第08天 常用API

今日内容介绍

* API概述
* Scanner类
* Object类与String类
* StringBuilder类

# API概述

## API概念

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

也称之为 : 帮助文档

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 Object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,接收的数据类型为基本数据类型和字符串类型.

### Scanner类接受键盘录入的字符串

#### 案例代码一:

**package** com.itheima;

**import** java.util.Scanner;

/\*

\* Scanner:用于获取键盘录入数据。(基本数据类型，字符串数据)

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

\*/

**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 line = sc.nextLine();

//输出结果

System.***out***.println("line:"+line);

}

}

## Object类

### Object类作用

Object:是类层次结构的根类，所有的类都直接的或者间接的继承自该类。

### Object类的toString()方法

#### 案例代码二:

**package** com.itheima\_01;

**public** **class** Student {

**private** String name;

**private** **int** age;

**public** Student() {}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Student [name=" + name + ", age=" + age + "]";

}

}

**package** com.itheima\_01;

/\*

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

\* 构造方法：Object()

\*

\* 直接输出对象名，输出底层调用的是该对象的toString()

\*

\* 查看API，我们发现建议所有子类都重写toString()。

\* 到底该如何重写该方法呢?自动生成的就可以。

\*/

**public** **class** ObjectDemo {

**public** **static** **void** main(String[] args) {

Student s = **new** Student();

s.setName("林青霞");

s.setAge(30);

System.***out***.println(s);

//System.out.println(s.toString());

/\*

public void println(Object x) { //Object x = s;

String s = String.valueOf(x);

synchronized (this) {

print(s);

newLine();

}

}

public static String valueOf(Object obj) { //Object obj = x;

return (obj == null) ? "null" : obj.toString();

}

\*/

}

}

### Object类的equals()方法

#### 案例代码三:

**package** com.itheima\_02;

**public** **class** Student {

**private** String name;

**private** **int** age;

**public** Student() {}

**public** Student(String name,**int** age) {

**this**.name = name;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Student [name=" + name + ", age=" + age + "]";

}

@Override

**public** **boolean** equals(Object obj) {

//s1.equals(s2)

//this -- s1

//obj -- s2

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

//比较的是对象是否是同一个类的对象

**if** (getClass() != obj.getClass())

**return** **false**;

Student other = (Student) obj; //other -- s2

**if** (age != other.age) //this.age -- s1.age

**return** **false**;

**if** (name == **null**) {

**if** (other.name != **null**)

**return** **false**;

} **else** **if** (!name.equals(other.name))

//字符串的equals()比较的是字符串的内容是否相同

**return** **false**;

**return** **true**;

}

}

**package** com.itheima\_02;

/\*

\* ==:

\* 可以比较基本数据类型，比较基本数据类型的时候比较的是基本数据类型的值是否相同

\* 也可以比较引用数据类型，比较引用数据类型的时候比较的是引用数据类型的地址值是否相同

\*

\* 而我们现在想比较的是对象的内容是否相同?该怎么办呢?

\* 通过查看API，我们找到了比较对象是否相等的方法：

\* public boolean equals(Object obj)

\* Object类中的equals()方法默认比较的是对象的地址是否相同。

\* 如果我们想比较对象的内容是否相同，就必须自己重写该方法。

\* 如何重写该方法呢?

\* 自动生成即可。

\*/

**public** **class** ObjectDemo {

**public** **static** **void** main(String[] args) {

Student s1 = **new** Student("林青霞", 30);

Student s2 = **new** Student("林青霞", 30);

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

System.***out***.println(s1.equals(s2));//false

/\*

public boolean equals(Object obj) {

//this -- s1

//obj -- s2

return (this == obj);

}

\*/

}

}

# 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

##### 案例代码四:

**package** com.itheima\_01;

/\*

\* String：代表字符串类。

\* 由多个字符组成的一串数据。

\* 字符串的本质就是一个字符数组。

\*

\* 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):把字符数组的一部分数据封装成字符串对象

\*

\* public String toString():返回此对象本身（它已经是一个字符串！）。

\*/

**public** **class** StringDemo {

**public** **static** **void** main(String[] args) {

//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);

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

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

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

String s2 = **new** String(value);

System.***out***.println(s2);

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

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

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

String s3 = **new** String(value,0,3);

System.***out***.println(s3);

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

//最常用的

String s4 = "hello";

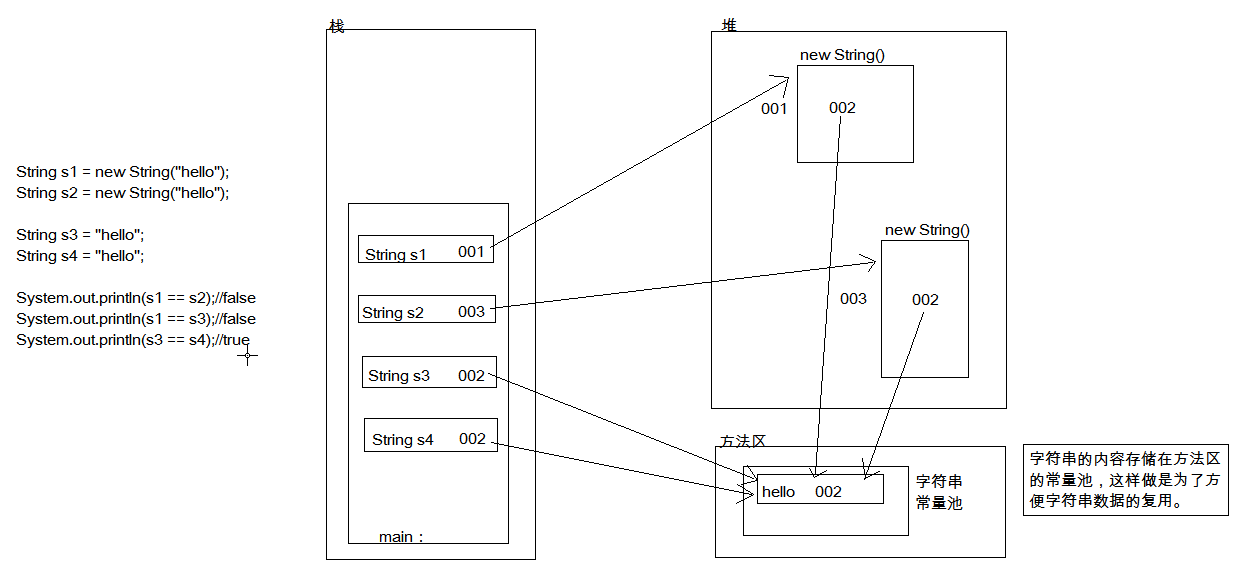
System.***out***.println(s4);

}

}

#### 创建字符串对象两种方式的区别

##### 案例代码五:



**package** com.itheima\_02;

/\*

\* String类创建对象的特点：

\* A:通过构造方法创建对象

\* B:通过直接赋值的方式创建对象

\* 这两种方式的创建是有区别的。

\* 通过构造方法创建的字符串对象是在堆内存。

\* 通过直接赋值的方式创建的字符串对象是在方法区的常量池。

\*/

**public** **class** StringDemo {

**public** **static** **void** main(String[] args) {

String s1 = **new** String("hello");

String s2 = **new** String("hello");

String s3 = "hello";

String s4 = "hello";

System.***out***.println(s1 == s2);//false

System.***out***.println(s1 == s3);//false

System.***out***.println(s3 == s4);//true

}

}

### String的练习之模拟用户登录

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

#### 模拟用户登录演示

需求：模拟登录,给三次机会,并提示还有几次

##### 案例代码六:

**package** com.itheima\_03;

**import** java.util.Scanner;

/\*

\* 需求：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去跟已经存在的用户名和密码进行比较

\* 如果内容相同，就提示登录成功

\* 如果内容不同，就提示登录失败，并提示还有几次机会

\* public boolean equals(Object anObject):比较字符串的内容，严格区分大小写(用户名和密码)

\* public boolean equalsIgnoreCase(String anotherString):比较字符串的内容，不考虑大小写(验证码)

\*/

**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";

**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去跟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("你的帐号被锁定，请与管理员联系");

}**else** {

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

}

}

}

}

}

### String类的获取功能

public char charAt(int index):返回指定索引处的值

public int length():返回字符串中的字符个数，字符串的长度

#### 遍历字符串的演示

需求:遍历字符串(获取字符串中的每一个字符)

##### 案例代码七:

**package** com.itheima\_03;

/\*

\* 需求：遍历字符串(获取字符串中的每一个字符)

\*/

**public** **class** StringTest2 {

**public** **static** **void** main(String[] args) {

//要遍历字符串，你首先得有一个字符串

String s = "abcde";

//思考：如何获取字符串中的每一个字符

//假如让我们来提供方法，我们应该提供一个根据索引返回指定位置的字符的方法

//返回值：char

//形式参数：int index

//public char charAt(int index):返回指定索引处的值

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

//目前for循环中的数据5是我们数出来的，那么字符串有没有提供一个方法，用于获取字符串中字符的个数呢?

//public int length():返回字符串中的字符个数，字符串的长度

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

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

}

}

}

### 对字符串的拼接案例

需求：把数组中的数据按照指定个格式拼接成一个字符串

#### 案例代码八:

**package** com.itheima\_03;

/\*

\* 需求：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** StringTest3 {

**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 result = *arrayToString*(arr);

//输出结果

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

}

/\*

\* 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;

}

}

### String类的反转案例

需求：字符串反转

#### 案例代码九:

**package** com.itheima\_03;

**import** java.util.Scanner;

/\*

\* 需求：字符串反转

\* 举例：键盘录入”abc”

\* 输出结果：”cba”

\*

\* 分析：

\* A:键盘录入字符串数据

\* B:写方法实现字符串数据的反转

\* 把字符串倒着遍历，在把每一个得到的字符拼接成一个字符串

\* C:调用方法

\* D:输出结果

\*/

**public** **class** StringTest4 {

**public** **static** **void** main(String[] args) {

//键盘录入字符串数据

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("请输入一个字符串：");

String s = sc.nextLine();

//写方法实现字符串数据的反转

//调用方法

String result = *reverse*(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 s

\*/

**public** **static** String reverse(String s) {

//把字符串倒着遍历，在把每一个得到的字符拼接成一个字符串

String ss = "";

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

ss += s.charAt(x);

}

**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。

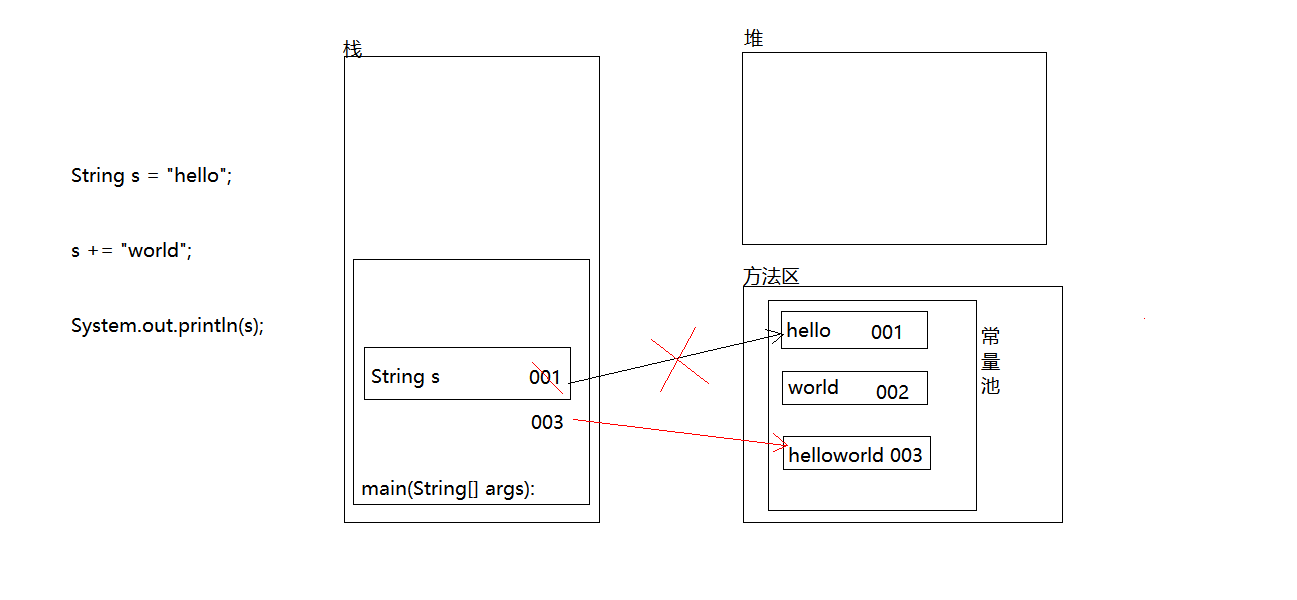
String和StringBuilder的区别：

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:

public StringBuilder()

public StringBuilder(String str)

B:成员方法:

public String toString():返回此序列中数据的字符串表示形式。

public StringBuilder append(任意类型):添加数据，并返回对象本身。

public StringBuilder reverse():字符串本身进行反转

### 案例代码十:

**package** com.itheima\_01;

/\*

\* StringBuilder:是一个可变的字符串类。

\*

\* 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

\* public StringBuilder()

\* public StringBuilder(String str)

\*

\* public String toString():返回此序列中数据的字符串表示形式。

\*/

**public** **class** StringBuilderDemo {

**public** **static** **void** main(String[] args) {

//public StringBuilder()

StringBuilder sb = **new** StringBuilder();

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

System.***out***.println(sb.length());

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

//public StringBuilder(String str)

StringBuilder sb2 = **new** StringBuilder("helloworld");

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

System.***out***.println(sb2.length());

}

}

### 案例代码十一:

**package** com.itheima\_02;

/\*

\* 添加功能

\* public StringBuilder append(任意类型):添加数据，并返回对象本身。

\* 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

#### 案例代码十二:

**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转换为String

\*

\* String -- StringBuilder

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

\*/

**public** **class** StringBuilderTest {

**public** **static** **void** main(String[] args) {

/\*

//StringBuilder -- String

StringBuilder sb = new StringBuilder();

sb.append("hello");

//错误的

//String s = sb;

//public String toString()

String s = sb.toString();

System.out.println(s);

\*/

//String -- StringBuilder

String s = "hello";

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]

#### 案例代码十三:

**package** com.itheima\_03;

/\*

\* 把数组拼接成一个字符串

\* 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** StringBuilderTest2 {

**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 result = *arrayToString*(arr);

//输出结果

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

}

/\*

\* 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 s = sb.toString();

**return** s;

}

}

### 案例三需求:

利用StringBuilder完成字符串反转

举例：键盘录入”abc”

输出结果：”cba”

#### 案例代码十四:

**package** com.itheima\_03;

**import** java.util.Scanner;

/\*

\* 把字符串反转

\* 举例：键盘录入”abc”

\* 输出结果：”cba”

\*

\* 分析：

\* A:键盘录入一个字符串数据

\* B:写方法实现字符串数据的反转

\* 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数据

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("请输入一个字符串：");

String s = sc.nextLine();

//写方法实现字符串数据的反转

//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 s

\*/

**public** **static** String myReverse(String s) {

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

/\*

StringBuilder sb = new StringBuilder(s);

sb.reverse();

String ss = sb.toString();

return ss;

\*/

**return** **new** StringBuilder(s).reverse().toString();

}

}