day04-String和StringBuilder和ArrayList

今日内容

- String类-----重要\掌握
  - 构造方法
  - 常用方法
- StringBuilder类-----重要\掌握
  - 构造方法
  - 常用方法
- ArrayList类-----重要\掌握
  - 构造方法
  - 常用方法

教学目标

- 能够知道字符串对象通过构造方法创建和直接赋值的区别
- 能够完成用户登录案例
- 能够完成统计字符串中大写,小写,数字字符的个数
- 能够知道String和StringBuilder的区别
- 能够完成String和StringBuilder的相互转换
- 能够使用StringBuilder完成字符串的拼接
- 能够使用StringBuilder完成字符串的反转
- 能够知道集合和数组的区别
- 能够完成ArrayList集合添加字符串并遍历
- 能够完成ArrayList集合添加学生对象并遍历

知识点--1.String类的常用方法

知识点--1.1 String类概述

目标

- 理解String类概述

路径

- String类的概述

讲解

String类的概述

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。String 类在 java.lang 包下，所以使用的时候不需要导包！

小结

略

知识点--1.2 String类的构造方法

目标

- 掌握String类构造方法的使用

路径

- String类常用的构造方法
- 使用String类的构造方法

讲解

String类常用的构造方法

- 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(byte[] bys)	根据字节数组的内容，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

使用String类的构造方法

```
package com.itheima.demo1_String类的构造方法;

/**
 * @Author：pengzhilin
 * @Date: 2020/9/9 8:44
 * 构造方法通过new来调用
 * 成员方法:
 *       非静态成员方法:通过对象名来调用
 *       静态成员方法:通过类名来调用
 * 方法:
 *       无返回值:直接调用
 *       有返回值:
 *            直接调用
 *            赋值调用
 *            输出调用
 */
public class Test {
    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
            public   String(char[] chs) 根据字符数组的内容，来创建字符串对象
            public   String(char[] value, int offset, int count) 根据指定字符数组范围的内容，来创建字符串对象
            public   String(byte[] bys) 根据字节数组的内容，来创建字符串对象
            public   String(byte[] bytes, int offset, int length)根据指定字节数组范围的内容，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
         */
        // 创建空白字符串对象
```

```
        String str1 = new String();// str1字符串内容: ""
        System.out.println("="+str1+"=");// ==

        // 根据字符数组的内容，来创建字符串对象
        char[] chs = {'a','b','c','d'};
        String str2 = new String(chs);// str2字符串内容:"abcd"
        System.out.println(str2);// abcd

        // 根据指定字符数组范围的内容，来创建字符串对象
        String str3 = new String(chs, 0, 3);// str3字符串内容:"abc"
        System.out.println(str3);// abc

        // 根据字节数组的内容，来创建字符串对象
        byte[] bys = {97,98,99,100,101,102};
        String str4 = new String(bys);// str4字符串内容:"abcdef"
        System.out.println(str4);// abcdef

        // 根据指定字节数组范围的内容，来创建字符串对象
        String str5 = new String(bys, 2, 3);// str5字符串内容:"cde"
        System.out.println(str5);// cde

        // 直接赋值的方式创建字符串对象
        String str6 = "abc";// str6字符串内容:"abc"
        System.out.println(str6);

    }
}
```

小结

略

知识点--1.3 创建字符串对象两种方式的区别

目标

- 能够知道通过**构造方法创建字符串对象**与**直接赋值方式创建字符串对象**的区别

路径

- 通过构造方法创建
- 直接赋值方式创建
- 绘制对比内存图

讲解

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，虽然字符串内容相同，但是地址值不同

```
char[] chs = {'a','b','c'};
String s1 = new String(chs);// s1字符串的内容: abc
String s2 = new String(chs);// s2字符串的内容: abc
// 上面的代码中,JVM首先会先创建一个字符数组,然后每一次new的时候都会有一个新的地址,只不过s1和s2参考的字符串内容是相同的
```

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

```
String s3 = "abc";
String s4 = "abc";
// 上面的代码中,针对第一行代码,JVM会建立一个String对象放在字符串池中,并给s3参考;第二行代码,则让s4直接参考字符串池中String对象,也就是说他们本质上是同一个对象
```

绘制内存图

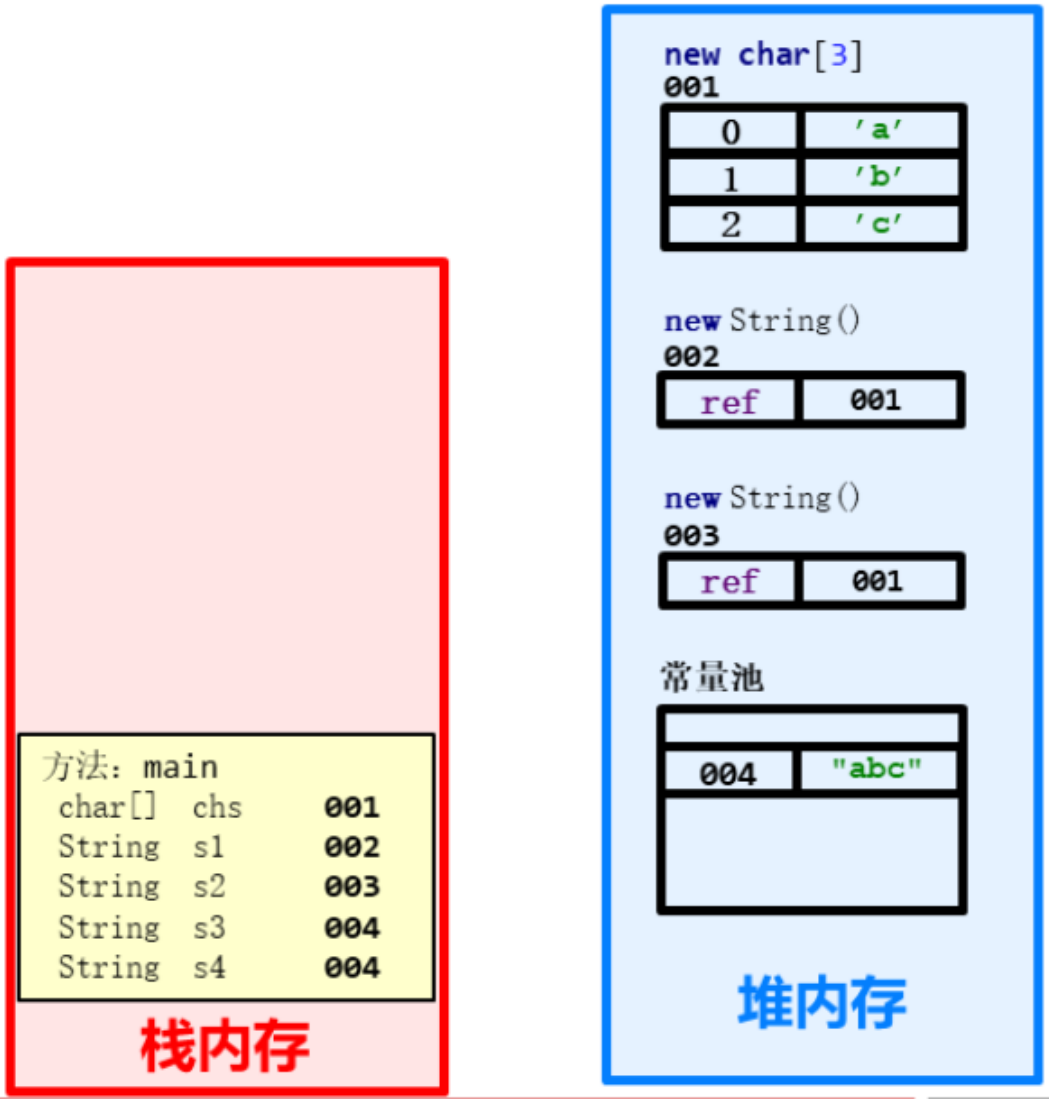
```
package com.itheima.demo2_创建字符串对象两种方式的区别;

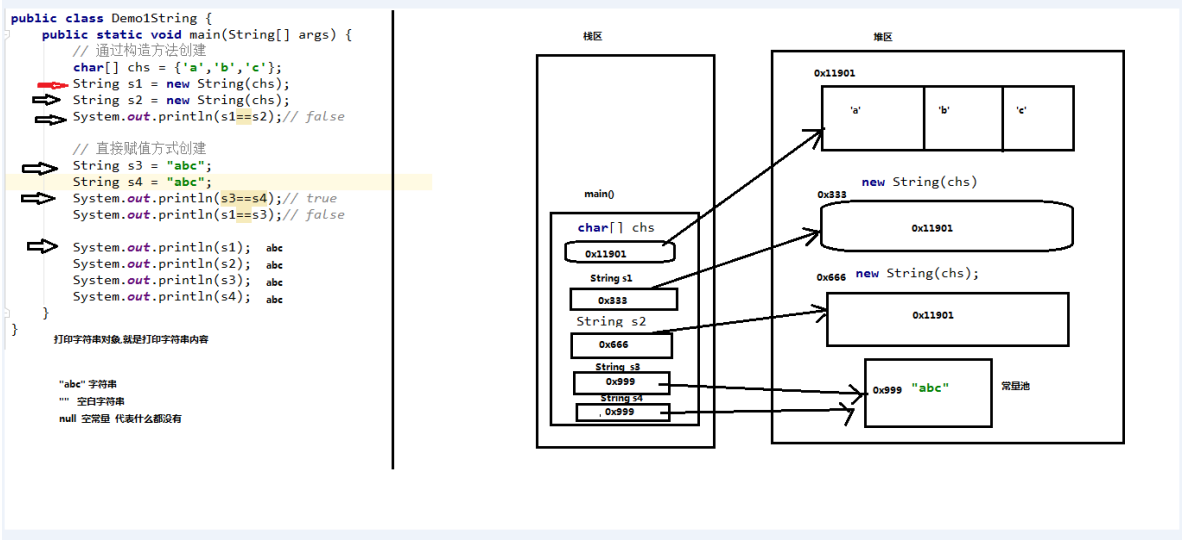
/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 9:05
 */
public class Test {
    public static void main(String[] args) {
        /*
            通过构造方法创建
                通过 new 创建的字符串对象，每一次 new 都会申请一个内存空间，虽然字符串内容相同，但是地址值不同
            直接赋值方式创建
                以“”方式给出的字符串，只要字符序列相同(顺序和大小写)，无论在程序代码中出现几次，
                JVM 都只会建立一个 String 对象，并在字符串池中维护

         */
        // 通过构造方法创建
        char[] chs = {'a','b','c'};
        String s1 = new String(chs);// s1字符串的内容: abc
        String s2 = new String(chs);// s2字符串的内容: abc
        System.out.println(s1 == s2);// 比较s1和s2的地址值 false

        // 直接赋值方式创建
        String str1 = "abc";// str1字符串的内容: abc
        String str2 = "abc";// str2字符串的内容: abc
        System.out.println(str1 == str2);// 比较str1和str2的地址值 true

    }
}
```





小结

- 通过new创建的字符串对象,每一次都会新开辟空间
- 通过""方式直接创建的字符串对象,是常量,在常量池中,只有一份

知识点--1.4 String类的特点

目标

- 能够理解String类的特点

路径

- String类的特点

讲解

String类的特点

- String类的字符串不可变，它们的值在创建后不能被更改

```
String s1 = "abc";
s1 += "d";
System.out.println(s1); // "abcd"
// 内存中有"abc"，"abcd","d",3个对象，s1从指向"abc"，改变指向，指向了"abcd"。
```

- 虽然 String 的值是不可变的，但是它们可以被共享

```
String s1 = "abc";
String s2 = "abc";
// 内存中只有一个"abc"对象被创建，同时被s1和s2共享。
```

- 字符串效果上相当于字符数组(char[])

```
例如：
String str = "abc";

相当于：
char[] data = {'a', 'b', 'c'};
String str = new String(data);
// String底层是靠字符数组实现的,jdk9底层是字节数组。

byte[] bys = {97,98,99};
String str = new String(bys);
```

小结

- String类的字符串不可变，它们的值在创建后不能被更改,**每一次拼接String类的字符串对象,都会创建一个新的字符串对象**
- 虽然 String 的值是不可变的，但是它们可以被共享

知识点--1.5 字符串的比较

目标

- 能够比较2个字符串内容是否相同

路径

- ==号的比较
- equals方法的作用
- equalsIgnoreCase 方法的使用

讲解

==号的比较

- 比较基本数据类型：比较的是具体的值

```
int num1 = 10;
int num2 = 20;
num1 == num2  ===> 10==20  结果:false
```

- 比较引用数据类型：比较的是对象地址值

```
String str1 = new String("abc");
String str2 = new String("abc");
str1 == str2 ===> str1存储的对象地址值 == str2存储的对象地址值  结果: false
```

```
public class StringDemo02 {
    public static void main(String[] args) {
        //构造方法的方式得到对象
        char[] chs = {'a', 'b', 'c'};
        String s1 = new String(chs);
        String s2 = new String(chs);

        //直接赋值的方式得到对象
        String s3 = "abc";
        String s4 = "abc";

        //比较字符串对象地址是否相同
        System.out.println(s1 == s2);//
        System.out.println(s1 == s3);//
        System.out.println(s3 == s4);//
        System.out.println("--------");
    }
}
```

equals方法的作用

- 字符串是对象,它比较内容是否相同,是通过一个方法来实现的,就是equals()方法
- 方法介绍

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

- 示例代码

```
public class StringDemo02 {
    public static void main(String[] args) {
        //构造方法的方式得到对象
        char[] chs = {'a', 'b', 'c'};
        String s1 = new String(chs);
        String s2 = new String(chs);

        //直接赋值的方式得到对象
        String s3 = "abc";
        String s4 = "abc";

        //比较字符串内容是否相同
        System.out.println(s1.equals(s2));// true
        System.out.println(s1.equals(s3));// true
```

```
        System.out.println(s3.equals(s4));// true
    }
}
```

扩展

- `public boolean equalsIgnoreCase (String anotherString)`：将此字符串与指定对象进行比较，忽略大小写。
- 示例

```
public static void main(String[] args) {
        //  public boolean equalsIgnoreCase (String anotherString)
        // 1.创建一个字符串对象
        String str1 = "hello";

        // 2.创建一个字符串对象
        String str2 = "Hello";

        // 3.使用equalsIgnoreCase()方法比较str1和str2字符串是否相等
        boolean res1 = str1.equalsIgnoreCase(str2);
        System.out.println("res1的值是:"+res1);// true
    }
```

小结

- 比较字符串内容是否相等,区分大小写,需要使用String类的equals方法,**千万不要用 == 比较**
- 如果比较字符串内容是否相等,不区分大小写,需要使用String类的equalsIgnoreCase()方法

知识点--1.6 String类获取功能的方法

目标

- 理解String类中各个方法的作用\调用

路径

- 介绍获取功能的方法
- 使用获取功能的方法

讲解

获取功能的方法

```
    - public int length () ：返回此字符串的长度。
    - public String concat (String str) ：将指定的字符串连接到该字符串的末尾。拼接
    - public char charAt (int index) ：返回指定索引处的 char值。
    - public int indexOf (String str) ：返回指定子字符串第一次出现在该字符串内的索引。
    - public int indexOf(String str, int fromIndex)  返回从指定索引位置查找,该子字符串第一次出现在该字符串内的索引。
    - public int lastIndexOf(String str) 返回指定子字符串最后一次出现在该字符串内的索引。
    - public int lastIndexOf(String str, int fromIndex) 返回从指定索引位置查找,,该子字符串最后一次出现在该字符串内的索引。
    - public String substring (int beginIndex) ：返回一个子字符串，从beginIndex开始截取字符串到字符串结尾。
    - public String substring (int beginIndex, int endIndex) ：返回一个子字符串，从beginIndex到endIndex截取字符串。含beginIndex，不含endIndex。
```

使用获取功能的方法

```
package com.itheima.demo4_String类获取功能的方法;

/**
 * @Author：pengzhilin
 * @Date: 2020/9/9 9:56
 */
public class Test {
    public static void main(String[] args) {
        /*
            String类获取功能的方法:
            - public int length () ：返回此字符串的长度。
            - public String concat (String str) ：将指定的字符串连接到该字符串的末尾。拼接
            - public char charAt (int index) ：返回指定索引处的 char值。
            - public int indexOf (String str) ：返回指定子字符串第一次出现在该字符串内的索引。
            - public int indexOf(String str, int fromIndex)  返回从指定索引位置查找,该子字符串第一次出现在该字符串内的索引。
            - public int lastIndexOf(String str) 返回指定子字符串最后一次出现在该字符串内的索引。
            - public int lastIndexOf(String str, int fromIndex) 返回从指定索引位置查找,,该子字符串最后一次出现在该字符串内的索引。
            - public String substring (int beginIndex) ：返回一个子字符串，从beginIndex开始截取字符串到字符串结尾。
            - public String substring (int beginIndex, int endIndex) ：返回一个子字符串，从beginIndex到endIndex截取字符串。含beginIndex，不含endIndex。
         */
        // 创建字符串对象
        String str = "hello-world!";

        // 获取字符串的长度(字符串的字符个数)
        int len = str.length();
        System.out.println("str的字符串长度是:"+len);// 12

        // str字符串末尾拼接上hello-itheima
        String newStr = str.concat("hello-itheima");
        System.out.println("拼接后的字符串:"+newStr);// hello-world!hello-itheima
```

```
        // 获取str中索引为1的字符
        char ch = str.charAt(1);
        System.out.println("索引为1的字符:"+ch);// e


System.out.println("============================================================");
        // 创建字符串对象
        String str1 = "hello-world-hello-itheima-hello-java-hello-itcast";

        // 查找hello第一次出现的索引位置
        int index1 = str1.indexOf("hello");
        System.out.println("hello第一次出现的索引位置:"+index1);// 0

        // 查找hello第二次出现的索引位置
        int index2 = str1.indexOf("hello", index1+1);
        System.out.println("hello第二次出现的索引位置:"+index2);// 12

        // 查找hello第三次出现的索引位置
        int index3 = str1.indexOf("hello", index2+1);
        System.out.println("hello第三次出现的索引位置:"+index3);// 26


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

        // 查找hello最后一次出现的索引位置
        int lastIndex1 = str1.lastIndexOf("hello");
        System.out.println("hello最后一次出现的索引位置:"+lastIndex1);// 37

        // 查找hello倒数第二次出现的索引位置
        int lastIndex2 = str1.lastIndexOf("hello", lastIndex1 - 1);
        System.out.println("hello倒数第二次出现的索引位置:"+lastIndex2);// 26


System.out.println("============================================================");
        String str3 = "hello-world-hello-itheima";

        // 获取world-hello-itheima子字符串
        String subStr1 = str3.substring(6);
        System.out.println("subStr1:"+subStr1);// world-hello-itheima

        // 获取world-hello子字符串
        String subStr2 = str3.substring(6, 17);
        System.out.println("subStr2:"+subStr2);// world-hello

    }
}
```

小结

略

实操--1.7 用户登录案例【应用】

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

分析

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

实现

```
package com.itheima.demo5_练习用户登录案例;

import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 10:24
 */
public class Test {
    public static void main(String[] args) {
        /*
            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
         */
        // 1.已知用户名和密码,定义2个字符串表示
        String username = "admin";
        String password = "123456";

        // 2.由于总共三次机会,循环次数确定,使用for循环来完成
        for (int i = 0; i < 3; i++) {
            // 3.在循环中,用户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.next();
            System.out.println("请输入密码:");
            String pwd = sc.next();

            // 4.在循环中,判断用户输入的用户名和密码和已知的用户名和密码是否相同
            if (username.equals(name) && password.equals(pwd)) {
                // 5.在循环中,如果相同,就登录成功,并结束循环
                System.out.println("恭喜您,登录成功!");
                break;
            } else {
                // i=0  2次机会
                // i=1  1次机会
                // i=2  您的账号已被锁定,请充钱解锁!
                // 5.在循环中,如果失败,就显示提示信息
                if (i == 2){
                    System.out.println("很遗憾,登录失败,您的账号已被锁定,请充钱解锁!");
                }else{
                    System.out.println("很遗憾,登录失败,您还有"+(2-i)+"次机会");
                }
            }
        }
    }
}
```

小结

- equals方法

实操--1.8 遍历字符串案例

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

分析

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
3. 遍历字符串，其次要能够获取到字符串的长度
4. 遍历字符串的通用格式

实现

```
package com.itheima.demo6_遍历字符串案例;

import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 10:46
 */
public class Test {
    public static void main(String[] args) {
        // 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
        // 1.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 str = sc.next();

        // 2.获取字符串的长度(字符个数)
        int len = str.length();

        // 3.循环遍历
        for (int i = 0; i < len; i++) {
            // 4.在循环中,通过charAt方法字符,打印输出
            char c = str.charAt(i);
            System.out.println(c);
        }
    }
}
```

小结

- length()
- charAt()

实操--1.9 统计字符次数案例

需求

键盘录入一个字符串，统计该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

1. 键盘录入一个字符串，用 Scanner 实现
2. 要统计三种类型的字符个数，需定义三个统计变量，初始值都为0
3. 遍历字符串，得到每一个字符
4. 在循环中,判断该字符属于哪种类型，然后对应类型的统计变量+1
    1. 大写字母：ch>='A' && ch<='Z'
    2. 小写字母： ch>='a' && ch<='z'
    3. 数字： ch>='0' && ch<='9'
5. 输出三种类型的字符个数

实现

```
package com.itheima.demo7_统计字符次数案例;

import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 10:50
 */
public class Test {
    public static void main(String[] args) {
        // 需求:键盘录入一个字符串，统计该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)
        // 1.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 str = sc.next();

        // 2.定义一个int类型的变量,用来统计大写字母字符个数
        int count1 = 0;
        // 2.定义一个int类型的变量,用来统计小写字母字符个数
        int count2 = 0;
        // 2.定义一个int类型的变量,用来统计数字字符个数
        int count3 = 0;

        // 3.循环遍历字符串
        for (int i = 0; i < str.length(); i++) {
            // 4.在循环中,获取遍历出来的字符
            char c = str.charAt(i);
            // 5.在循环中,判断该字符:
            if (c >= 'A' && c <= 'Z') {
                // 5.如果该字符是大写字母字符,那么统计大写字母字符个数的变量+1
                count1++;
            } else if (c >= 'a' && c <= 'z') {
                // 5.如果该字符是小写字母字符,那么统计小写字母字符个数的变量+1
                count2++;
            } else if (c >= '0' && c <= '9'){
                // 5.如果该字符是数字字符,那么统计数字字符个数的变量+1
```

```
                count3++;
            }
        }

        System.out.println("大写字母字符个数:"+count1);
        System.out.println("小写字母字符个数:"+count2);
        System.out.println("数字字符个数:"+count3);

    }
}
```

小结

略

实操--1.10 字符串拼接案例

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]

分析

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. 在方法中遍历数组，按照要求进行拼接
4. 调用方法，用一个变量接收结果
5. 输出结果

实现

```
package com.itheima.demo8_字符串拼接案例;

/**
 * @Author：pengzhilin
 * @Date: 2020/9/9 11:18
 */
public class Test {
    public static void main(String[] args) {
        /*
            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]
         */
        int[] arr = {1,2,3};
        System.out.println(arrayToString(arr));// [1, 2, 3]
    }

    public static String arrayToString(int[] arr) {
        // 1.创建空白的字符串对象
        String str = "";
```

```
        // 2.循环遍历传入的数组
        for (int i = 0; i < arr.length; i++) {
            // 3.在循环中,获取遍历出来的元素
            int e = arr[i];
            // 4.判断遍历出来的元素:
            if (i == 0) {
                // 5.如果遍历出来的元素是第一个元素,那么拼接的格式: [ + 元素 + 逗号空格
                str += "[" + e + ", ";
            } else if (i == arr.length - 1) {
                // 5.如果遍历出来的元素是最后一个元素,那么拼接的格式: 元素 + ]
                str += e + "]";
            } else {
                // 5.如果遍历出来的元素是中间元素,那么拼接的格式: 元素 + 逗号空格
                str += e + ", ";
            }
        }

        // 6.返回str
        return str;
    }
}
```

小结

略

实操--1.11 字符串反转案例

需求

定义一个方法，实现字符串反转。键盘录入一个字符串，调用该方法后，在控制台输出结果

例如，键盘录入 abc，输出结果 cba

分析

1. 键盘录入一个字符串，用 Scanner 实现
2. 定义一个方法，实现字符串反转。返回值类型 String，参数 String s
3. 在方法中把字符串倒着遍历，然后把每一个得到的字符拼接成一个字符串并返回
4. 调用方法，用一个变量接收结果
5. 输出结果

实现

```
package com.itheima.demo9_字符串反转案例;

import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 11:27
 */
public class Test {
    public static void main(String[] args) {
        /*
```

```
            需求:
            定义一个方法，实现字符串反转。键盘录入一个字符串，调用该方法后，在控制台输出反转后的字符串
            例如，键盘录入 abc，输出结果 cba
         */
        // 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 str = sc.next();

        // 调用reverse
        System.out.println(reverse(str));
    }

    public static String reverse(String str){
        // 功能:字符串反转
        // 1.创建空白的字符串对象
        String newStr = "";

        // 2.倒叙遍历传入的字符串对象
        for (int i = str.length()-1;i >= 0; i--) {
            // 3.在循环中,获取遍历出来的字符
            char c = str.charAt(i);
            // 4.在循环中,拼接遍历出来的字符
            newStr += c;
        }
        // 5.返回反转后的字符串
        return newStr;
    }

}
```

小结

略

希望--有时间的同学

```
常用的方法：
        concat();  //把两个字符串拼接起来，获取一个新的字符串
        ★length();  //获取字符串的长度(其实就是获取字符串中 字符的个数)
        ★equals();  //比较两个字符串的内容是否相同。      //区分大小写
        equalsIgnoreCase();  //比较两个字符串的内容是否相同。      //忽略大小写
        ★charAt();  //根据给定的索引，获取对应位置的字符
        ★indexOf(); //获取指定的字符 在字符串中 第一次出现的位置(索引)，找不到返回-1
            //int index = a1.indexOf('h');    从头找，'h'第一次出现的位置
            //int index = a1.indexOf('h',3); 从索引为3的元素开始往后找，'h'第一次出现的位置
       lastIndexOf();  //获取指定的字符 在字符串中 最后一次出现的位置(索引)，找不到返回-1
            //int index = a1.lastIndexOf('h');    从尾部找，'h'最后一次出现的位置
            //int index = a1.lastIndexOf('h',3); 从索引为3的元素开始往前找，'h'最后一次出现的位置
       ★substring(); //截取字符串，返回新的字符串
                    //String newStr = a1.substring(2);        //从给定索引，直接截取到字符串末尾
```

```
                    //String newStr = a1.substring(2,5);      //包左不包右(前闭后开), 能取索引2的元素，不能取索引5的元素
        ★isEmpty(); //判断字符串是否为空(长度为0返回true，不为0返回false)
        ★contains();     //判断字符串中是否包含 给定的字符串。
        endsWith(); //判断字符串是否以 给定的字符串 结尾。
        startsWith(); //判断字符串是否以 给定的字符串 开头。

        ★replace(); //用新内容替代旧内容，返回新的字符串
        toLowerCase();  //把字母都转成其对应的小写形式。
        toUpperCase();  //把字母都转成其对应的大写形式。
        toCharArray() // 把字符串转换为数组
        getBytes() // 把字符串转换为字节数组
        ★trim();              //移除首尾空格。
        ★split();    //根据给定的内容，切割字符串，返回字符串数组

    理解的口诀:
        判断功能:    首尾中空加判断

        获取功能:    截长取位取元素

        转换功能:     大小拼串转数组
            //toCharArray(), getBytes();
        其他功能:    除空切换字典拍
            //copareTo();
```

```
package com.itheima.demo10_扩展String类的方法;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 11:44
 */
public class Test {
    public static void main(String[] args) {
        /*
            判断功能的方法
            ★boolean isEmpty(); //判断字符串是否为空(长度为0返回true，不为0返回false)
            ★boolean contains(CharSequence s);     //判断字符串中是否包含 给定的字符串。

            boolean endsWith(String s); //判断字符串是否以 给定的字符串 结尾。
            boolean startsWith(String s); //判断字符串是否以 给定的字符串 开头。
            注意:null不能调用方法,如果调用了就会报空指针异常NullPointerException
         */
        // 判断字符串是否为空
        String str1 = "";// 空字符串
        String str2 = "a";// 非空字符串
        System.out.println(str1.isEmpty());// true
        System.out.println(str2.isEmpty());// false

        System.out.println("====================================");
        String str3 = "hello-world-hello-heima";
        // 判断str3字符串是否包含指定的子字符串
        System.out.println(str3.contains("heima"));// true
        System.out.println(str3.contains("itcast"));// false

        System.out.println("====================================");
        // 判断str3字符串是否以hel开头
        System.out.println(str3.startsWith("hel"));// true
```

```
        // 判断str3字符串是否以hea开头
        System.out.println(str3.startsWith("hea"));// false

        // 判断str3字符串是否以ma结尾
        System.out.println(str3.endsWith("ma"));// true
        // 判断str3字符串是否以mab结尾
        System.out.println(str3.endsWith("mab"));// false
        // Hello.java   hb.jpg
        // 开发中,判断文件类型
        String fileName = "Hello.java";
        System.out.println("判断上传的文件是否是java文
件:"+fileName.endsWith(".java"));// true

    }
}
```

```
package com.itheima.demo10_扩展String类的方法;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 11:57
 */
public class Test2 {
    public static void main(String[] args) {
        /*
            转换功能和其他功能
             ★String replace(char oldChar,char newChar); //用新内容替代旧内容，返回
新的字符串
             ★String replace(CharSequence oldStr,CharSequence newStr); //用新内容
替代旧内容，返回新的字符串
            String toLowerCase();  //把字母都转成其对应的小写形式。
            String toUpperCase();  //把字母都转成其对应的大写形式。
            Char[] toCharArray() // 把字符串转换为数组
            byte[] getBytes() // 把字符串转换为字节数组
            ★String trim();            //移除首尾空格。
            ★String[] split(String regex);   //根据给定的内容，切割字符串，返回字符串
数组
                    参数regex: 分割的规则(普通的字符串,也可以是特殊的字符串(正则表达式))
         */
        // 创建字符串对象
        String str = "hello-world-hello-heima";

        // 把hello替换为hi
        String replaceStr = str.replace("hello", "hi");
        System.out.println("替换后的字符串内容:"+replaceStr);// hi-world-hi-heima
        System.out.println("替换前的字符串内容:"+str);// hello-world-hello-heima

        System.out.println("================================");
        String str1 = "Hello-World-Hello-Heima";
        // 把str1中的字母全部转换为大写字母
        String s1 = str1.toUpperCase();
        System.out.println(s1);// HELLO-WORLD-HELLO-HEIMA

        // 把str1中的字母全部转换为大写字母
        String s2 = str1.toLowerCase();
        System.out.println(s2);// hello-world-hello-heima
```

```
        System.out.println("==================================");
        // 把str1字符串转换为字符数组
        char[] arr1 = str1.toCharArray();
        for (int i = 0; i < arr1.length; i++) {
            System.out.print(arr1[i]+" ");// H e l l o - W o r l d - H e l l o -
H e i m a
        }
        System.out.println();// 换行

        // 把str1字符串转换为字节数组
        byte[] arr2 = str1.getBytes();
        for (int i = 0; i < arr2.length; i++) {
            System.out.print(arr2[i]+" ");// 72 101 108 108 111 45 87 111 114
108 100 45 72 101 108 108 111 45 72 101 105 109 97
        }
        System.out.println();// 换行

        System.out.println("==================================");
        // 移除首尾空格
        String username = "             admin ";
        String trimStr = username.trim();
        System.out.println("="+trimStr+"=");// =admin=

        System.out.println("==================================");
        String str2 = "Hello-World-Hello-Heima";
        // 以-对str2进行分割
        String[] arr = str2.split("-");
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }

        System.out.println("==================================");
        String str3 = "Hello.World.Hello.Heima";
        // 以.对str3进行分割
        String[] arr3 = str3.split("\\.");// .特殊字符串(正则表达式),而在正则表达式中
点表示任意字符
        for (int i = 0; i < arr3.length; i++) {
            System.out.println(arr3[i]);
        }

    }
}
```

知识点--2 StringBuilder类

知识点--2.1 String类字符串拼接问题

目标:

- String类字符串拼接问题

步骤:

- 字符串拼接问题

讲解:

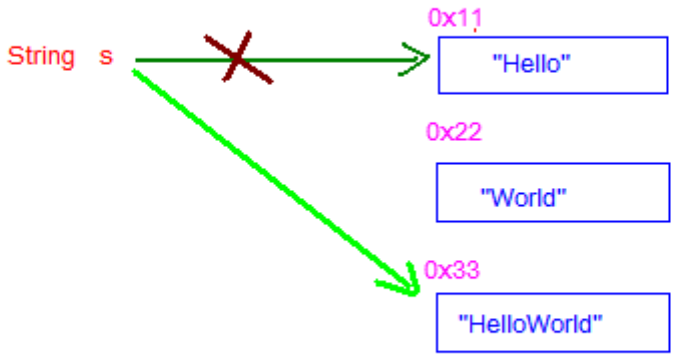
字符串拼接问题

由于String类的对象内容不可改变，所以每当进行字符串拼接时，总是会在内存中创建一个新的对象。例如：

```
public class StringDemo {
    public static void main(String[] args) {
        String s = "Hello";
        s += "World";// s = s + "World";
        System.out.println(s);// HelloWorld
    }
}
```

在API中对String类有这样的描述：字符串是常量，它们的值在创建后不能被更改。

根据这句话分析我们的代码，其实总共产生了三个字符串，
即`"Hello"`、`"World"`和`"HelloWorld"`。引用变量s首先指向`Hello`对象，最终指向拼接出来的新字符串对象，即`HelloWord`。



由此可知，如果对字符串进行拼接操作，每次拼接，都会构建一个新的String对象，**既耗时，又浪费空间。**为了解决这一问题，可以使用`java.lang.StringBuilder`类。

小结

- 结论:
  - String类的字符串拼接,每一次拼接完都会得到一个新的字符串对象,所以比较耗时,也浪费空间

知识点--2.2 StringBuilder类概述以及与String类的区别

目标

- 理解StringBuilder的概述和与String类的区别

路径

- StringBuilder类的概述
- StringBuilder类和String类的区别

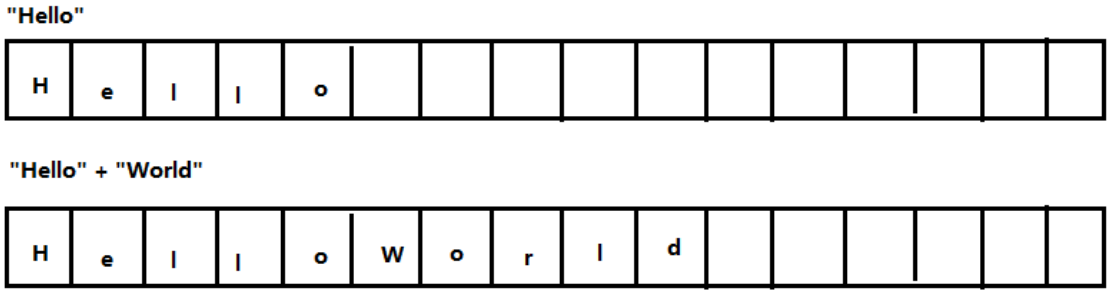
讲解

StringBuilder类概述

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

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



小结

- StringBuilder 是一个可变的字符串类,表示字符串
- StringBuilder 拼接是直接在本身的字符串末尾进行拼接的

知识点--2.3 StringBuilder类的构造方法

目标:

- StringBuilder构造方法

路径:

- StringBuilder构造方法

讲解:

- 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

```
package com.itheima.demo11_StringBuilder类的构造方法;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 12:26
 */
public class Test {
    public static void main(String[] args) {
        /*
            StringBuilder类的构造方法:
```

```
            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
         */
        // 创建空白可变字符串对象  ""
        StringBuilder sb1 = new StringBuilder();
        System.out.println("sb1:"+sb1+"=");// sb1:=
        System.out.println("sb1的长度:"+sb1.length());// 0

        // 把不可变字符串转换为可变字符串
        String str = "itheima";// 不可变的字符串
        StringBuilder sb2 = new StringBuilder(str);// 可变的字符串,字符串内容
是:itheima
        System.out.println("sb2:"+sb2);// sb2:itheima
        System.out.println("sb2的长度:"+sb2.length());// 7

    }
}
```

小结

略

知识点--2.4 StringBuilder类拼接和反转方法

目标:

- StringBuilder类拼接和反转方法

路径:

- StringBuilder类拼接和反转方法

讲解:

- 添加和反转方法

```
public StringBuilder append(任意类型) 拼接数据，并返回对象本身
public StringBuilder insert(int offset, 任意类型) 在指定位置插入数据,并返回对象本
身
public StringBuilder reverse()  反转字符串,并返回对象本身
```

- 拼接数据
- 插入数据
- 反转字符串

```
package com.itheima.demo12_StringBuilder类的常用方法;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 14:51
 */
public class Test {
    public static void main(String[] args) {
        /*
            public StringBuilder append(任意类型) 拼接数据，并返回对象本身
```

```
            public StringBuilder insert(int offset, 任意类型) 在指定位置插入数据,并返回对象本身
            public StringBuilder reverse()  反转字符串,并返回对象本身
         */
        // 创建可变字符串对象
        StringBuilder sb1 = new StringBuilder("hello");

        // sb1字符串末尾拼接上"-world"
        StringBuilder sb2 = sb1.append("-world");
        System.out.println(sb1);// hello-world
        System.out.println(sb2);// hello-world
        System.out.println(sb1==sb2);// sb1和sb2的地址值是一样的,true

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

        // 把-java插入到sb1可变字符串中,使得sb1为:hello-java-world
        sb1.insert(5,"-java");
        System.out.println(sb1);// hello-java-world

        System.out.println("========================================");
        // 把sb1中的字符串内容进行反转
        StringBuilder sb3 = sb1.reverse();
        System.out.println(sb1);// dlrow-avaj-olleh
        System.out.println(sb3);// dlrow-avaj-olleh
        System.out.println(sb1==sb3);// true

    }
}
```

小结

```
StringBuilder类常用方法:
    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
    public StringBuilder insert(int offset, 任意类型) 在指定位置插入数据,并返回对象本身
```

知识点--2.5 StringBuilder和String相互转换

目标:

- StringBuilder和String相互转换

路径:

- StringBuilder转换为String
- String转换为StringBuilder

讲解:

String转换为StringBuilder

`public StringBuilder(String s)`：通过StringBuilder的构造方法就可以实现把 String 转换为 StringBuilder

StringBuilder转换为String

`public String toString()`：通过StringBuilder类中的 toString() 就可以实现把 StringBuilder 转换为 String

- 示例代码

```
package com.itheima.demo13_StringBuilder和String相互转换;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 15:12
 */
public class Test {
    public static void main(String[] args) {
        /*
            StringBuilder和String相互转换:
                String--->StringBuilder:  public StringBuilder(String str)
                StringBuilder-->String :  public String toString()
         */
        //String--->StringBuilder:  public StringBuilder(String str)
        String str = "hello-world";
        StringBuilder sb = new StringBuilder(str);

        //StringBuilder-->String :  public String toString()
        String s = sb.toString();
        System.out.println(sb);// hello-world  可变字符串
        System.out.println(s);//  hello-world  不可变的字符串
    }
}
```

小结

- ```
  StringBuilder:
      public StringBuilder(String str);  String--->StringBuilder
      public String toString();    StringBuilder--->String
  ```

实操--2.6 字符串拼接升级版案例

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]

分析

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
  ```

实现

```
package com.itheima.demo14_字符串拼接升级版案例;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 15:11
 */
public class Test {
    public static void main(String[] args) {
        /*
                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]

         */
        int[] arr = {1,2,3};
        System.out.println(arrayToString(arr));
    }

    public static String arrayToString(int[] arr) {
        // 1.创建可变的空白字符串对象
        StringBuilder sb = new StringBuilder();

        // 2.循环遍历传入的数组元素
        for (int i = 0; i < arr.length; i++) {
            // 3.在循环中,获取遍历出来的元素
            int e = arr[i];
            // 4.在循环中,判断该元素:
            if (i == 0) {
                // 5.如果遍历出来的元素是第一个元素,那么拼接的格式为: [ + 元素 + 逗号空格
                sb.append("[").append(e).append(", ");
            } else if (i == arr.length - 1) {
                // 5.如果遍历出来的元素是最后一个元素,那么拼接的格式为: 元素 + ]
                sb.append(e).append("]");
            } else {
                // 5.如果遍历出来的元素是中间元素,那么拼接的格式为: 元素 + 逗号空格
                sb.append(e).append(", ");
            }

        }
        // 6.返回字符串
        return sb.toString();
    }
}
```

小结

略

实操--2.7 字符串反转升级版案例

需求

定义一个方法，实现字符串反转。键盘录入一个字符串，调用该方法后，在控制台输出结果

例如，键盘录入abc，输出结果 cba

分析

1. 键盘录入一个字符串，用 Scanner 实现
2. 定义一个方法，实现字符串反转。返回值类型 String，参数 String s
3. 在方法中用StringBuilder实现字符串的反转，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

实现

```
package com.itheima.demo15_字符串反转升级版案例;

import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 15:24
 */
public class Test {
    public static void main(String[] args) {
        /*
            需求
                定义一个方法，实现字符串反转。键盘录入一个字符串，调用该方法后，在控制台输出
结果
                例如，键盘录入abc，输出结果 cba
         */
        // 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 str = sc.next();
        // 反转
        System.out.println(reverse(str));
    }

    public static String reverse(String str){
        // 1.根据传入的字符串内容,创建对应的可变字符串
        // StringBuilder sb = new StringBuilder(str);
        // 2.调用reverse方法
        // sb.reverse();
        // 3.返回反转后的字符串
        // return sb.toString();

        return new StringBuilder(str).reverse().toString();
    }
}
```

小结

略

知识点--3. ArrayList

知识点--3.1 集合的概述以及与数组的区别

目标

- 能够了解集合的概念和知道集合与数组的区别

路径

- 集合的概述
- 集合与数组的区别

讲解

集合的概述

- 思考: 编程的时候如果要存储多个数据,首先我们想到的就是数组,但是数组的特点是长度固定,所以数组不一定能满足我们的需求,更适应不了变化的需求,那么应该如何选择呢?
- 什么是集合
    - 集合其实就是一个大小可变的容器,可以用来存储多个引用数据类型的数据

集合与数组的区别

- 数组：数组大小固定
- 集合：集合大小可动态扩展

小结

- 集合: 是一个大小可变的容器,可以用来存储多个引用类型的数据
- 集合和数组的区别:
    - 数组大小是固定
    - 集合大小是可变

知识点--3.2 ArrayList类概述

目标

- 能够理解ArrayList类的概念

路径

- ArrayList类概述

讲解

集合类有很多,目前我们先学习一个: `ArrayList`

ArrayList类概述

- 通过查看ArrayList类的API:
    - `java.util.ArrayList <E>` ： 该类需要 import导入使后使用。
    - 表示一种未知的数据类型,叫做泛型,用于约束集合中存储元素的数据类型
    - 怎么用呢?
        - 在出现E的地方我们使用**引用数据类型**替换即可，表示我们将存储哪种引用类型的元素。
        - 例如:
            - ArrayList 表示ArrayList集合中只能存储String类型的对象

        - ArrayList 表示ArrayList集合中只能存储Student类型的对象
        - .....
        - ArrayList 编译报错,因为E这个位置只能写引用数据类型
    - 概述:
        - ArrayList类底层是大小可变的数组的实现，存储在内的数据称为元素。也就是说 ArrayList 中可以不断添加元素，其大小会自动增长。

小结

- 概述: ArrayList类底层是一个大小可变的数组实现
- 使用ArrayList类的时候,在E出现的位置使用引用数据类型替换,表示该集合可以存储哪种引用类型的元素

知识点--3.3 ArrayList类构造方法

目标

- 能够使用ArrayList类的构造方法创建ArrayList集合对象

路径

- 介绍ArrayList的构造方法
- 案例演示

讲解

- 介绍ArrayList的构造方法
    - `ArrayList()` 构造一个初始容量为 10 的空列表。
- 案例演示

```
package com.itheima.demo16_ArrayList类构造方法;

import java.util.ArrayList;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 15:48
 */
class Student{

}
public class Test {
    public static void main(String[] args) {
        /*
            ArrayList类构造方法:
                    public ArrayList() 构造一个初始容量为 10 的空列表。

                    基本数据类型             对应             包装类类型
                    byte                                      Byte
                    short                                     Short
                    int                                       Integer
                    long                                      Long
                    float                                     Float
                    double                                    Double
                    boolean                                   Boolean
                    char                                      Character
```

```
         */
        // 创建ArrayList集合对象,限制集合中元素的类型为String类型
        ArrayList<String> list1 = new ArrayList<String>();
        ArrayList<String> list2 = new ArrayList<>();

        // 创建ArrayList集合对象,限制集合中元素的类型为Student类型
        ArrayList<Student> list3 = new ArrayList<>();

        //  创建ArrayList集合对象,限制集合中元素的类型为int类型
        //ArrayList<int> list4 = new ArrayList<int>();// 编译报错
        //  集合中存储基本数据类型
        // 创建ArrayList集合对象,限制集合中元素的类型为Integer类型
        ArrayList<Integer> list5 = new ArrayList<>();// 存储int类型的数据

        // 创建ArrayList集合对象,不限制集合中元素的类型
        ArrayList list6 = new ArrayList();// 存储任意类型的对象

    }
}
```

小结

- 略

知识点-- 3.4 ArrayList类添加元素方法

目标

- 能够往ArrayList集合中添加元素

路径

- ArrayList添加元素的方法

讲解

- ArrayList添加元素的方法
  - public boolean add(E e)：将指定的元素追加到此集合的末尾
  - public void add(int index,E element)：在此集合中的指定位置插入指定的元素
- 案例演示:

```
package com.itheima.demo17_ArrayList类添加元素方法;

import java.util.ArrayList;

/**
 * @Author：pengzhilin
 * @Date: 2020/9/9 15:59
 */
public class Test {
    public static void main(String[] args) {
        /*
            ArrayList<E>添加元素的方法
                - public boolean add(E e)：将指定的元素追加到此集合的末尾
                - public void add(int index,E element)：在此集合中的指定位置插入
指定的元素
```

```
         */
        // 创建ArrayList集合,限制集合中元素的类型为String类型
        ArrayList<String> list = new ArrayList<>();

        // 往list集合中添加元素
        list.add("谢霆锋");
        list.add("王宝强");
        list.add("贾乃亮");
        System.out.println(list);// [谢霆锋, 王宝强, 贾乃亮]

        // 把陈羽凡插入在索引为1的位置
        list.add(1,"陈羽凡");
        System.out.println(list);// [谢霆锋, 陈羽凡, 王宝强, 贾乃亮]


    }
}
```

小结

- 略

扩展--集合存储基本数据类型以及指定位置添加元素的注意事项

```
public class Test {
    public static void main(String[] args) {
        // 创建ArrayList集合对象,限制集合中元素的类型为Integer类型
        ArrayList<Integer> list = new ArrayList<>();

        // 往集合中添加元素
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);
        System.out.println("list:"+list);// list:[10, 20, 30, 40]

        // 思考一:指定索引为4的位置添加一个元素50,是否可以?---可以
        list.add(4,50);// 相当于list.add(50);
        System.out.println("list:"+list);// list:[10, 20, 30, 40, 50]

        // 思考二:指定索引为6的位置添加一个元素100,是否可以?----不可以
        //list.add(6,100);// 运行报异常,IndexOutOfBoundsException索引越界异常

    }
}
```

知识点--3.5 ArrayList类常用方法

目标

- 会使用ArrayList常用方法

步骤

- ArrayList常用方法

讲解

ArrayList常用方法

```
public boolean   remove(Object o) 删除指定的元素，返回删除是否成功
public E   remove(int   index) 删除指定索引处的元素，返回被删除的元素
public E   set(int index, E  element) 修改指定索引处的元素，返回被修改的元素
public E   get(int   index) 返回指定索引处的元素
public int   size() 返回集合中的元素的个数
```

示例代码

```
package com.itheima.demo19_ArrayList类常用方法;

import java.util.ArrayList;

/**
 * @Author：pengzhilin
 * @Date: 2020/9/9 16:15
 */
public class Test {
    public static void main(String[] args) {
        /*
            ArrayList类常用方法:
            public boolean   remove(Object o) 删除指定的元素，返回删除是否成功
            public E   remove(int   index) 删除指定索引处的元素，返回被删除的元素
            public E   set(int index, E  element) 修改指定索引处的元素，返回被修改的元
素
            public E   get(int   index) 返回指定索引处的元素
            public int   size() 返回集合中的元素的个数
         */
        // 创建ArrayList集合,限制集合中元素的类型为String类型
        ArrayList<String> list = new ArrayList<>();

        // 往list集合中添加元素
        list.add("谢霆锋");
        list.add("王宝强");
        list.add("贾乃亮");
        System.out.println(list);// [谢霆锋, 王宝强, 贾乃亮]

        // 需求1:删除王宝强这个元素
        boolean res1 = list.remove("王宝强");
        System.out.println("是否删除成功:"+res1);// true
        System.out.println(list);//  [谢霆锋, 贾乃亮]

        // 需求2:删除索引为1的元素
        String removeE = list.remove(1);
        System.out.println("被删除的元素:"+removeE);// 贾乃亮
        System.out.println(list);// [谢霆锋]

        // 需求3: 修改0索引上的元素为陈冠希
        String setE = list.set(0, "陈冠希");
        System.out.println("被替换的元素:"+setE);// 谢霆锋
```

```
        System.out.println(list);// [陈冠希]

        // 需求4:获取索引为0的元素
        String e = list.get(0);
        System.out.println("索引为0的元素:"+e);// 陈冠希

        // 需求5:统计集合中元素的个数
        int size = list.size();
        System.out.println("集合中元素的个数:"+size);// 1
    }
}
```

- 注意事项

```
package com.itheima.demo19_ArrayList类常用方法;

import java.util.ArrayList;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 16:22
 */
public class Test2 {
    public static void main(String[] args) {
        //  public boolean remove(Object o) 删除指定的元素，返回删除是否成功
        //  public E   remove(int   index) 删除指定索引处的元素，返回被删除的元素
        // 1.创建ArrayList集合对象,限制集合中元素的类型为Integer
        ArrayList<Integer> list = new ArrayList<>();

        // 2.往集合中存储int类型的整数
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        System.out.println(list);// [1, 2, 3, 4]

        list.remove(1);// 删除索引为1的元素
        System.out.println(list);// [1, 3, 4]

        // 删除1这个元素,需要传入1的Integer对象
        list.remove(new Integer(1));
        System.out.println(list);// [3, 4]
    }
}
```

小结

```
ArrayList常用方法:
    public boolean remove(Object o)   删除指定的元素，返回删除是否成功
    public E   remove(int   index)    删除指定索引处的元素，返回被删除的元素
    public E   set(int index, E  element) 修改指定索引处的元素，返回被修改的元素
    public E   get(int   index) 返回指定索引处的元素
    public int   size() 返回集合中的元素的个数
```

实操-3.6 ArrayList存储字符串并遍历

需求

- 创建一个存储字符串的集合，存储3个字符串元素，使用程序实现在控制台遍历该集合

分析

1. 创建集合对象
2. 往集合中添加字符串对象
3. 遍历集合，首先要能够获取到集合中的每一个元素，这个通过get(int index)方法实现
4. 遍历集合，其次要能够获取到集合的长度，这个通过size()方法实现
5. 遍历集合的通用格式

实现

```
package com.itheima.demo20_ArrayList存储字符串并遍历;

import java.util.ArrayList;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 16:27
 */
public class Test {
    public static void main(String[] args) {
        // 需求:创建一个存储字符串的集合，存储3个字符串元素，使用程序实现在控制台遍历该集合
        // 1.创建ArrayList集合,限制集合中元素的类型为String类型
        ArrayList<String> list = new ArrayList<>();

        // 2.往集合中存储3个元素
        list.add("张柏芝");
        list.add("马蓉");
        list.add("李小璐");

        // 3.获取集合中元素的个数
        int size = list.size();

        // 4.循环遍历
        for (int i = 0; i < size; i++) {
            // 5.在循环中,获取元素,打印输出
            String e = list.get(i);
            System.out.println(e);
        }
    }
}
```

小结

略

实操--3.7 ArrayList存储学生对象并遍历

需求

- 创建一个存储学生对象的集合，存储3个学生对象，使用程序实现在控制台遍历该集合

分析

1. 定义学生类
2. 创建集合对象
3. 创建学生对象
4. 添加学生对象到集合中
5. 遍历集合，采用通用遍历格式实现

实现

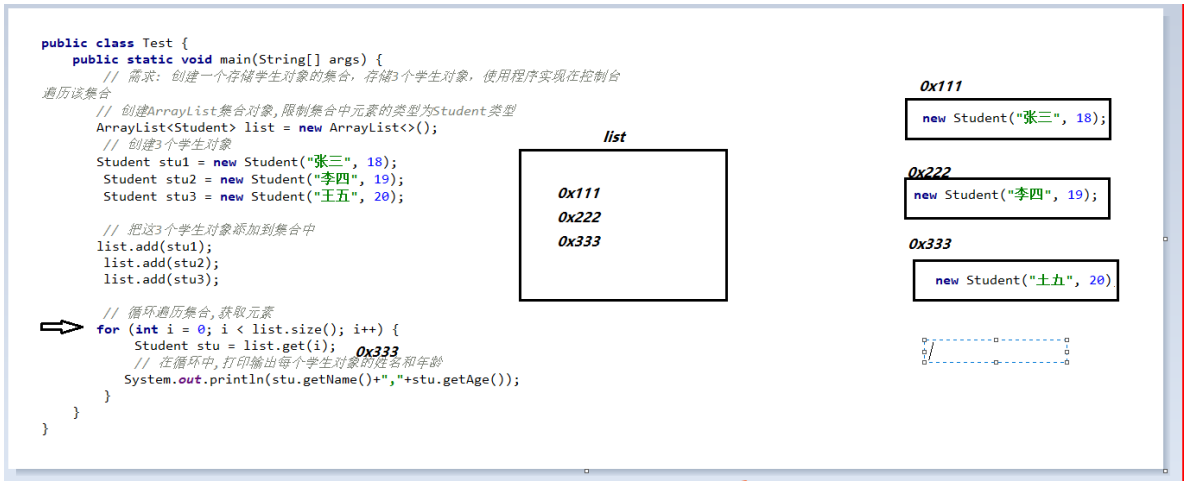
```
public class Test {
    public static void main(String[] args) {
        /*
            需求
                - 创建一个存储学生对象的集合，存储3个学生对象，使用程序实现在控制台遍历该集合
         */
        // 创建ArrayList集合,限制集合中元素的类型为Student类型
        ArrayList<Student> list = new ArrayList<>();

        // 往集合中存储3个学生对象
        Student stu1 = new Student("张三",18);
        Student stu2 = new Student("李四",19);
        Student stu3 = new Student("王五",20);

        list.add(stu1);
        list.add(stu2);
        list.add(stu3);

        // 循环遍历集合元素,打印输出
        for (int i = 0; i < list.size(); i++) {
            // 获取元素
            Student stu = list.get(i);
            System.out.println("姓名:"+stu.getName()+",年龄:"+stu.getAge());
        }

    }
}
```



小结

略

实操--3.8 ArrayList存储学生对象并遍历升级版

需求

- 创建一个存储学生对象的集合，存储3个学生对象，使用程序实现在控制台遍历该集合 (学生的姓名和年龄来自于键盘录入)

分析

1. 定义学生类，为了键盘录入数据方便，把学生类中的成员变量都定义为String类型
2. 创建集合对象
3. 键盘录入学生对象所需要的数据
4. 创建学生对象，把键盘录入的数据赋值给学生对象的成员变量
5. 往集合中添加学生对象
6. 遍历集合，采用通用遍历格式实现

实现

```
package com.itheima.demo22_ArrayList存储学生对象并遍历升级版;

import java.util.ArrayList;
import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2020/9/9 16:48
 */
public class Test {
    public static void main(String[] args) {
        /*
            需求
            - 创建一个存储学生对象的集合，存储3个学生对象，使用程序实现在控制台遍历该集合
            ( 学生的姓名和年龄来自于键盘录入)
         */
        // 1.创建ArrayList集合,限制集合中元素的类型为Student
        ArrayList<Student> list = new ArrayList<>();

        // 2.键盘录入姓名和年龄， 创建3个学生对象,并添加到集合中
        Scanner sc = new Scanner(System.in);
```

```
        // 添加学生
        addStudent(list, sc);
        addStudent(list, sc);
        addStudent(list, sc);


        // 3.循环遍历集合
        for (int i = 0; i < list.size(); i++) {
            // 5.在循环中,获取集合中的元素,打印输出
            Student stu = list.get(i);
            System.out.println(stu.getName() + "," + stu.getAge());
        }

    }

    // 定义一个方法,实现键盘录入学生姓名和年龄,创建学生对象,并添加到集合中
    public static void addStudent(ArrayList<Student> list, Scanner sc) {
        System.out.println("请输入姓名:");
        String name = sc.next();
        System.out.println("请输入年龄:");
        int age = sc.nextInt();
        // 创建学生对象
        Student stu = new Student(name, age);
        // 往集合中添加这个学生对象
        list.add(stu);
    }

    private static void method02() {
        // 1.创建ArrayList集合,限制集合中元素的类型为Student
        ArrayList<Student> list = new ArrayList<>();

        // 2.键盘录入姓名和年龄, 创建3个学生对象,并添加到集合中
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 3; i++) {
            System.out.println("请输入姓名:");
            String name = sc.next();
            System.out.println("请输入年龄:");
            int age = sc.nextInt();
            // 创建学生对象
            Student stu = new Student(name, age);
            // 往集合中添加这个学生对象
            list.add(stu);
        }


        // 3.循环遍历集合
        for (int i = 0; i < list.size(); i++) {
            // 5.在循环中,获取集合中的元素,打印输出
            Student stu = list.get(i);
            System.out.println(stu.getName() + "," + stu.getAge());
        }
    }

    // 抽取代码到方法中: 选中代码-->ctrl+alt+m-->输入方法名,回车即可
    private static void method01() {
        // 1.创建ArrayList集合,限制集合中元素的类型为Student
```

```
        ArrayList<Student> list = new ArrayList<>();

        // 2.键盘录入姓名和年龄，创建3个学生对象
        Scanner sc = new Scanner(System.in);

        System.out.println("请输入姓名:");
        String name1 = sc.next();
        System.out.println("请输入年龄:");
        int age1 = sc.nextInt();

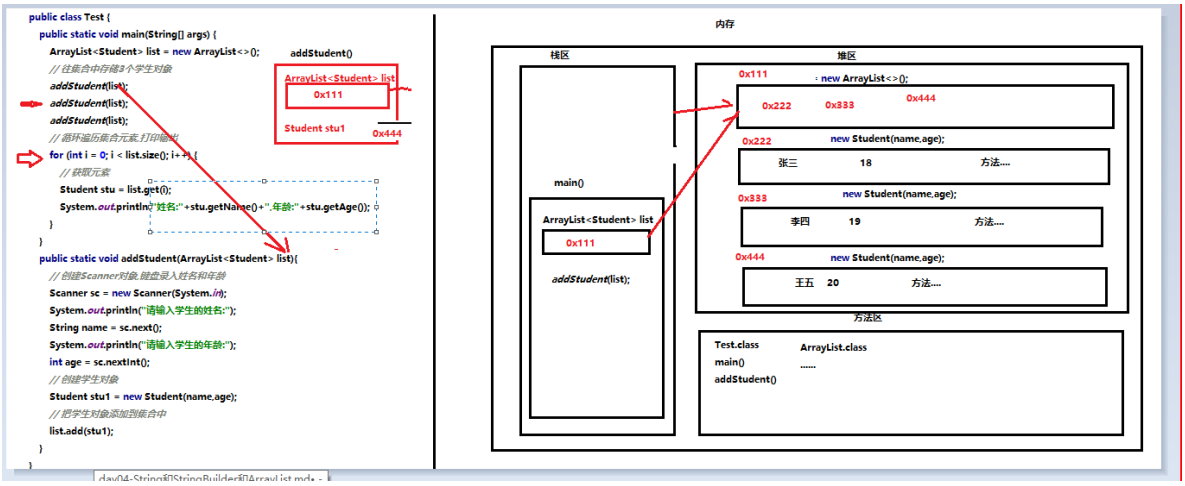
        Student stu1 = new Student(name1, age1);

        System.out.println("请输入姓名:");
        String name2 = sc.next();
        System.out.println("请输入年龄:");
        int age2 = sc.nextInt();
        Student stu2 = new Student(name2, age2);

        System.out.println("请输入姓名:");
        String name3 = sc.next();
        System.out.println("请输入年龄:");
        int age3 = sc.nextInt();
        Student stu3 = new Student(name3, age3);

        // 3.往集合中添加这3个学生对象
        list.add(stu1);
        list.add(stu2);
        list.add(stu3);

        // 4.循环遍历集合
        for (int i = 0; i < list.size(); i++) {
            // 5.在循环中,获取集合中的元素,打印输出
            Student stu = list.get(i);
            System.out.println(stu.getName() + "," + stu.getAge());
        }
    }
}
```



小结

略

总结

必须练习：
String类
    1.概述:表示不可变的字符串
    2.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(char[] value, int offset, int count) 根据指定字符数组范围的内容，来创建字符串对象
        public   String(byte[] bys) 根据字节数组的内容，来创建字符串对象
        public   String(byte[] bytes, int offset, int length)根据指定字节数组范围的内容，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
    3.成员方法:
        concat(); //把两个字符串拼接起来，获取一个新的字符串
        ★length(); //获取字符串的长度(其实就是获取字符串中 字符的个数)
        ★equals(); //比较两个字符串的内容是否相同。   //区分大小写
        equalsIgnoreCase(); //比较两个字符串的内容是否相同。   //忽略大小写
        ★charAt(); //根据给定的索引，获取对应位置的字符
        ★indexOf(); //获取指定的字符 在字符串中 第一次出现的位置(索引)，找不到返回-1
            //int index = a1.indexOf('h');   从头找，'h'第一次出现的位置
            //int index = a1.indexOf('h',3); 从索引为3的元素开始往后找，'h'第一次出现的位置
       lastIndexOf(); //获取指定的字符 在字符串中 最后一次出现的位置(索引)，找不到返回-1
            //int index = a1.lastIndexOf('h');   从尾部找，'h'最后一次出现的位置
            //int index = a1.lastIndexOf('h',3); 从索引为3的元素开始往前找，'h'最后一次出现的位置
       ★substring(); //截取字符串，返回新的字符串
                    //String newStr = a1.substring(2);   //从给定索引，直接截取到字符串末尾
                    //String newStr = a1.substring(2,5);   //包左不包右(前闭后开)，能取索引2的元素，不能取索引5的元素
        ★isEmpty(); //判断字符串是否为空(长度为0返回true，不为0返回false)
        ★contains();   //判断字符串中是否包含 给定的字符串。
        endsWith(); //判断字符串是否以 给定的字符串 结尾。
        startsWith(); //判断字符串是否以 给定的字符串 开头。

        ★replace(); //用新内容替代旧内容，返回新的字符串
        toLowerCase(); //把字母都转成其对应的小写形式。
        toUpperCase(); //把字母都转成其对应的大写形式。
        toCharArray() // 把字符串转换为数组
        getBytes() // 把字符串转换为字节数组
        ★trim();   //移除首尾空格。
        ★split();   //根据给定的内容，切割字符串，返回字符串数组

StringBuilder类
    1.概述:表示可变字符串
    2.构造方法:
        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
    3.成员方法:
            public StringBuilder append(任意类型) 拼接数据，并返回对象本身
            public StringBuilder insert(int offset, 任意类型) 在指定位置插入数据,并返回对象本身
            public StringBuilder reverse() 反转字符串,并返回对象本身

```
            public String toString()
ArrayList<E>类: E,在使用的时候,需要使用引用数据类型来代替,表示集合中能存储的元素类型
    1.概述:表示集合,大小不固定,可以扩容
    2.构造方法:
        public ArrayList();
    3.成员方法:
        - public boolean add(E e):将指定的元素追加到此集合的末尾
        - public void add(int index,E element):在此集合中的指定位置插入指定的元素
          public boolean   remove(Object o) 删除指定的元素，返回删除是否成功
          public E   remove(int   index) 删除指定索引处的元素，返回被删除的元素
          public E   set(int index, E  element) 修改指定索引处的元素，返回被修改的元素
          public E   get(int   index) 返回指定索引处的元素
          public int   size() 返回集合中的元素的个数

* 构造方法通过new来调用
 * 成员方法:
 *       非静态成员方法:通过对象名来调用
 *       静态成员方法:通过类名来调用
 * 方法:
 *       无返回值:直接调用
 *       有返回值:
 *           直接调用
 *           赋值调用
 *           输出调用


- 能够知道字符串对象通过构造方法创建和直接赋值的区别
    通过构造方法创建: 在堆区
    直接赋值: 在常量池

- 能够完成用户登录案例
    0.定义2个已知的用户名和密码字符串
    1.键盘录入用户名和密码
    2.判断用户输入的用户名和密码是否和已知的用户名和密码相等
    3.如果相等,登录成功
    4.如果不相等,登录失败

- 能够完成统计字符串中大写,小写,数字字符的个数
    1.定义三个int变量,分别用来统计大写,小写,数字字符的个数
    2.循环遍历字符串
    3.在循环中,获取遍历出来的字符
    4.判断该字符:
    5.如果是大写...,对应的统计变量+1
    6.如果是小写...,对应的统计变量+1
    7.如果是数字...,对应的统计变量+1

- 能够知道String和StringBuilder的区别
      String:不可变字符串
      StringBuilder:可变字符串

- 能够完成String和StringBuilder的相互转换
      String---StringBuilder: public StringBuilder(String str)
      StringBuilder--String: public String toString();

- 能够使用StringBuilder完成字符串的拼接
  StringBuilder append(任意数据类型)

- 能够使用StringBuilder完成字符串的反转
```

```
    StringBuilder reverse();
```

- 能够知道集合和数组的区别
  数字: 大小固定,不可变
  集合; 大小不固定,可变

- 能够完成ArrayList集合添加字符串并遍历
  ```
  add(E e)   size()   get(int index)
  ```

- 能够完成ArrayList集合添加学生对象并遍历
  ```
  add(E e)   size()   get(int index)
  ```