

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：

```
1  package com.itheima.api;
2
3  import java.util.Scanner;
4
5  public class Demo1Scanner {
6      /*
7          next() : 遇到了空格, 就不再录入数据了
8
9          结束标记: 空格, tab键
10
11         nextLine() : 可以将数据完整的接收过来
12
13         结束标记: 回车换行符
14     */
15     public static void main(String[] args) {
16         // 1. 创建Scanner对象
17         Scanner sc = new Scanner(System.in);
18         System.out.println("请输入:");
```

```

19      // 2. 调用nextLine方法接收字符串
20      // ctrl + alt + v : 快速生成方法的返回值
21      String s = sc.nextLine();
22
23      System.out.println(s);
24  }
25 }
26

```

```

1  package com.itheima.api;
2
3  import java.util.Scanner;
4
5  public class Demo2Scanner {
6      /*
7          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
8
9          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。
10     */
11     public static void main(String[] args) {
12         Scanner sc = new Scanner(System.in);
13         System.out.println("请输入整数:");
14         int num = sc.nextInt(); // 10 + 回车换行
15         System.out.println("请输入字符串:");
16         String s = sc.nextLine();
17
18
19         System.out.println(num);
20         System.out.println(s);
21     }
22 }

```

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

```
1 package com.itheima.string;
2
3 public class Demo2StringConstructor {
4     /*
5         String类常见构造方法:
6
7         public String() : 创建一个空白字符串对象，不含有任何内容
8         public String(char[] chs) : 根据字符数组的内容，来创建字符串对象
9         public String(String original) : 根据传入的字符串内容，来创建字符串对象
10        String s = "abc"; 直接赋值的方式创建字符串对象，内容就是abc
11
12        注意:
13            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
14            而是该对象所记录的真实内容。
15
16            面向对象-继承，Object类
17        */
18    public static void main(String[] args) {
19        // public String() : 创建一个空白字符串对象，不含有任何内容
20        String s1 = new String();
21        System.out.println(s1);
22
23        // public String(char[] chs) : 根据字符数组的内容，来创建字符串对象
24        char[] chs = {'a','b','c'};
25        String s2 = new String(chs);
26        System.out.println(s2);
27
28        // public String(String original) : 根据传入的字符串内容，来创建字符串对象
29        String s3 = new String("123");
30        System.out.println(s3);
31    }
32 }
```

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：

```
1 package com.itheima.stringmethod;
2
3 public class Demo1Equals {
4     public static void main(String[] args) {
5         String s1 = "abc";
6         String s2 = "ABC";
7         String s3 = "abc";
8
9         // equals : 比较字符串内容，区分大小写
10        System.out.println(s1.equals(s2));
11        System.out.println(s1.equals(s3));
12
13        // equalsIgnoreCase : 比较字符串内容，忽略大小写
14        System.out.println(s1.equalsIgnoreCase(s2));
15    }
16 }
17
```

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结束循环

代码实现：

```
1 package com.itheima.test;
2
3 import java.util.Scanner;
```

```

4
5 public class Test1 {
6     /*
7         需求：已知用户名和密码，请用程序实现模拟用户登录。
8             总共给三次机会，登录之后，给出相应的提示
9
10        思路：
11        1. 已知用户名和密码，定义两个字符串表示即可
12        2. 键盘录入要登录的用户名和密码，用 Scanner 实现
13        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。
14            字符串的内容比较，用equals() 方法实现
15        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
16
17        */
18        public static void main(String[] args) {
19            // 1. 已知用户名和密码，定义两个字符串表示即可
20            String username = "admin";
21            String password = "123456";
22            // 2. 键盘录入要登录的用户名和密码，用 Scanner 实现
23            Scanner sc = new Scanner(System.in);
24            // 4. 用循环实现多次机会，这里的次数明确，采用for循环实现
25            for(int i = 1; i <= 3; i++){
26                System.out.println("请输入用户名:");
27                String scUsername = sc.nextLine();
28                System.out.println("请输入密码:");
29                String scPassword = sc.nextLine();
30                // 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。
31                if(username.equals(scUsername) && password.equals(scPassword)){
32                    System.out.println("登录成功");
33                    break;
34                }else{
35                    if(i == 3){
36                        System.out.println("您的登录次数已达到今日上限，请明天再来");
37                    }else{
38                        System.out.println("登录失败,您还剩余" + (3-i) + "次机会");
39                    }
40                }
41            }
42        }
43    }
44 }
45 }

```

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：

```
1 package com.itheima.test;
2
3 import java.util.Scanner;
4
5 public class Test2 {
6     /*
7         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
8
9         思路：
10        1. 键盘录入一个字符串，用 Scanner 实现
11        2. 遍历字符串，首先要能够获取到字符串中的每一个字符
12           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
13        3. 遍历字符串，其次要能够获取到字符串的长度
14           public int length(): 返回此字符串的长度
15        4. 遍历打印
16    */
17
18    public static void main(String[] args) {
19        // 1. 键盘录入一个字符串，用 Scanner 实现
20        Scanner sc = new Scanner(System.in);
21        System.out.println("请输入:");
22        String s = sc.nextLine();
23        // 2. 遍历字符串，首先要能够获取到字符串中的每一个字符
24        for(int i = 0; i < s.length(); i++){
25            // i : 字符串的每一个索引
26            char c = s.charAt(i);
27            System.out.println(c);
28        }
29    }
30 }
```

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：

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

```

2
3 import java.util.Scanner;
4
5 public class Test3 {
6     /*
7         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
8
9         思路：
10        1. 键盘录入一个字符串，用 Scanner 实现
11        2. 将字符串拆分为字符数组
12           public char[] toCharArray( ): 将当前字符串拆分为字符数组并返回
13        3. 遍历字符数组
14
15        */
16        public static void main(String[] args) {
17            // 1. 键盘录入一个字符串，用 Scanner 实现
18            Scanner sc = new Scanner(System.in);
19            System.out.println("请输入:");
20            String s = sc.nextLine();
21            // 2. 将字符串拆分为字符数组
22            char[] chars = s.toCharArray();
23            // 3. 遍历字符数组
24            for (int i = 0; i < chars.length; i++) {
25                System.out.println(chars[i]);
26            }
27        }
28    }

```

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：

```

1 package com.itheima.test;
2
3 import java.util.Scanner;
4
5 public class Test5 {
6     /*
7         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
8         最终效果为：156****1234
9
10        思路：

```

```

11      1. 键盘录入一个字符串，用 Scanner 实现
12      2. 截取字符串前三位
13      3. 截取字符串后四位
14      4. 将截取后的两个字符串，中间加上****进行拼接，输出结果
15
16      */
17      public static void main(String[] args) {
18          // 1. 键盘录入一个字符串，用 Scanner 实现
19          Scanner sc = new Scanner(System.in);
20          System.out.println("请输入手机号:");
21          String telString = sc.nextLine();
22          // 2. 截取字符串前三位
23          String start = telString.substring(0,3);
24          // 3. 截取字符串后四位
25          String end = telString.substring(7);
26          // 4. 将截取后的两个字符串，中间加上****进行拼接，输出结果
27          System.out.println(start + "****" + end);
28      }
29  }

```

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：

```

1  package com.itheima.test;
2
3  import java.util.Scanner;
4
5  public class Test6 {
6      /*
7          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
8
9          思路：
10         1. 键盘录入一个字符串，用 Scanner 实现
11         2. 替换敏感词
12             String replace(CharSequence target, CharSequence replacement)
13             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
14         3. 输出结果
15
16         */
17         public static void main(String[] args) {

```



```

18      // 1. 键盘录入一个字符串, 用 Scanner 实现
19      Scanner sc = new Scanner(System.in);
20      System.out.println("请输入:");
21      String s = sc.nextLine();
22      // 2. 替换敏感词
23      String result = s.replace("TMD", "****");
24      // 3. 输出结果
25      System.out.println(result);
26  }
27 }

```

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：

```

1  package com.itheima.test;
2
3  import com.itheima.domain.Student;
4
5  import java.util.Scanner;
6
7  public class Test7 {
8      /*
9          需求：以字符串的形式从键盘录入学生信息，例如：“张三 ， 23”
10             从该字符串中切割出有效数据,封装为Student学生对象
11         思路：
12             1. 编写Student类，用于封装数据
13             2. 键盘录入一个字符串，用 Scanner 实现
14             3. 根据逗号切割字符串，得到（张三）（23）
15                 String[] split(String regex)：根据传入的字符串作为规则进行切割
16                 将切割后的内容存入字符串数组中，并将字符串数组返回
17             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
18             5. 调用对象getXxx方法，取出数据并打印。
19
20             */

```

```

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

```
1 public class StringBuilderDemo01 {
2     public static void main(String[] args) {
3         //public StringBuilder(): 创建一个空白可变字符串对象，不含有任何内容
4         StringBuilder sb = new StringBuilder();
5         System.out.println("sb:" + sb);
6         System.out.println("sb.length():" + sb.length());
7
8         //public StringBuilder(String str): 根据字符串的内容，来创建可变字符串对象
9         StringBuilder sb2 = new StringBuilder("hello");
10        System.out.println("sb2:" + sb2);
11        System.out.println("sb2.length():" + sb2.length());
12    }
13 }
```

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**

```
1 public class StringBuilderDemo01 {
2     public static void main(String[] args) {
3         //创建对象
4         StringBuilder sb = new StringBuilder();
5
6         //public StringBuilder append(任意类型): 添加数据，并返回对象本身
7         //     StringBuilder sb2 = sb.append("hello");
8         //
9         //     System.out.println("sb:" + sb);
```

```

10 //      System.out.println("sb2:" + sb2);
11 //      System.out.println(sb == sb2);
12
13 //      sb.append("hello");
14 //      sb.append("world");
15 //      sb.append("java");
16 //      sb.append(100);
17
18 //链式编程
19 sb.append("hello").append("world").append("java").append(100);
20
21 System.out.println("sb:" + sb);
22
23 //public StringBuilder reverse(): 返回相反的字符序列
24 sb.reverse();
25 System.out.println("sb:" + sb);
26 }
27 }

```

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**

```

1 public class StringBuilderDemo02 {
2     public static void main(String[] args) {
3         /*
4         //StringBuilder 转换为 String
5         StringBuilder sb = new StringBuilder();
6         sb.append("hello");
7
8         //String s = sb; //这个是错误的做法
9
10        //public String toString(): 通过 toString() 就可以实现把 StringBuilder 转换为 String
11        String s = sb.toString();
12        System.out.println(s);
13        */
14
15        //String 转换为 StringBuilder
16        String s = "hello";
17
18        //StringBuilder sb = s; //这个是错误的做法
19
20        //public StringBuilder(String s): 通过构造方法就可以实现把 String 转换为 StringBuilder
21        StringBuilder sb = new StringBuilder(s);
22
23        System.out.println(sb);

```

```
24     }
25 }
```

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：

```
1  /*
2      思路：
3          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
4          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。
5              返回值类型 String，参数列表 int[] arr
6          3:在方法中用 StringBuilder 按照要求进行拼接，并把结果转成 String 返回
7          4:调用方法，用一个变量接收结果
8          5:输出结果
9  */
10 public class StringBuilderTest01 {
11     public static void main(String[] args) {
12         //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
13         int[] arr = {1, 2, 3};
14
15         //调用方法，用一个变量接收结果
16         String s = arrayToString(arr);
17
18         //输出结果
19         System.out.println("s:" + s);
20
21     }
22
23     //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
24     /*
25         两个明确：
26             返回值类型: String
27             参数: int[] arr
28     */
29     public static String arrayToString(int[] arr) {
30         //在方法中用 StringBuilder 按照要求进行拼接，并把结果转成 String 返回
31         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 }
```