1.API

1.1 API概述-帮助文档的使用

- 什么是API

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

- java中的API

  指的就是 JDK 中提供的各种功能的 Java类，这些类将底层的实现封装了起来，我们不需要关心这些类是如何实现的，只需要学习这些类如何使用即可，我们可以通过帮助文档来学习这些API如何使用。

如何使用API帮助文档：

- 打开帮助文档
- 找到索引选项卡中的输入框
- 在输入框中输入Random
- 看类在哪个包下
- 看类的描述
- 看构造方法
- 看成员方法

1.2 键盘录入字符串

Scanner类：

next() : 遇到了空格, 就不再录入数据了 , 结束标记: 空格, tab键

nextLine() : 可以将数据完整的接收过来 , 结束标记: 回车换行符

代码实现：

```
package com.itheima.api;

import java.util.Scanner;

public class Demo1Scanner {
    /*
        next() : 遇到了空格，就不再录入数据了

                结束标记：空格，tab键

        nextLine() : 可以将数据完整的接收过来

                结束标记：回车换行符
     */
    public static void main(String[] args) {
        // 1. 创建Scanner对象
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入:");
```

```
        // 2. 调用nextLine方法接收字符串
        // ctrl + alt + v : 快速生成方法的返回值
        String s = sc.nextLine();

        System.out.println(s);
    }
}

```

```
package com.itheima.api;

import java.util.Scanner;

public class Demo2Scanner {
    /*
        nextInt和nextLine方法配合使用的时候，nextLine方法就没有键盘录入的机会了

        建议：今后键盘录入数据的时候，如果是字符串和整数一起接受，建议使用next方法接受字符串.
     */
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入整数:");
        int num = sc.nextInt(); // 10 + 回车换行
        System.out.println("请输入字符串:");
        String s = sc.nextLine();


        System.out.println(num);
        System.out.println(s);
    }
}
```

2. String类

2.1 String概述

1 String 类在 java.lang 包下，所以使用的时候不需要导包

2 String 类代表字符串，Java 程序中的所有字符串文字（例如“abc”）都被实现为此类的实例也就是说，Java 程序中所有的双引号字符串，都是 String 类的对象

3 字符串不可变，它们的值在创建后不能被更改

2.2 String类的构造方法

常用的构造方法

方法名	说明
public String()	创建一个空白字符串对象，不含有任何内容
public String(char[] chs)	根据字符数组的内容，来创建字符串对象
public String(String original)	根据传入的字符串内容，来创建字符串对象
String s = “abc”；	直接赋值的方式创建字符串对象，内容就是abc

示例代码

```
package com.itheima.string;

public class Demo2StringConstructor {
    /*
        String类常见构造方法:

            public String() : 创建一个空白字符串对象，不含有任何内容
            public String(char[] chs) : 根据字符数组的内容，来创建字符串对象
            public String(String original) : 根据传入的字符串内容，来创建字符串对象
            String s = “abc”;  直接赋值的方式创建字符串对象，内容就是abc

         注意:
                String这个类比较特殊，打印其对象名的时候，不会出现内存地址
                而是该对象所记录的真实内容.

                面向对象-继承，Object类
     */
    public static void main(String[] args) {
        // public String() : 创建一个空白字符串对象，不含有任何内容
        String s1 = new String();
        System.out.println(s1);

        // public String(char[] chs) : 根据字符数组的内容，来创建字符串对象
        char[] chs = {'a','b','c'};
        String s2 = new String(chs);
        System.out.println(s2);

        // public String(String original) : 根据传入的字符串内容，来创建字符串对象
        String s3 = new String("123");
        System.out.println(s3);
    }
}
```

2.4 创建字符串对象的区别对比

- **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

2.5 字符串的比较

2.5.1 字符串的比较

- == 比较基本数据类型：比较的是具体的值
- == 比较引用数据类型：比较的是对象地址值

String类 : public boolean equals(String s) 比较两个字符串内容是否相同、区分大小写

代码 :

```
package com.itheima.stringmethod;

public class Demo1Equals {
    public static void main(String[] args) {
        String s1 = "abc";
        String s2 = "ABC";
        String s3 = "abc";

        // equals : 比较字符串内容，区分大小写
        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));

        // equalsIgnoreCase : 比较字符串内容，忽略大小写
        System.out.println(s1.equalsIgnoreCase(s2));
    }
}

```

2.6 用户登录案例【应用】

案例需求 :

已知用户名和密码，请用程序实现模拟用户登录。总共给三次机会，登录之后，给出相应的提示

实现步骤 :

1. 已知用户名和密码，定义两个字符串表示即可
2. 键盘录入要登录的用户名和密码，用 Scanner 实现
3. 拿键盘录入的用户名、密码和已知的用户名、密码进行比较，给出相应的提示。
4. 字符串的内容比较，用equals() 方法实现
5. 用循环实现多次机会，这里的次数明确，采用for循环实现，并在登录成功的时候，使用break结束循

代码实现 :

```
package com.itheima.test;


import java.util.Scanner;
```

```java

public class Test1 {
    /*
        需求：已知用户名和密码，请用程序实现模拟用户登录。
              总共给三次机会，登录之后，给出相应的提示

        思路：
        1. 已知用户名和密码，定义两个字符串表示即可
        2. 键盘录入要登录的用户名和密码，用 Scanner 实现
        3. 拿键盘录入的用户名、密码和已知的用户名、密码进行比较，给出相应的提示。
           字符串的内容比较，用equals() 方法实现
        4. 用循环实现多次机会，这里的次数明确，采用for循环实现，并在登录成功的时候，使用break结束循环

     */
    public static void main(String[] args) {
        // 1. 已知用户名和密码，定义两个字符串表示即可
        String username = "admin";
        String password = "123456";
        // 2. 键盘录入要登录的用户名和密码，用 Scanner 实现
        Scanner sc = new Scanner(System.in);
        // 4. 用循环实现多次机会，这里的次数明确，采用for循环实现
        for(int i = 1; i <= 3; i++){
            System.out.println("请输入用户名:");
            String scUsername = sc.nextLine();
            System.out.println("请输入密码:");
            String scPassword = sc.nextLine();
            // 3. 拿键盘录入的用户名、密码和已知的用户名、密码进行比较，给出相应的提示。
            if(username.equals(scUsername) && password.equals(scPassword)){
                System.out.println("登录成功");
                break;
            }else{
                if(i == 3){
                    System.out.println("您的登录次数已达到今日上限，请明天再来");
                }else{
                    System.out.println("登录失败,您还剩余" + (3-i) +"次机会");
                }

            }
        }

    }
}
```

2.7 遍历字符串案例【应用】

案例需求：

键盘录入一个字符串，使用程序实现在控制台遍历该字符串

实现步骤：

1. 键盘录入一个字符串，用 Scanner 实现

2. 遍历字符串，首先要能够获取到字符串中的每一个字符, public char charAt(int index)：返回指定索引处的char值，字符串的索引也是从0开始的
3. 遍历字符串，其次要能够获取到字符串的长度， public int length()：返回此字符串的长度
4. 遍历打印

代码实现：

```
package com.itheima.test;

import java.util.Scanner;

public class Test2 {
    /*
        需求：键盘录入一个字符串，使用程序实现在控制台遍历该字符串

        思路：
        1. 键盘录入一个字符串，用 Scanner 实现
        2. 遍历字符串，首先要能够获取到字符串中的每一个字符
            public char charAt(int index)：返回指定索引处的char值，字符串的索引也是从0开始的
        3. 遍历字符串，其次要能够获取到字符串的长度
            public int length()：返回此字符串的长度
        4. 遍历打印
9
     */
    public static void main(String[] args) {
        //  1. 键盘录入一个字符串，用 Scanner 实现
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入:");
        String s = sc.nextLine();
        // 2. 遍历字符串，首先要能够获取到字符串中的每一个字符
        for(int i = 0; i < s.length(); i++){
            // i : 字符串的每一个索引
            char c = s.charAt(i);
            System.out.println(c);
        }
    }
}
```

2.8 统计字符次数案例【应用】

案例需求：

键盘录入一个字符串，使用程序实现在控制台遍历该字符串

实现步骤：

1. 键盘录入一个字符串，用 Scanner 实现
2. 将字符串拆分为字符数组 , public char[] toCharArray( )：将当前字符串拆分为字符数组并返回
3. 遍历字符数

代码实现：

```
package com.itheima.test;
```

```java

import java.util.Scanner;

public class Test3 {
    /*
       需求：键盘录入一个字符串，使用程序实现在控制台遍历该字符串

       思路：
       1. 键盘录入一个字符串，用 Scanner 实现
       2. 将字符串拆分为字符数组
                public char[] toCharArray( )：将当前字符串拆分为字符数组并返回
       3. 遍历字符数组

    */
    public static void main(String[] args) {
        //  1. 键盘录入一个字符串，用 Scanner 实现
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入:");
        String s = sc.nextLine();
        // 2. 将字符串拆分为字符数组
        char[] chars = s.toCharArray();
        // 3. 遍历字符数组
        for (int i = 0; i < chars.length; i++) {
            System.out.println(chars[i]);
        }
    }
}
```

2.9 手机号屏蔽-字符串截取

案例需求：

以字符串的形式从键盘接受一个手机号，将中间四位号码屏蔽 最终效果为：1561234

实现步骤：

1. 键盘录入一个字符串，用 Scanner 实现
2. 截取字符串前三位
3. 截取字符串后四位
4. 将截取后的两个字符串，中间加上进行拼接，输出结果

代码实现：

```java
package com.itheima.test;

import java.util.Scanner;

public class Test5 {
    /*
        需求：以字符串的形式从键盘接受一个手机号，将中间四位号码屏蔽
        最终效果为：156****1234


        思路：
```

```
        1. 键盘录入一个字符串，用 Scanner 实现
        2. 截取字符串前三位
        3. 截取字符串后四位
        4. 将截取后的两个字符串，中间加上****进行拼接，输出结果

     */
    public static void main(String[] args) {
        // 1. 键盘录入一个字符串，用 Scanner 实现
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入手机号:");
        String telString = sc.nextLine();
        // 2. 截取字符串前三位
        String start = telString.substring(0,3);
        // 3. 截取字符串后四位
        String end = telString.substring(7);
        // 4. 将截取后的两个字符串，中间加上****进行拼接，输出结果
        System.out.println(start + "****" + end);
    }
}
```

2.10 敏感词替换-字符串替换

案例需求：

键盘录入一个 字符串，如果字符串中包含（TMD），则使用***替换

实现步骤：

1. 键盘录入一个字符串，用 Scanner 实现
2. 替换敏感词
   String replace(CharSequence target, CharSequence replacement)
   将当前字符串中的target内容，使用replacement进行替换，返回新的字符串
3. 输出结果

代码实现：

```
package com.itheima.test;

import java.util.Scanner;

public class Test6 {
    /*
        需求：键盘录入一个 字符串，如果字符串中包含（TMD），则使用***替换

        思路：
        1. 键盘录入一个字符串，用 Scanner 实现
        2. 替换敏感词
                String replace(CharSequence target, CharSequence replacement)
                将当前字符串中的target内容，使用replacement进行替换，返回新的字符串
        3. 输出结果

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

```
        // 1. 键盘录入一个字符串，用 Scanner 实现
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入:");
        String s = sc.nextLine();
        // 2. 替换敏感词
        String result = s.replace("TMD","***");
        // 3. 输出结果
        System.out.println(result);
    }
}
```

2.11 切割字符串

案例需求：

以字符串的形式从键盘录入学生信息，例如：“张三 , 23”

从该字符串中切割出有效数据,封装为Student学生对象

实现步骤：

1. 编写Student类，用于封装数据
2. 键盘录入一个字符串，用 Scanner 实现
3. 根据逗号切割字符串，得到（张三） （23）

   String[] split(String regex)：根据传入的字符串作为规则进行切割 将切割后的内容存入字符串数组中，并将字符串数组返回
4. 从得到的字符串数组中取出元素内容，通过Student类的有参构造方法封装为对象
5. 调用对象getXxx方法，取出数据并打印。

代码实现：

```
package com.itheima.test;

import com.itheima.domain.Student;

import java.util.Scanner;

public class Test7 {
    /*
          需求：以字符串的形式从键盘录入学生信息，例如：“张三 ， 23”
                从该字符串中切割出有效数据,封装为Student学生对象
          思路：
             1. 编写Student类，用于封装数据
             2. 键盘录入一个字符串，用 Scanner 实现
             3. 根据逗号切割字符串，得到（张三） （23）
                    String[] split(String regex) ：根据传入的字符串作为规则进行切割
                    将切割后的内容存入字符串数组中，并将字符串数组返回
             4. 从得到的字符串数组中取出元素内容，通过Student类的有参构造方法封装为对象
             5. 调用对象getXxx方法，取出数据并打印。


     */
```

```
21      public static void main(String[] args) {
22          // 2. 键盘录入一个字符串，用 Scanner 实现
23          Scanner sc = new Scanner(System.in);
24          System.out.println("请输入学生信息:");
25          String stuInfo = sc.nextLine();
26          // stuInfo = "张三,23";
27          // 3. 根据逗号切割字符串，得到（张三） （23）
28          String[] sArr = stuInfo.split(",");
29
30  //          System.out.println(sArr[0]);
31  //          System.out.println(sArr[1]);
32
33          // 4. 从得到的字符串数组中取出元素内容，通过Student类的有参构造方法封装为对象
34          Student stu = new Student(sArr[0],sArr[1]);
35
36          // 5. 调用对象getXxx方法，取出数据并打印。
37          System.out.println(stu.getName() + "..." + stu.getAge());
38      }
39  }
```

2.12 String方法小结

String类的常用方法：

public boolean equals(Object anObject) 比较字符串的内容，严格区分大小写

public boolean equalsIgnoreCase(String anotherString) 比较字符串的内容，忽略大小写

public int length() 返回此字符串的长度

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

public char[] toCharArray() 将字符串拆分为字符数组后返回

public String substring(int beginIndex, int endIndex) 根据开始和结束索引进行截取，得到新的字符串（包含头，不包含尾）

public String substring(int beginIndex) 从传入的索引处截取，截取到末尾，得到新的字符串

public String replace(CharSequence target, CharSequence replacement) 使用新值，将字符串中的旧值替换，得到新的字符串

public String[] split(String regex) 根据传入的规则切割字符串，得到字符串数组

3 StringBuilder类

3.1 StringBuilder类概述

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

3.2 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

3.3StringBuilder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());
    }
}
```

3.4 StringBuilder常用的成员方法

- **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); //true 同个对象

//        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);
    }
}
```

3.5StringBuilder和String相互转换【应用】

- **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);
```

```
    }
}
```

3.6 StringBuilder拼接字符串案例

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]

实现步骤：

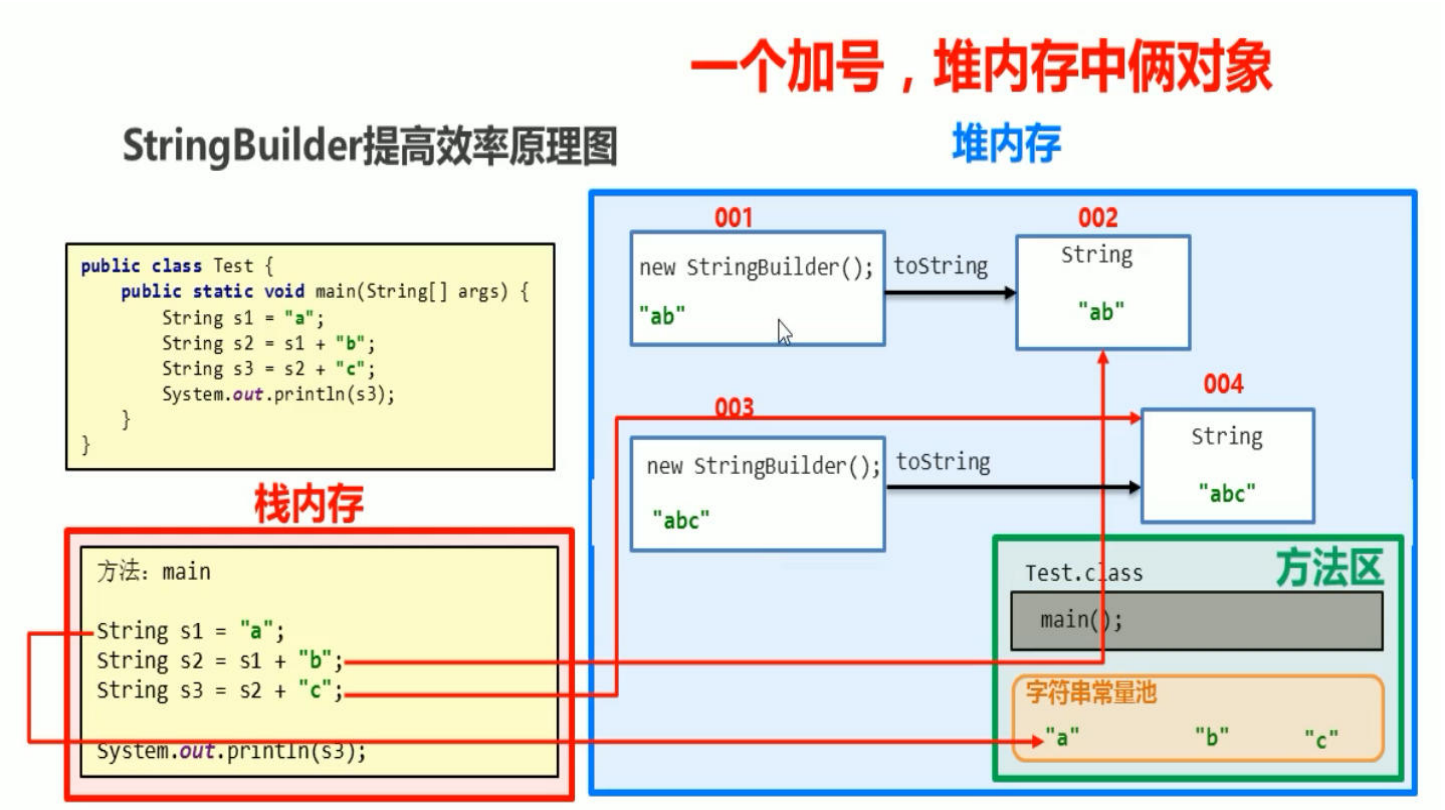
1. 定义一个 int 类型的数组，用静态初始化完成数组元素的初始化
2. 定义一个方法，用于把 int 数组中的数据按照指定格式拼接成一个字符串返回。
   返回值类型 String，参数列表 int[] arr
3. 在方法中用 StringBuilder 按照要求进行拼接，并把结果转成 String 返回
4. 调用方法，用一个变量接收结果
5. 输出结果

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();
```

```
32
33          sb.append("[");
34
35          for(int i=0; i<arr.length; i++) {
36              if(i == arr.length-1) {
37                  sb.append(arr[i]);
38              } else {
39                  sb.append(arr[i]).append(", ");
40              }
41          }
42
43          sb.append("]");
44
45          String s = sb.toString();
46
47          return  s;
48      }
49  }
```

StringBuilder 提高效率原理



常量池是在堆内存里

进来之后开始执行第1句代码，
1，这块使用双引号创建字符串对象，会先检查常量池里面有没有，没有的话就创建。
s1会记录常量池当中这个a的内存地址，好再往下，
变量s2= s1 + “ b ”，
这又来了一个双引号，他检查常量池里面有没有，没有就创建，
在这儿出现了字符串的加号拼接，那么在Java当中只要出现了字符串的加号拼接。
系统底层就会自动的把我们在堆内存当中创建出来一个StringBuilder的对象，
然后也会自动调用该对象的append方法完成拼接，
所以就完成了s1的a和b进行拼接变成了ab，
拼完了之后还不能让s2直接记录001这份内存地址，
因为s2现在是StringBuilder，我001现在呢是String，
这一块类型不匹配，没法直接复制，
所以系统呢还会自动调用标准的to string方法，
把其转换成一个String类型。
转换完毕之后，f2才能去记录002这款内存地址，
是这样的一个过程，好再往下，
s3=s2+"c"，
这又是双引号创建的，常量池里面有吗？
没有没有的创建，然后这块呢又出现了字符串加号的拼接，
系统底层再次自动。帮我们创建使用StringBuilder对象，
创建完之后自动调用append的方法完成拼接，
拼接完毕之后还要手动调用toString方法，转换成String的字符串类型，
然后s3在这一步才能记录004这一份内存地址。
我们可以做出来一个结论，
10万次的字符串拼接，内存当中。就只有一个StringBuilder对象，这就是StringBuilder提高效率的一个原理