# 字符串

## 字符串也是对象

虽然字符串的基本使用很简单，但它属于引用数据类型。无论通过何种方式创建了字符串，其实都是一个**java.lang.String**类的对象。那为什么不需要import导包语句，就可以直接使用字符串类型String呢？

所有的类在使用的时候都需要导包，除了以下两种情况：

1. 需要使用的目标类和当前类位于同一个包下；
2. 需要使用的目标类直接位于java.lang包下（不含其子包）。

由于字符串类型的数据特别常用，所以代码中直接定义一个双引号的数据，就自动创建了字符串对象，通常不需要手动进行new操作。请注意，**直接写上双引号就是字符串对象**。

## API文档

API（Application Programming Interface，应用程序编程接口）其实就是一本字典，帮我们列出了JDK中所有的类、所有的方法，是一份使用说明书。

使用步骤为：

1. 打开CHM文档，点击左上角“显示”按钮。
2. 在“索引”当中输入需要查找的类名称。
3. 根据需要阅读右侧的说明信息：包、构造方法、普通方法。

## 构造方法

字符串String的构造方法很多，仅记住常用的2+1种创建方式即可，其余不常用的方式可以查询API文档。创建字符串的常用方式有：

1. 直接使用双引号。
2. 构造方法：public String(char[] array)，使用字符数组来创建字符串。
3. 构造方法：public String(char[] array, int offset, int count)，使用字符数组的一部分创建字符串。

**public** **class** Demo03StringInit {

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

**char**[] array = { 'H', 'e', 'l', 'l', 'o' };

// 根据字符数组来创建字符串

// 类名称 对象名 = new 类名称(构造参数);

String str1 = **new** String(array);

System.***out***.println(str1); // Hello

// 根据字符数组的一部分来创建字符串

String str2 = **new** String(array, 2, 3);

System.***out***.println(str2); // llo

// 如果指定的索引或者个数超出了合理范围，那么将会发生索引越界异常，错误写法：

// String str3 = new String(array, 100, 200);

// 直接赋值

String str4 = "Hello, World!";

System.***out***.println(str4); // Hello, World!

}

}

## 字符串池

由于字符串是对象，而又特别常用，所以为了解决内存中存在大量重复字符串对象的问题，引入了“字符串池”的设计来节省内存。关于字符串池的几点描述：

* 字符串池位于堆当中，里面保存了字符串的若干地址信息。
* 字符串池中绝对不会存放多个重复的字符串。
* 直接使用双引号的字符串默认在池中，而new出来的字符串默认不在池中。

**public** **class** Demo04StringPool {

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

String str1 = "Hello";

String str2 = "Hello";

System.***out***.println(str1 == str2); // true

**char**[] array = { 'H', 'e', 'l', 'l', 'o' };

String str3 = **new** String(array);

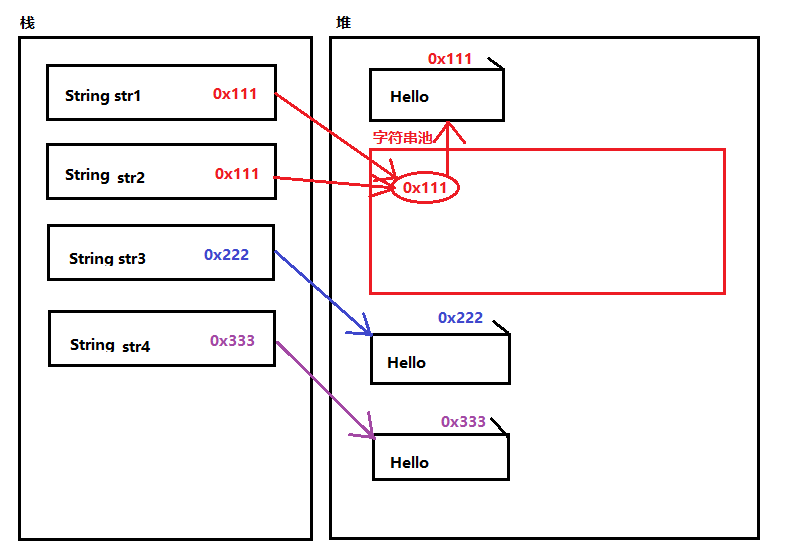
String str4 = **new** String(array);

System.***out***.println(str3 == str4); // false

System.***out***.println(str1 == str3); // false

}

}



## 字符串内容不可变

字符串对象一旦被创建，内容不可改变。下面的代码中，改变的是字符串对象名（其实是一个局部变量）中保存的地址值。

**public** **class** Demo05StringEternal {

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

String str = "Hello";

System.***out***.println(str); // Hello

str = "World";

System.***out***.println(str); // World

String str2 = "Love";

str2 += "Java";

System.***out***.println(str2); // LoveJava

}

}

## 比较方法

要想进行字符串的内容比较，不能使用==进行地址比较，应该使用两个方法：

* **public boolean equals(Object obj)**：区分大小写的内容比较，参数是需要比较的另一个对象（和谁比）。备注：Object类型代表可以接收任意类型，包括字符串。
* **public boolean equalsIgnoreCase(String str)**：忽略大小写的内容比较。

**public** **class** Demo01StringEquals {

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

**char**[] array = { 'a', 'b', 'c' };

String str1 = **new** String(array);

String str2 = "abc";

System.***out***.println(str1 == str2); // false

// 进行内容比较，应该使用equals方法

// 拿着str1和str2进行内容比较。

**boolean** same = str1.equals(str2);

System.***out***.println(same); // true

System.***out***.println(str2.equals(str1)); // true

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

String str3 = **null**;

System.***out***.println("abc".equals(str3)); // false，正确，没有异常

// System.out.println(str3.equals("abc")); // 错误写法！发生异常！

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

String str4 = "Hello";

String str5 = "hello";

System.***out***.println(str4.equals(str5)); // false

System.***out***.println("Hello".equals("hello")); // false

System.***out***.println(str4.equalsIgnoreCase(str5)); // true

}

}

## 练习：模拟登录

使用equals和equalsIgnoreCase方法，即可模拟简单的登录场景。注意两点问题：

* 用户名忽略大小写
* 密码区分大小写

**public** **class** Demo02Login {

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

// 首先需要注册的时候，指定一个正确的用户名，和正确的密码

String registUsername = "admin"; // 注册时候正确的用户名

String registPassword = "123456abc"; // 注册时候正确的密码

// 需要键盘输入两个字符串

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

System.***out***.println("请输入用户名：");

String inputUsername = sc.next(); // 获取键盘输入的用户名字符串

System.***out***.println("请输入密码：");

String inputPassword = sc.next(); // 获取键盘栓书的密码字符串

// 判断一下，两个条件：

// A. 输入的用户名和注册时候的正确用户名必须一样

// （用户名一般不区分大小写，内容判断应该使用equalsIgnoreCase）

// B. 输入的密码和注册时候的正确密码也要一样

// （密码一定是严格区分大小写，所以内容判断应该使用equals）

// A和B两个条件都要同时满足才行：&&

// 既然要判断，肯定使用选择结构：if

**if** (inputUsername.equalsIgnoreCase(registUsername) &&

inputPassword.equals(registPassword)) {

System.***out***.println("欢迎使用！");

} **else** {

System.***out***.println("登录失败！退出！");

}

}

}

## 练习：模拟登录（重试）

如果加入重试次数限制为3次的功能，则需要使用循环：

**public** **class** Demo03LoginRetry {

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

// 首先需要注册的时候，指定一个正确的用户名，和正确的密码

String registUsername = "admin"; // 注册时候正确的用户名

String registPassword = "123456abc"; // 注册时候正确的密码

// 需要键盘输入两个字符串

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

**for** (**int** i = 1; i <= 3; i++) {

System.***out***.println("请输入用户名：");

String inputUsername = sc.next(); // 获取键盘输入的用户名字符串

System.***out***.println("请输入密码：");

String inputPassword = sc.next(); // 获取键盘栓书的密码字符串

**if** (inputUsername.equalsIgnoreCase(registUsername) &&

inputPassword.equals(registPassword)) {

System.***out***.println("登录成功！欢迎使用！");

// 如果登录成功，不应该再循环，应该退出喜欢

**break**;

} **else** { // 如果登录失败

// 区分一下是不是最后一次

// 如果是最后一次，那么就提示重试次数耗尽

// 如果不是最后一次，那么提示剩余次数

**if** (i == 3) { // 如果是最后一次

System.***out***.println("重试次数耗尽，你是盗号的吧？哥屋恩！");

} **else** { // 如果不是最后一次

System.***out***.println("剩余次数" + (3 - i) + "次，请重试。");

}

}

} // for

}

}

## 替换方法

如果希望替换字符串当中的内容，则可以使用方法：

* **public String replace(CharSequence oldStr, CharSequence newStr)**

其中CharSequence现阶段可以简单地认为就是String字符串。而oldStr代表需要被替换的老字符串，newStr代表需要替换成为的新字符串。

**注意**，replace方法会将结果作为一个新的字符串返回。

**public** **class** Demo04StringReplace {

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

String str1 = "How do you do?";

String str2 = str1.replace("o", "\*");

System.***out***.println(str1); // How do you do?

System.***out***.println(str2); // H\*w d\* y\*u d\*?

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

String str3 = "会不会玩儿啊！你大爷！真是服了！";

String str4 = str3.replace("你大爷", "\*\*\*");

System.***out***.println(str4); // 会不会玩儿啊！\*\*\*！真是服了！

}

}

## 切割方法

如果希望将字符串按照指定的标记切分成为若干段儿，可以使用方法：

* **public String[] split(String regex)**：参数regex代表切割标记，返回值是字符串切割后成为的字符串数组。

**注意**，强烈推荐现阶段使用英文逗号作为切割标记，不要使用英文句点，因为英文句点在正则表达式（今后学习）中有特殊含义。

**public** **class** Demo05StringSplit {

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

String str1 = "AAA,BBB,CCC";

String[] array1 = str1.split(",");

System.***out***.println("数组的长度：" + array1.length);

**for** (**int** i = 0; i < array1.length; i++) {

System.***out***.println(array1[i]);

}

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

String str2 = "aaa.bbb.ccc";

String[] array2 = str2.split("\\.");

System.***out***.println("数组的长度：" + array2.length);

**for** (**int** i = 0; i < array2.length; i++) {

System.***out***.println(array2[i]);

}

}

}

# 集合

## 概述

集合与数组类似，也是一种引用类型，也可以存放很多个数据。但是数组的长度不可改变，而集合的长度是可变的，所以使用更加方便。

集合有很多种，现阶段我们学习最为简单的**java.util.ArrayList**集合。

## 构造方法

创建一个ArrayList，最常用的就是默认无参数的构造方法：

* **public ArrayList()**

创建集合的时候要注意尖括号中要指定泛型，也就是里面存放的都是统一的什么数据类型。**注意**，泛型只能是引用类型，不能是基本类型。

**public** **class** Demo01ArrayList {

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

// 创建一个集合，存放的全都是String字符串类型的数据

ArrayList<String> list1 = **new** ArrayList<>();

// 泛型只能写引用类型，不能写基本类型，下面是错误写法！

// ArrayList<int> list2 = new ArrayList<>();

}

}

## 常用方法

集合主要是用来存储数据的，所以常用的方法有：

* **public boolean add(E element)**：向集合中放入一个数据对象，参数是被放入的对象，返回值一定为true。
* **public E get(int index)**：从集合中取出一个对象，参数是对象的索引值（从0开始），返回值是获取的对象。
* **public int size()**：获取集合的长度，返回值是集合的长度int数字，是可以变化的。

**public** **class** Demo02ArrayListMethod {

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

ArrayList<String> list = **new** ArrayList<>();

System.***out***.println(list); // []

// 向集合当中添加元素：add

list.add("鹿晗");

list.add("吴亦凡");

list.add("张艺兴");

System.***out***.println(list); // [鹿晗, 吴亦凡, 张艺兴]

**boolean** success = list.add("关晓彤");

System.***out***.println("添加元素是否成功：" + success); // true

System.***out***.println(list); // [鹿晗, 吴亦凡, 张艺兴, 关晓彤]

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

String name = list.get(1); // 获取第1号元素

System.***out***.println(name); // 吴亦凡

System.***out***.println(list); // [鹿晗, 吴亦凡, 张艺兴, 关晓彤]

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

System.***out***.println("集合的长度：" + list.size()); // 4

list.add("赵丽颖");

list.add("迪丽热巴");

list.add("古力娜扎");

list.add("玛尔扎哈");

System.***out***.println("集合的长度：" + list.size()); // 8

}

}

## 练习：遍历集合字符串

**public** **class** Demo03ArrayListEach {

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

ArrayList<String> list = **new** ArrayList<>();

// 添加：add

list.add("迪丽热巴");

list.add("古力娜扎");

list.add("玛尔扎哈");

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

// 遍历输出每一个字符串对象

// 集合的长度获取：size()

**for** (**int** i = 0; i < list.size(); i++) {

// 获取一个当前元素：get(int)

String name = list.get(i);

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

}

}

}

## 存储基本类型

集合的泛型必须是引用类型，不能是基本类型。如果一定要存储基本类型，需要使用对应的**包装类**（都位于java.lang包下）。基本类型与包装类的对应关系如下：

*  byte Byte
*  short Short
*  int **Integer**
*  long Long
*  float Float
*  long Long
*  char **Character**
*  boolean Boolean

**public** **class** Demo04ArrayListWrapper {

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

ArrayList<Integer> list = **new** ArrayList<>();

list.add(100);

list.add(200);

list.add(300);

**int** result = list.get(1);

System.***out***.println(result); // 200

}

}

## 存储自定义类型

集合可以存储任何引用类型，包括自定义的Person、Student等各种类。

**public** **class** Demo05ArrayListCustom {

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

// 首先创建几个Person对象

Person one = **new** Person("迪丽热巴", 18);

Person two = **new** Person("古力娜扎", 19);

Person three = **new** Person("玛尔扎哈", 200);

// 准备一个集合，用来存放多个Person对象

ArrayList<Person> list = **new** ArrayList<>();

// 将对象添加到集合当中

list.add(one);

list.add(two);

list.add(three);

// 遍历集合

**for** (**int** i = 0; i < list.size(); i++) {

Person per = list.get(i);

System.***out***.println("姓名：" + per.getName() + "，年龄：" + per.getAge());

}

}

}

## 练习：王者荣耀阵容展示

创建一个Hero类代表一位英雄，其中包含有名字和战斗力两个成员变量。使用一个集合存储5位英雄，然后遍历集合展示所有英雄的信息。

**public** **class** Demo01Hero {

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

// 创建一个集合，代表一个队伍，里面有5个英雄

ArrayList<Hero> team = **new** ArrayList<>();

// 创建5个英雄对象

Hero one = **new** Hero("蔡文姬", 200);

Hero two = **new** Hero("貂蝉", 250);

Hero three = **new** Hero("狄仁杰", 360);

Hero four = **new** Hero("东皇太一", 150);

Hero five = **new** Hero("韩信", 220);

// 将5个英雄对象添加到集合中

team.add(one);

team.add(two);

team.add(three);

team.add(four);

team.add(five);

// 遍历集合，进行阵容展示

**for** (**int** i = 0; i < team.size(); i++) {

// 获取当前英雄

Hero hero = team.get(i);

System.***out***.println("英雄名称：" + hero.getName() + "，攻击力：" + hero.getAttack());

}

}

}

## 练习：王者荣耀平均战斗力

仍然是一个集合当中存储5个英雄对象，但是要求不进行遍历，而是进行整个队伍平均战斗力的计算。

**public** **class** Demo02HeroAvg {

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

// 创建一个集合，代表一个队伍，里面有5个英雄

ArrayList<Hero> team = **new** ArrayList<>();

// 创建5个英雄对象

Hero one = **new** Hero("蔡文姬", 200);

Hero two = **new** Hero("貂蝉", 250);

Hero three = **new** Hero("狄仁杰", 360);

Hero four = **new** Hero("东皇太一", 150);

Hero five = **new** Hero("韩信", 220);

// 将5个英雄对象添加到集合中

team.add(one);

team.add(two);

team.add(three);

team.add(four);

team.add(five);

// 需要计算集合当中5个英雄平均战斗力。

// 如何计算战斗力的平均值？

// 1. 需要求出总和，也就是5个战斗力求和

// 2. 将战斗力的总和除以5，得到平均战斗力

**int** total = 0; // 战斗力总和

**for** (**int** i = 0; i < team.size(); i++) {

// 获取当前英雄

Hero hero = team.get(i);

**int** attack = hero.getAttack(); // 获取当前英雄的战斗力

total += attack; // 将战斗力累加到total当中

}

**int** avg = total / 5;

System.***out***.println("平均战斗力：" + avg);

}

}

# 知识总结

1. 只要是字符串，必然就是对象。
2. API文档的基本使用。
3. 如何创建字符串：
   1. 直接赋值双引号，也是一个字符串对象。
   2. 可以通过new关键字来调用String的构造方法：
      1. public String(char[] array)：根据字符数组来创建字符串
      2. public String(char[] array, int offset, int count)：根据字符数组当中的一部分来创建字符串。
4. 字符串池：是为了节省内存，重复利用相同的字符串。
   1. 凡是双引号直接写上的字符串，默认就在池中；
   2. 凡是new出来的字符串，默认不在池中。
5. 字符串的内容永不可变。每当你觉得好像是应该变量，必然是创建了一个新的字符串。
6. 字符串的比较方法：
   1. public boolean equals(Object obj)：严格区分大小写。
   2. public boolean equalsIgnoreCase(String str)：忽略大小写
7. 字符串的替换方法：  
   public String replace(CharSequence oldStr, CharSequence newStr)
8. 字符串的切割方法：  
   public String[] split(String regex)  
   注意：不要使用英文句点作为切分的标识，如果非要用英文句点，应该写[\\.](file:///\\.)
9. 集合：和数组类似，但是主要区别就是集合的长度可变，数组的长度不可变。
10. 如何创建一个ArrayList集合？
    1. 导包：import java.util.ArrayList
    2. 创建：ArrayList<泛型> list = new ArrayList<>()
    3. 泛型：集合当中存储的是统一的什么类型。
    4. 注意：泛型只能使用引用类型，不能使用基本类型。
11. 如何使用ArrayList集合，常用方法：
    1. 添加：public boolean add(E element)
    2. 获取：public E get(int index)
    3. 长度：public int size()
12. 如何让集合存储基本类型：使用包装类。记住两个特例（int对应Integer，char对应的Character），其他都是首字母大写即可。
13. 集合能否存储自定义的类型，例如Person？可以，没问题。