到字符串结尾。
- `public String substring (int beginIndex, int endIndex)`：返回一个子字符串，从beginIndex到endIndex截取字符串。含beginIndex，不含endIndex。

方法演示，代码如下：

```
public class String_Demo02 {
    public static void main(String[] args) {
        //创建字符串对象
        String s = "helloworld";

        // int length():获取字符串的长度，其实也就是字符个数
        System.out.println(s.length());
        System.out.println("--------");

        // String concat (String str):将将指定的字符串连接到该字符串的末尾.
        String s = "helloworld";
        String s2 = s.concat("**hello itheima");
        System.out.println(s2);// helloworld**hello itheima

        // char charAt(int index):获取指定索引处的字符
        System.out.println(s.charAt(0));
        System.out.println(s.charAt(1));
        System.out.println("--------");

        // int indexOf(String str):获取str在字符串对象中第一次出现的索引,没有返回-1
        System.out.println(s.indexOf("l"));
        System.out.println(s.indexOf("owo"));
        System.out.println(s.indexOf("ak"));
        System.out.println("--------");

        // int lastIndexOf(String str):获取str在字符串对象中最后一次出现的索引,没有返
回-1
        System.out.println(s.lastIndexOf("l"));
        System.out.println(s.lastIndexOf("owo"));
        System.out.println(s.lastIndexOf("ak"));
        System.out.println("--------");

        // String substring(int start):从start开始截取字符串到字符串结尾
        System.out.println(s.substring(0));
        System.out.println(s.substring(5));
        System.out.println("--------");

        // String substring(int start,int end):从start到end截取字符串。含start，不含
end。
        System.out.println(s.substring(0, s.length()));
        System.out.println(s.substring(3,8));
    }
}
```

1.4.3 转换功能的方法

- `public char[] toCharArray ()`：将此字符串转换为新的字符数组。
- `public byte[] getBytes ()`：使用平台的默认字符集将该 String编码转换为新的字节数组。
- `public String toLowerCase()`：使用默认语言环境的规则将此 String所有字符转换为小写。

- `public String toUpperCase()`：将此 String所有字符转换为大写，使用默认语言环境的规则。
- `public String replace (CharSequence target, CharSequence replacement)`：将与target匹配的字符串使用replacement字符串替换。

方法演示，代码如下：

```
public class String_Demo03 {
    public static void main(String[] args) {
        //创建字符串对象
        String s = "abcde";

        // char[] toCharArray():把字符串转换为字符数组
        char[] chs = s.toCharArray();
        for(int x = 0; x < chs.length; x++) {
            System.out.println(chs[x]);
        }
        System.out.println("-----------");

        // byte[] getBytes ():把字符串转换为字节数组
        byte[] bytes = s.getBytes();
        for(int x = 0; x < bytes.length; x++) {
            System.out.println(bytes[x]);
        }
        System.out.println("-----------");

        // 替换字母it为大写IT
        String str = "itcast itheima";
        String replace = str.replace("it", "IT");
        System.out.println(replace); // ITcast ITheima
        System.out.println("-----------");
    }
}
```

> CharSequence 是一个接口，也是一种引用类型。作为参数类型，可以把String对象传递到方法中。

1.4.4 分割功能的方法

- `public String[] split(String regex)`：将此字符串按照给定的regex（规则）拆分为字符串数组。
- `public String trim()`：去掉当前字符串的前后空格，并返回一个新字符串，原字符串不变。

方法演示，代码如下：

```
public class String_Demo03 {
    public static void main(String[] args) {
        //创建字符串对象
        String s = "aa,bb,cc";
        String[] strArray = s.split(","); // ["aa","bb","cc"]
        for(int x = 0; x < strArray.length; x++) {
            System.out.println(strArray[x]); // aa bb cc
        }

        String str = "  ad  min     ";
```

```
        System.out.println("去掉前后空格后|" + str.trim() + "|");//去掉前后空格后|ad
min|
        System.out.println("原字符串|" + str + "|");//原字符串|  ad  min     |
    }
}
```

1.5 String类的练习

1.5.1 统计字符个数

键盘录入一个字符，统计字符串中大小写字母及数字字符个数

```
public class StringTest2 {
    public static void main(String[] args) {
        //键盘录入一个字符串数据
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入一个字符串数据：");
        String s = sc.nextLine();

        //定义三个统计变量，初始化值都是0
        int bigCount = 0;
        int smallCount = 0;
        int numberCount = 0;

        //遍历字符串，得到每一个字符
        for(int x=0; x<s.length(); x++) {
            char ch = s.charAt(x);
            //拿字符进行判断
            if(ch>='A'&&ch<='Z') {
                bigCount++;
            }else if(ch>='a'&&ch<='z') {
                smallCount++;
            }else if(ch>='0'&&ch<='9') {
                numberCount++;
            }else {
                System.out.println("该字符"+ch+"非法");
            }
        }

        //输出结果
        System.out.println("大写字符："+bigCount+"个");
        System.out.println("小写字符："+smallCount+"个");
        System.out.println("数字字符："+numberCount+"个");
    }
}
```

第二章 StringBuilder类

2.1 StringBuilder类概述

StringBuilder 是一个可变的字符串类，我们可以把它看成是一个容器，这里的可变指的是 StringBuilder 对象中的内容是可变的

2.2 StringBuilder类和String类的区别

- String类：内容是不可变的
- StringBuilder类：内容是可变的

2.3 StringBuilder类的构造方法

- 常用的构造方法

方法名	说明
public StringBuilder()	创建一个空白可变字符串对象，不含有任何内容
public StringBuilder(String str)	根据字符串的内容，来创建可变字符串对象

- 示例代码

```
public class StringBuilderDemo01 {
    public static void main(String[] args) {
        //public StringBuilder()：创建一个空白可变字符串对象，不含有任何内容
        StringBuilder sb = new StringBuilder();
        System.out.println("sb:" + sb);
        System.out.println("sb.length():" + sb.length());

        //public StringBuilder(String str)：根据字符串的内容，来创建可变字符串对象
        StringBuilder sb2 = new StringBuilder("hello");
        System.out.println("sb2:" + sb2);
        System.out.println("sb2.length():" + sb2.length());
    }
}
```

2.4 StringBuilder类添加和反转方法

- 添加和反转方法

方法名	说明
public StringBuilder append(任意类型)	添加数据，并返回对象本身
public StringBuilder reverse()	返回相反的字符序列

- 示例代码

```
public class StringBuilderDemo01 {
    public static void main(String[] args) {
        //创建对象
        StringBuilder sb = new StringBuilder();

        //public StringBuilder append(任意类型)：添加数据，并返回对象本身
//        StringBuilder sb2 = sb.append("hello");
//
//        System.out.println("sb:" + sb);
//        System.out.println("sb2:" + sb2);
//        System.out.println(sb == sb2);

//        sb.append("hello");
```

```
//        sb.append("world");
//        sb.append("java");
//        sb.append(100);

        //链式编程
        sb.append("hello").append("world").append("java").append(100);

        System.out.println("sb:" + sb);

        //public StringBuilder reverse()：返回相反的字符序列
        sb.reverse();
        System.out.println("sb:" + sb);
    }
}
```

2.5 StringBuilder和String相互转换

- StringBuilder转换为String

  public String toString()：通过 toString() 就可以实现把 StringBuilder 转换为 String

- String转换为StringBuilder

  public StringBuilder(String s)：通过构造方法就可以实现把 String 转换为 StringBuilder

- 示例代码

```
public class StringBuilderDemo02 {
    public static void main(String[] args) {
        /*
        //StringBuilder 转换为 String
        StringBuilder sb = new StringBuilder();
        sb.append("hello");

        //String s = sb; //这个是错误的做法

        //public String toString()：通过 toString() 就可以实现把 StringBuilder 转换
为 String
        String s = sb.toString();
        System.out.println(s);
        */

        //String 转换为 StringBuilder
        String s = "hello";

        //StringBuilder sb = s; //这个是错误的做法

        //public StringBuilder(String s)：通过构造方法就可以实现把 String 转换为
StringBuilder
        StringBuilder sb = new StringBuilder(s);

        System.out.println(sb);
    }
}
```

2.6 字符串拼接

2.6.1 案例需求

定义一个方法，把 int 数组中的数据按照指定的格式拼接成一个字符串返回，调用该方法，

并在控制台输出结果。例如，数组为int[] arr = {1,2,3}; ，执行方法后的输出结果为：[1, 2, 3]

2.6.2 代码实现

```
/*
    思路：
        1:定义一个 int 类型的数组，用静态初始化完成数组元素的初始化
        2:定义一个方法，用于把 int 数组中的数据按照指定格式拼接成一个字符串返回。
          返回值类型 String，参数列表 int[] arr
        3:在方法中用 StringBuilder 按照要求进行拼接，并把结果转成 String 返回
        4:调用方法，用一个变量接收结果
        5:输出结果
 */
public class StringBuilderTest01 {
    public static void main(String[] args) {
        //定义一个 int 类型的数组，用静态初始化完成数组元素的初始化
        int[] arr = {1, 2, 3};

        //调用方法，用一个变量接收结果
        String s = arrayToString(arr);

        //输出结果
        System.out.println("s:" + s);

    }

    //定义一个方法，用于把 int 数组中的数据按照指定格式拼接成一个字符串返回
    /*
        两个明确：
            返回值类型：String
            参数：int[] arr
     */
    public static String arrayToString(int[] arr) {
        //在方法中用 StringBuilder 按照要求进行拼接，并把结果转成 String 返回
        StringBuilder sb = new StringBuilder();

        sb.append("[");

        for(int i=0; i<arr.length; i++) {
            if(i == arr.length-1) {
                sb.append(arr[i]);
            } else {
                sb.append(arr[i]).append(", ");
            }
        }

        sb.append("]");

        String s = sb.toString();

        return  s;
    }
}
```