第08天 常用API

今日内容介绍

* API概述
* Scanner类与String类
* StringBuilder类

# API概述

## API概念

API(Application Programming Interface) : 应用程序编程接口

编写一个机器人程序去控制机器人踢足球，程序就需要向机器人发出向前跑、向后跑、射门、抢球等各种命令，没有编过程序的人很难想象这样的程序如何编写。但是对于有经验的开发人员来说，知道机器人厂商一定会提供一些用于控制机器人的Java类，这些类中定义好了操作机器人各种动作的方法。其实，这些Java类就是机器人厂商提供给应用程序编程的接口，大家把这些类称为API。本章涉及的Java API指的就是JDK中提供的各种功能的Java类

## 快速使用API步骤:

A:打开帮助文档

B:点击显示，找到索引，看到输入框

C:你要学习什么内容，你就在框框里面输入什么内容

举例：Random

D:看包

java.lang包下的类在使用的时候是不需要导包的

E:看类的描述

Random类是用于生成随机数的类

F:看构造方法

Random():无参构造方法

Random r = new Random();

G:看成员方法

public int nextInt(int n):产生的是一个[0,n)范围内的随机数

调用方法：

看返回值类型：人家返回什么类型，你就用什么类型接收

看方法名：名字不要写错了

看形式参数：人家要几个参数，你就给几个，人家要什么数据类型的，你就给什么数据类型的

int number = r.nextInt(100);

# Scanner类 与 String类

## Scanner类

### Scanner类作用

用Scanner类的方法可以完成接收键盘录入的数据

### Scanner类接受键盘录入的字符串

#### 案例代码一:

package com.itheima\_01;

import java.util.Scanner;

/\*

\* Scanner:用于获取键盘录入的数据。(基本数据类型，字符串数据)

\* public String nextLine():获取键盘录入的字符串数据

\*/

public class ScannerDemo {

public static void main(String[] args) {

//创建键盘录入对象

Scanner sc = new Scanner(System.in);

//接收数据

System.out.println("请输入一个字符串数据：");

String s = sc.nextLine();

//输出结果

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

}

}

## String类

### String类概述

通过JDK提供的API，查看String类的说明

A:"abc"是String类的一个实例,或者成为String类的一个对象

B:字符串字面值"abc"也可以看成是一个字符串对象

C:字符串是常量，一旦被赋值，就不能被改变

D:字符串本质是一个字符数组

### String类的构造方法

String(String original):把字符串数据封装成字符串对象

String(char[] value):把字符数组的数据封装成字符串对象

String(char[] value, int index, int count):把字符数组中的一部分数据封装成字符串对象

#### 常用构造方法演示

##### 案例代码二:

package com.itheima\_02;

/\*

\* String:字符串类

\* 由多个字符组成的一串数据

\* 字符串其本质是一个字符数组

\*

\* 构造方法：

\* String(String original):把字符串数据封装成字符串对象

\* String(char[] value):把字符数组的数据封装成字符串对象

\* String(char[] value, int index, int count):把字符数组中的一部分数据封装成字符串对象

\*

\* 注意：字符串是一种比较特殊的引用数据类型，直接输出字符串对象输出的是该对象中的数据。

\*/

public class StringDemo {

public static void main(String[] args) {

//方式1

//String(String original):把字符串数据封装成字符串对象

String s1 = new String("hello");

System.out.println("s1:"+s1);

System.out.println("---------");

//方式2

//String(char[] value):把字符数组的数据封装成字符串对象

char[] chs = {'h','e','l','l','o'};

String s2 = new String(chs);

System.out.println("s2:"+s2);

System.out.println("---------");

//方式3

//String(char[] value, int index, int count):把字符数组中的一部分数据封装成字符串对象

//String s3 = new String(chs,0,chs.length);

String s3 = new String(chs,1,3);

System.out.println("s3:"+s3);

System.out.println("---------");

//方式4

String s4 = "hello";

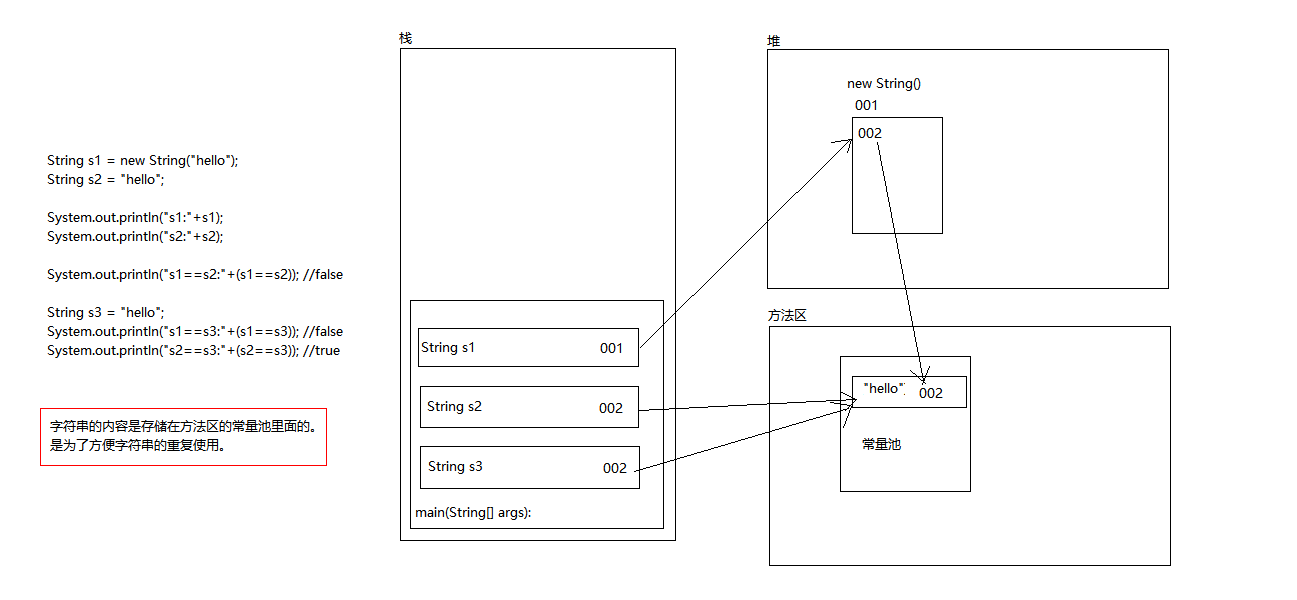
System.out.println("s4:"+s4);

}

}

#### 创建字符串对象两种方式的区别

##### 案例代码三:



package com.itheima\_02;

/\*

\* 通过构造方法创建的字符串对象和直接赋值方式创建的字符串对象有什么区别呢?

\* 通过构造方法创建字符串对象是在堆内存。

\* 直接赋值方式创建对象是在方法区的常量池。

\*

\* ==:

\* 基本数据类型：比较的是基本数据类型的值是否相同

\* 引用数据类型：比较的是引用数据类型的地址值是否相同

\*/

public class StringDemo2 {

public static void main(String[] args) {

String s1 = new String("hello");

String s2 = "hello";

System.out.println("s1:"+s1);

System.out.println("s2:"+s2);

System.out.println("s1==s2:"+(s1==s2)); //false

String s3 = "hello";

System.out.println("s1==s3:"+(s1==s3)); //false

System.out.println("s2==s3:"+(s2==s3)); //true

}

}

### String类的判断功能

boolean equals(Object obj):比较字符串的内容是否相同

boolean equalsIgnoreCase(String str):比较字符串的内容是否相同,忽略大小写

boolean startsWith(String str):判断字符串对象是否以指定的str开头

boolean endsWith(String str):判断字符串对象是否以指定的str结尾

#### 判断方法演示

##### 案例代码四:

package com.itheima\_03;

/\*

\* Object:是类层次结构中的根类，所有的类都直接或者间接的继承自该类。

\* 如果一个方法的形式参数是Object，那么这里我们就可以传递它的任意的子类对象。

\*

\* String类的判断功能：

\* boolean equals(Object obj):比较字符串的内容是否相同

\* boolean equalsIgnoreCase(String str):比较字符串的内容是否相同,忽略大小写

\* boolean startsWith(String str):判断字符串对象是否以指定的str开头

\* boolean endsWith(String str):判断字符串对象是否以指定的str结尾

\*/

public class StringDemo {

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));

System.out.println(s1.equals(s3));

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));

System.out.println(s1.equalsIgnoreCase(s3));

System.out.println("-----------");

//boolean startsWith(String str):判断字符串对象是否以指定的str开头

System.out.println(s1.startsWith("he"));

System.out.println(s1.startsWith("ll"));

}

}

#### 判断功能案例

需求:模拟登录,给三次机会,并提示还有几次

##### 案例代码五:

package com.itheima\_03;

import java.util.Scanner;

/\*

\* 模拟登录,给三次机会,并提示还有几次。

\*

\* 分析：

\* A:定义两个字符串对象，用于存储已经存在的用户名和密码

\* B:键盘录入用户名和密码

\* C:拿键盘录入的用户名和密码和已经存在的用户名和密码进行比较

\* 如果内容相同，提示登录成功

\* 如果内容不同，提示登录失败，并提示还有几次机会

\*/

public class StringTest {

public static void main(String[] args) {

//定义两个字符串对象，用于存储已经存在的用户名和密码

String username = "admin";

String password = "admin";

//给三次机会，用for循环实现

for(int x=0; x<3; x++) {

//键盘录入用户名和密码

Scanner sc = new Scanner(System.in);

System.out.println("请输入用户名：");

String name = sc.nextLine();

System.out.println("请输入密码：");

String pwd = sc.nextLine();

//拿键盘录入的用户名和密码和已经存在的用户名和密码进行比较

if(username.equals(name) && password.equals(pwd)) {

System.out.println("登录成功");

break;

}else {

if((2-x) ==0) {

System.out.println("用户名和密码被锁定,请与管理员联系");

}else {

System.out.println("登录失败,你还有"+(2-x)+"次机会"); //2,1,0

}

}

}

}

}

### String类的获取功能

#### 获取方法演示

package com.itheima\_04;

/\*

\* String类的获取功能：

\* int length():获取字符串的长度，其实也就是字符个数

\* char charAt(int index):获取指定索引处的字符

\* int indexOf(String str):获取str在字符串对象中第一次出现的索引

\* String substring(int start):从start开始截取字符串

\* String substring(int start,int end):从start开始，到end结束截取字符串。包括start，不包括end

\*/

public class StringDemo {

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("--------");

//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

System.out.println(s.indexOf("l"));

System.out.println(s.indexOf("owo"));

System.out.println(s.indexOf("ak"));

System.out.println("--------");

//String substring(int start):从start开始截取字符串

System.out.println(s.substring(0));

System.out.println(s.substring(5));

System.out.println("--------");

//String substring(int start,int end):从start开始，到end结束截取字符串

System.out.println(s.substring(0, s.length()));

System.out.println(s.substring(3,8));

}

}

#### 获取功能案例

##### 案例代码六:

package com.itheima\_04;

/\*

\* 遍历字符串(获取字符串中的每一个字符)

\*/

public class StringTest {

public static void main(String[] args) {

//创建一个字符串对象

String s = "abcde";

//原始做法

System.out.println(s.charAt(0));

System.out.println(s.charAt(1));

System.out.println(s.charAt(2));

System.out.println(s.charAt(3));

System.out.println(s.charAt(4));

System.out.println("---------");

//用for循环改进

for(int x=0; x<5; x++) {

System.out.println(s.charAt(x));

}

System.out.println("---------");

//用length()方法获取字符串的长度

for(int x=0; x<s.length(); x++) {

System.out.println(s.charAt(x));

}

}

}

##### 案例代码七:

package com.itheima\_04;

import java.util.Scanner;

/\*

\* 统计一个字符串中大写字母字符，小写字母字符，数字字符出现的次数。(不考虑其他字符)

\*

\* 分析：

\* A:键盘录入一个字符串数据

\* B:定义三个统计变量，初始化值都是0

\* C:遍历字符串，得到每一个字符

\* D:拿字符进行判断

\* 假如ch是一个字符。

\* 大写：ch>='A' && ch<='Z'

\* 小写：ch>='a' && ch<='z'

\* 数字：ch>='0' && ch<='9'

\* E:输出结果

\*/

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");

}

}

### String类的转换功能

#### 转换方法演示

char[] toCharArray():把字符串转换为字符数组

String toLowerCase():把字符串转换为小写字符串

String toUpperCase():把字符串转换为大写字符串

##### 案例代码八:

package com.itheima\_05;

/\*

\* String类的转换功能：

\* char[] toCharArray():把字符串转换为字符数组

\* String toLowerCase():把字符串转换为小写字符串

\* String toUpperCase():把字符串转换为大写字符串

\*

\* 字符串的遍历：

\* A:length()加上charAt()

\* B:把字符串转换为字符数组，然后遍历数组

\*/

public class StringDemo {

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("-----------");

//String toLowerCase():把字符串转换为小写字符串

System.out.println("HelloWorld".toLowerCase());

//String toUpperCase():把字符串转换为大写字符串

System.out.println("HelloWorld".toUpperCase());

}

}

#### 转换功能案例

##### 案例代码九:

package com.itheima\_05;

import java.util.Scanner;

/\*

\* 键盘录入一个字符串，把该字符串的首字母转成大写，其余为小写。(只考虑英文大小写字母字符)

\*

\* 分析：

\* A:键盘录入一个字符串

\* B:截取首字母

\* C:截取除了首字母以外的字符串

\* D:B转大写+C转小写

\* E:输出即可

\*/

public class StringTest {

public static void main(String[] args) {

//键盘录入一个字符串

Scanner sc = new Scanner(System.in);

System.out.println("请输入一个字符串：");

String s = sc.nextLine();

//截取首字母

String s1 = s.substring(0, 1);

//截取除了首字母以外的字符串

String s2 = s.substring(1);

//B转大写+C转小写

String s3 = s1.toUpperCase()+s2.toLowerCase();

//输出即可

System.out.println("s3:"+s3);

}

}

### String类的其它功能

#### 其它方法演示

##### 案例代码十:

package com.itheima\_06;

/\*

\* 去除字符串两端空格

\* String trim()

\* 按照指定符号分割字符串

\* String[] split(String str)

\*/

public class StringDemo {

public static void main(String[] args) {

//创建字符串对象

String s1 = "helloworld";

String s2 = " helloworld ";

String s3 = " hello world ";

System.out.println("---"+s1+"---");

System.out.println("---"+s1.trim()+"---");

System.out.println("---"+s2+"---");

System.out.println("---"+s2.trim()+"---");

System.out.println("---"+s3+"---");

System.out.println("---"+s3.trim()+"---");

System.out.println("-------------------");

//String[] split(String str)

//创建字符串对象

String s4 = "aa,bb,cc";

String[] strArray = s4.split(",");

for(int x=0; x<strArray.length; x++) {

System.out.println(strArray[x]);

}

}

}

### String类的其它案例

#### 案例代码十一:

package com.itheima\_07;

/\*

\* 把数组中的数据按照指定个格式拼接成一个字符串

\* 举例：int[] arr = {1,2,3};

\* 输出结果：[1, 2, 3]

\*

\* 分析：

\* A:定义一个int类型的数组

\* B:写方法实现把数组中的元素按照指定的格式拼接成一个字符串

\* C:调用方法

\* D:输出结果

\*/

public class StringTest {

public static void main(String[] args) {

//定义一个int类型的数组

int[] arr = {1,2,3};

//写方法实现把数组中的元素按照指定的格式拼接成一个字符串

//调用方法

String s = arrayToString(arr);

//输出结果

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

}

/\*

\* 两个明确：

\* 返回值类型：String

\* 参数列表：int[] arr

\*/

public static String arrayToString(int[] arr) {

String s = "";

//[1, 2, 3]

s += "[";

for(int x=0; x<arr.length; x++) {

if(x==arr.length-1) {

s += arr[x];

}else {

s += arr[x];

s += ", ";

}

}

s += "]";

return s;

}

}

#### 案例代码十二:

package com.itheima\_07;

import java.util.Scanner;

/\*

\* 字符串反转

\* 举例：键盘录入”abc”

\* 输出结果：”cba”

\*

\* 分析：

\* A:键盘录入一个字符串

\* B:写方法实现字符串的反转

\* a:把字符串倒着遍历，得到的每一个字符拼接成字符串。

\* b:把字符串转换为字符数组，然后对字符数组进行反转，最后在把字符数组转换为字符串

\* C:调用方法

\* D:输出结果

\*/

public class StringTest2 {

public static void main(String[] args) {

//键盘录入一个字符串

Scanner sc = new Scanner(System.in);

System.out.println("请输入一个字符串：");

String s = sc.nextLine();

//写方法实现字符串的反转

//调用方法

String result = reverse(s);

//输出结果

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

}

/\*

\* 把字符串倒着遍历，得到的每一个字符拼接成字符串。

\*

\* 两个明确：

\* 返回值类型：String

\* 参数列表：String s

\*/

/\*

public static String reverse(String s) {

String ss = "";

for(int x=s.length()-1; x>=0; x--) {

ss += s.charAt(x);

}

return ss;

}

\*/

//把字符串转换为字符数组，然后对字符数组进行反转，最后在把字符数组转换为字符串

public static String reverse(String s) {

//把字符串转换为字符数组

char[] chs = s.toCharArray();

//对字符数组进行反转

for(int start=0,end=chs.length-1; start<=end; start++,end--) {

char temp = chs[start];

chs[start] = chs[end];

chs[end] = temp;

}

//最后在把字符数组转换为字符串

String ss = new String(chs);

return ss;

}

}

# StringBuilder类

## StringBuilder类概述

StringBuilder:是一个可变的字符串。字符串缓冲区类。

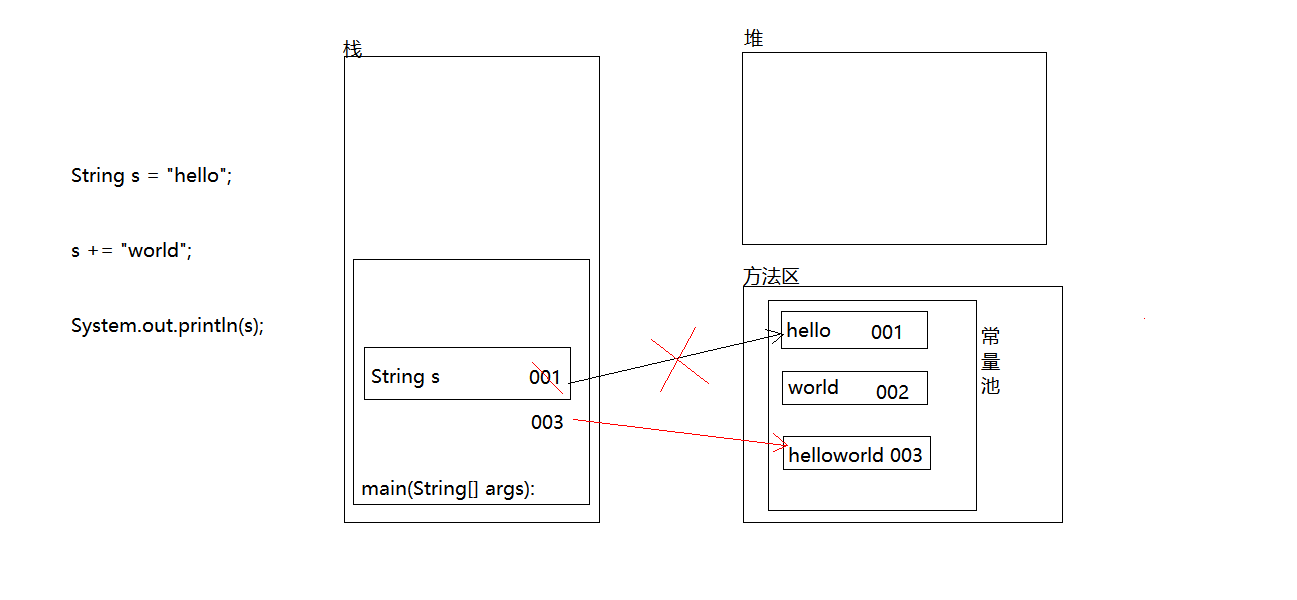
String和StringBuilder的区别：

String的内容是固定的

StringBuilder的内容是可变的

### +=拼接字符串耗费内存原因:

每次拼接都会产生新的字符串对象,而利用StringBuilder来拼接字符串自始至终用的都是同一个StringBuilder容器



## StringBuilder类的常用方法

A:构造方法:

StringBuilder()

B:成员方法:

public int capacity():返回当前容量 (理论值)

public int length():返回长度(已经存储的字符个数)

public StringBuilder append(任意类型):添加数据，并返回自身对象

public StringBuilder reverse():反转功能

### 案例代码十三:

package com.itheima\_01;

/\*

\* StringBuilder:是一个可变的字符串。字符串缓冲区类。

\*

\* String和StringBuilder的区别：

\* String的内容是固定的。

\* StringBuilder的内容是可变的。

\*

\* 构造方法：

\* StringBuilder()

\*

\* 成员方法：

\* public int capacity():返回当前容量

\* public int length():返回长度（字符数）

\*

\* 容量：理论值

\* 长度：实际值

\*/

public class StringBuilderDemo {

public static void main(String[] args) {

//创建对象

StringBuilder sb = new StringBuilder();

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

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

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

}

}

### 案例代码十四:

package com.itheima\_02;

/\*

\* 添加功能

\* public StringBuilder append(任意类型):添加数据，并返回自身对象

\* 反转功能

\* public StringBuilder reverse()

\*/

public class StringBuilderDemo {

public static void main(String[] args) {

//创建对象

StringBuilder sb = new StringBuilder();

//public StringBuilder append(任意类型)

//StringBuilder sb2 = sb.append("hello");

/\*

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

System.out.println("sb2:"+sb2);

System.out.println(sb == sb2); //true

\*/

/\*

sb.append("hello");

sb.append("world");

sb.append(true);

sb.append(100);

\*/

//链式编程

sb.append("hello").append("world").append(true).append(100);

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

//public StringBuilder reverse()

sb.reverse();

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

}

}

## StringBuilder案例

### 案例一需求:

StringBuilder和String通过方法完成相互转换

#### 案例代码十五:

package com.itheima\_03;

/\*

\* StringBuilder和String的相互转换

\*

\* StringBuilder -- String

\* public String toString():通过toString()就可以实现把StringBuilder转成String

\*

\* String -- StringBuilder

\* StringBuilder(String str):通过构造方法就可以实现把String转成StringBuilder

\*/

public class StringBuilderTest {

public static void main(String[] args) {

//StringBuilder -- String

/\*

StringBuilder sb = new StringBuilder();

sb.append("hello").append("world");

String s = sb.toString();

System.out.println(s);

\*/

//String -- StringBuilder

String s = "helloworld";

StringBuilder sb = new StringBuilder(s);

System.out.println(sb);

}

}

### 案例二需求:

利用StringBuilder把数组拼接成一个字符串

举例：

int[] arr = {1,2,3};

结果：

[1, 2, 3]

#### 案例代码十六:

package com.itheima\_03;

/\*

\* 把数组拼接成一个字符串

\* 举例：

\* int[] arr = {1,2,3};

\* 结果：

\* [1, 2, 3]

\*/

public class StringBuilderTest2 {

public static void main(String[] args) {

//定义一个数组

int[] arr = {1,2,3};

//写方法实现拼接

//调用方法

String s = arrayToString(arr);

//输出结果

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

}

/\*

\* 两个明确：

\* 返回值类型：String

\* 参数列表：int[] arr

\*/

public static String arrayToString(int[] arr) {

StringBuilder sb = new StringBuilder();

//[1, 2, 3]

sb.append("[");

for(int x=0; x<arr.length; x++) {

if(x==arr.length-1) {

sb.append(arr[x]);

}else {

sb.append(arr[x]).append(", ");

}

}

sb.append("]");

String result = sb.toString();

return result;

}

}

### 案例三需求:

利用StringBuilder完成字符串反转

#### 案例代码十七:

package com.itheima\_03;

import java.util.Scanner;

/\*

\* 把字符串反转

\*

\* 分析：

\* A:键盘录入一个字符串

\* B:写方法实现反转

\* String -- StringBuilder -- reverse() -- String

\* C:调用方法

\* D:输出结果

\*/

public class StringBuilderTest3 {

public static void main(String[] args) {

//键盘录入一个字符串

Scanner sc = new Scanner(System.in);

System.out.println("请输入一个字符串：");

String s = sc.nextLine();

//写方法实现反转

//调用方法

String result = myReverse(s);

//输出结果

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

}

/\*

\* 两个明确：

\* 返回值类型：String

\* 参数列表：String

\*/

public static String myReverse(String s) {

//String -- StringBuilder -- reverse() -- String

StringBuilder sb = new StringBuilder(s);

sb.reverse();

String result = sb.toString();

return result;

}

}

### 案例四需求:

判断一个字符串是否是对称字符串

例如"abc"不是对称字符串，"aba"、"abba"、"aaa"、"mnanm"是对称字符串

#### 案例代码十八:

package com.itheima\_03;

import java.util.Scanner;

/\*

\* 判断一个字符串是否是对称字符串

\* 例如"abc"不是对称字符串，"aba"、"abba"、"aaa"、"mnanm"是对称字符串

\*

\* 分析：

\* A:键盘录入一个字符串

\* B:写方法实现判断一个字符串是否是对称字符串

\* 把字符串反转，和反转前的字符串进行比较，如果内容相同，就说明是对称字符串

\* C:调用方法

\* D:输出结果

\*/

public class StringBuilderTest4 {

public static void main(String[] args) {

//键盘录入一个字符串

Scanner sc = new Scanner(System.in);

System.out.println("请输入一个字符串：");

String s = sc.nextLine();

//写方法实现判断一个字符串是否是对称字符串

//调用方法

boolean b = isSymmetry(s);

//输出结果

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

}

/\*

\* 两个明确：

\* 返回值类型：boolean

\* 参数列表：String s

\*/

public static boolean isSymmetry(String s) {

//把字符串反转，和反转前的字符串进行比较，如果内容相同，就说明是对称字符串

StringBuilder sb = new StringBuilder(s);

sb.reverse();

String result = sb.toString();

return result.equals(s);

}

}