一、String定义变量存储字符串

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

```
1. Java写好的技术(功能代码)，咱们可以直接调用。
2. Oracle 也为Java提供的这些功能代码提供了相应的 API文档(程序使用说明书) 。
```

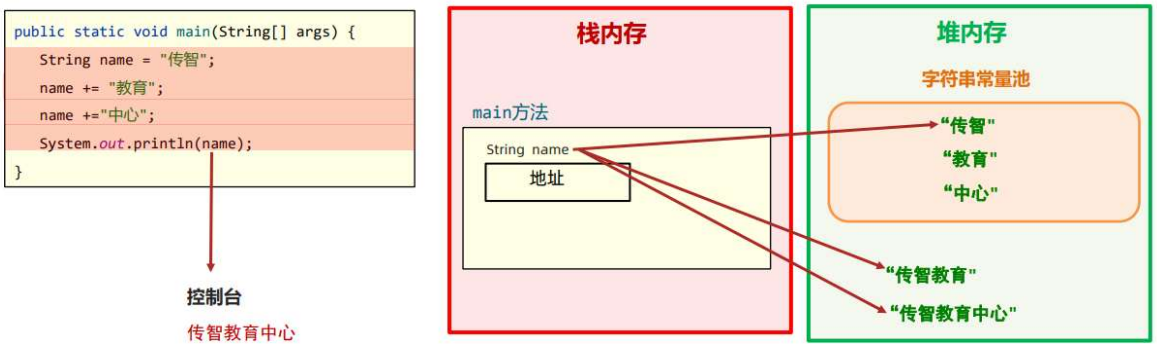
- String类的概述

```
1. java.lang.String 类代表字符串，String类定义的变量可以用于指向字符串对象，同时String类提供了很多操作字符串的功能
2. Java 程序中的所有字符串文字（例如“abc”）都为此类的对象。
   这种是存放由于字符串常量池中;new String("xx")是存放于堆中的,产生了两个对象，一个在字符串常量池中，一个在堆中
```

- 一些特点

```
1. String其实常被称为不可变字符串类型，它的对象在创建后不能被更改。
        String变量每次的修改其实都是产生并指向了新的字符串对象。
        原来的字符串对象 都是没有改变的，所以称不可变字符串
2. 字符串常量池中的字符通过拼接(运算)，产生一个新的字符串存放于堆中
    先创建StringBuilder实例用于拼接字符串，然后再把StringBuilder转换成字符串
```

  - 以 “” 方式给出的字符串对象，在字符串常量池中存储。




```
/**
 * ClassName: StringDemo
 * Description:不可变字符串验证
 * date:2022/3/5
 */

@Test
public void test1() {
    String name = "1";
    //String类重写了Object的toString方法，所以打出来的不是地址，而是字符串
    System.out.println("1的hashcode" + "1".hashCode());
    System.out.println("1的hashcode" + name.hashCode());
    name += "2";
    System.out.println("2的hashcode" + "2".hashCode());
    System.out.println("12的hashcode" + name.hashCode());
```

```
        name += "3";
        System.out.println("3的hashcode" + "3".hashCode());
        System.out.println("123的hashcode" + name.hashCode());
    }
========================================
1的hashcode49
1的hashcode49
2的hashcode50
12的hashcode1569
3的hashcode51
123的hashcode48690
```

- 创建String对象的两种方式

```
1. 直接通过双引号获得String对象（常用）
2. 通过String类的构造器创建对象
```

构造器	说明
public String()	创建一个空白字符串对象，不含有任何内容
public String(String original)	根据传入的字符串内容，来创建字符串对象
public String(char[] chs)	根据字符数组的内容，来创建字符串对象
public String(byte[] chs)	根据字节数组的内容，来创建字符串对象

```
    /**
     * ClassName: StringDemo
     * Description:创建String对象的两种方式
     * date:2022/3/6
     */

    @Test
    public void test2() throws Exception {
        java.lang.String name = "苏未晓";
        System.out.println("name1=" + name);

        final String name2 = new String();//相当于name2 = ""
        System.out.println("name2=" + name2);

        final String name3 = new String("苏未晓");
        System.out.println("name3=" + name3);

        //字符数组
        char[] chars = {'我', '是', '你'};
        final String name4 = new String(chars);
        System.out.println("name4=" + name4);
        //字节数组
        byte[] bytes = {65, 97, 66, 98, 46, 47, 48, 49, 50};
        //将字节数组按特定编码转为字符串，指定起始位置，长度
        final String name5 = new String(bytes, 4, 5, "utf-8");
        System.out.println("name5=" + name5);
    }
```

```
==========================================================
name1=苏未晓
name2=
name3=苏未晓
name4=我是你
name5=./012
```

- 两种创建对象的方式有什么区别?

```
 /**
     * ClassName: StringDemo
     * Description: String类的两种创建方式的区别
     * date:2022/3/6
     */
    @Test
    public void test3() {
        String s1 = "abc";
        String s2 = "abc";
        //以“”方式给出的字符串对象，在字符串常量池中存储，而且相同内容只会在其中存储一份 true
        System.out.println(s1 == s2);
        //字符数组
        char[] chs = {'a', 'b', 'c'};//数组的静态初始化3
        String s3 = new String(chs);
        String s4 = new String(chs);
        System.out.println(s3 == s4); //通过构造器new对象，每new一次都会产生一个新对象，
放在堆内存中 false
    }
```

- 小结

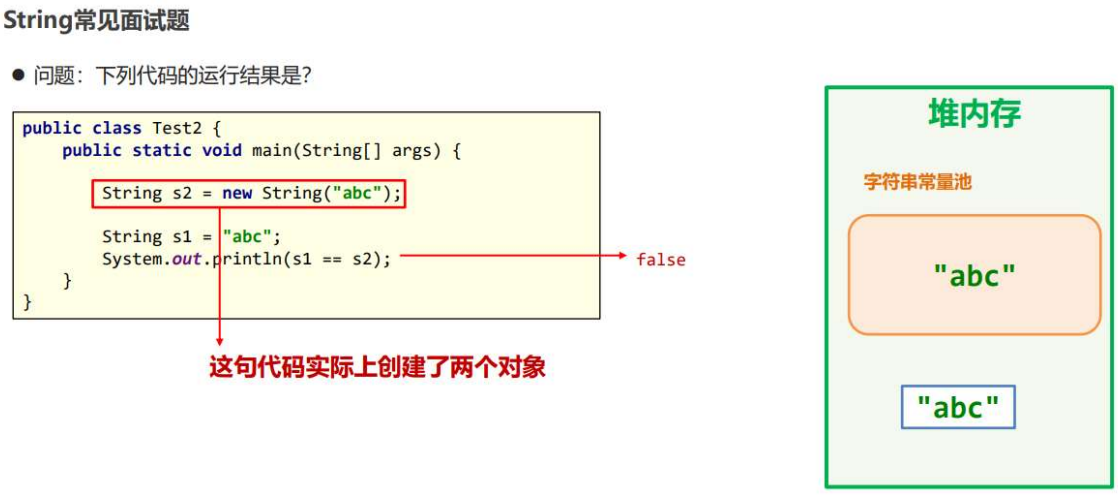
```
1. 以“”方式给出的字符串对象，在字符串常量池中存储，而且相同内容只会在其中存储一份。
2. 通过构造器new对象，每new一次都会产生一个新对象，放在堆内存中
```

- 常见面试题

```
    @Test
    public void test4() {
        //创建了两个对象，字面量在字符串常量池中，构造器的在堆中
        final String s = new String("abc");
        //String的hashCode()方法是返回对象的哈希码
        System.out.println("s的hashcode：" + s.hashCode());
        //没有创建对象，因为字符串常量池中已经有了
        final String s2 = "abc";
        //String的hashCode()方法是返回对象的哈希码
        System.out.println("s2的hashcode：" + s2.hashCode());
        //  == 比较的是地址，s的对象在堆内存中，s2的对象在字符串常量池中，所以不同
        //String重写了equals方法，比较的是字符串的内容
        System.out.println(s == s2);
    }
==================================================
s的hashcode：96354
s2的hashcode：96354
```

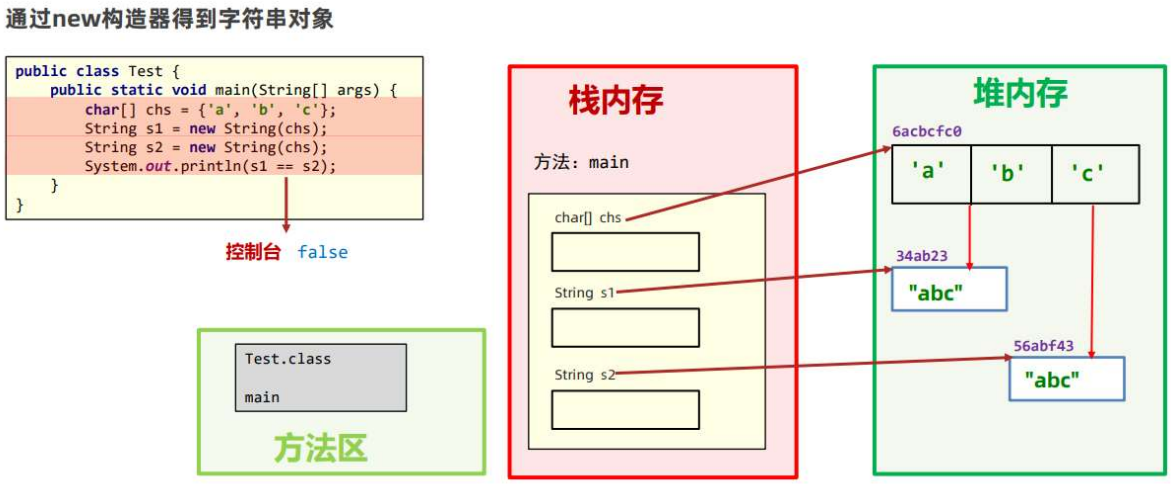
```
false
```

```java
@Test
public void test5() {
    String s1 = "abc";
    String s2 = "ab";
    //字符串在进行运算（拼接）时，产生的
    //通过StringBuilder将s2和”c“拼接起来，再转为字符串
    String s3 = s2 + "c";
    System.out.println(s1 == s3);
    //因为都是字面量值，java存在编译优化机制，在编译期间就可以将字符串拼接成 “abc”
    String s4 = "a" + "b" + "c";
    System.out.println(s1 == s4);
}
```



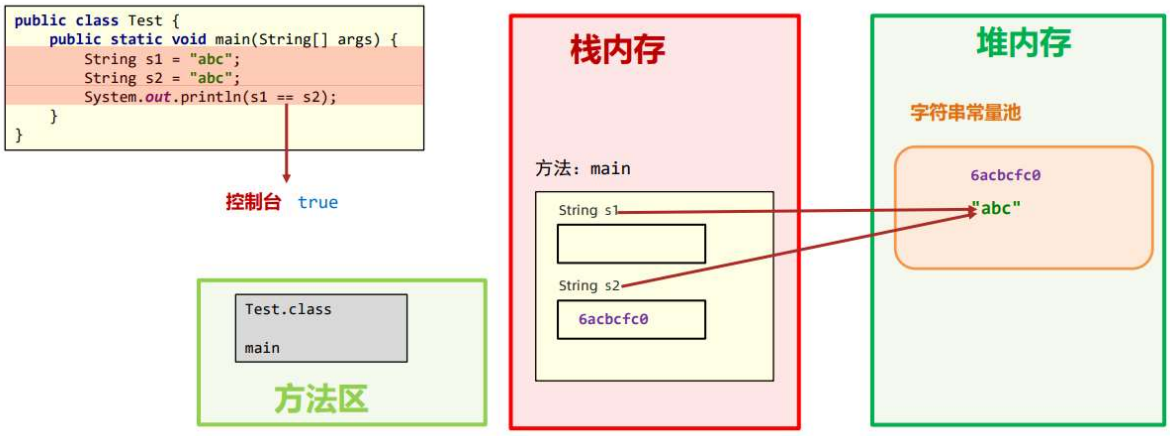
String的内存原理

- 通过String类的构造器得到String对象



- 通过双引号（字面量）的形式获得String对象

通过“”定义字符串内存原理



String类的常用API

方法名	说明
public boolean equals (Object anObject)	将此字符串与指定对象进行比较。只关心字符内容是否一致!
public boolean equalsIgnoreCase (String anotherString)	将此字符串与指定对象进行比较，忽略大小写比较字符串。只关心字符内容是否一致!

1. == 是用来比较引用数据类型的地址，基本数据类型的数据值
2. String 的equals方法用于比较内容是否相同，Object的equals方法比较是地址，因为底层还是调用了==;

String常用API

方法名	说明
public int length()	返回此字符串的长度
public char charAt(int index)	获取某个索引位置处的字符
public char[] toCharArray():	将当前字符串转换成字符数组返回
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（包前不包后）
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，得到字符串数组返回

```
    @Test
    public void test6() {
        String s = "我是一个不知道是什么的人";
        //String的length属性得到的是字符串长度
        for (int i = 0; i < s.length(); i++) {
            //String的charAt方法是获取字符串某个索引的字符的值
            System.out.print(s.charAt(i));
        }
    }
==========
//我是一个不知道是什么的人
```

```
    @Test
    public void test7() {
        String info = "竹杖芒鞋轻胜马";
        final String information = new String("竹杖芒鞋轻胜马");
        //将字符串转为字符数组
        final char[] charArray = info.toCharArray();
        System.out.println(Arrays.toString(charArray));//[竹, 杖, 芒, 鞋, 轻, 胜,
马]
        //截取字符串，只有一个参数就是，参数值索引到结尾
        final String s1 = info.substring(4);
        System.out.println(s1);//轻胜马
        //截取字符串，两个参数就是，开始索引到结束索引 开始下标和结束下标（不是长度）包前不包后
        final String s2 = info.substring(2, 4);
        System.out.println(s2);//芒鞋
        //字符串替换，将字符串中的某些字符替换成某些字符
        final String s3 = info.replace("马", "*");
        System.out.println(s3);//竹杖芒鞋轻胜*
        //判断字符串是否包含某些字符
        System.out.println(info.contains("胜"));//true
        //判断字符串是否以某些字符打头的
        System.out.println(info.startsWith("竹杖芒"));//true
        System.out.println(info.startsWith("杖芒"));//false
        //判断字符串是否以某些字符结尾的
        System.out.println(info.endsWith("马"));//true
        System.out.println(info.endsWith("胜"));//false
        //判断两个字符串的内容是否相等
        System.out.println(info.equals(information));//true
        //判断两个字符串地址是否相等
        System.out.println(info == information);//false
    }
```

String解决实际案例

```
    /**
     * ClassName: StringDemo
     * Description:三次输入验证登录
     * date:2022/3/6
     */

    private static void login() {
        final Scanner sc = new Scanner(System.in);
        final String username = "admin";
        final String password = "admin";
        int count = 0;
        while (true) {
            System.out.println("请输入用户名：");
            final String name = sc.next();
            System.out.println("请输入密码：");
            final String pwd = sc.next();
            if (username.equals(name) && password.equals(pwd)) {
                System.out.println("登陆成功！");
                break;
            }
            System.out.println("第" + ++count + "输入错误");
            //第一次过来是一；
            if (count >= 3) {
```

```
                System.out.println("次数已用完");
                break;
            }
        }
    }
```

```
    /**
     * ClassName: StringDemo
     * Description:验证码
     * date:2022/3/6
     */

    @Test
    public void test8() {
        String info =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
        final Random r = new Random();
        String code = new String();
        for (int i = 0; i < 5; i++) {
            final int s = r.nextInt(info.length());
            code += info.charAt(s);
        }
        System.out.println(code);
    }
/*
next() 方法遇见第一个有效字符（非空格，非换行符）时，开始扫描，当遇见第一个分隔符或结束符(空格或换行符)时，结束扫描，获取扫描到的内容，即获得第一个扫描到的不含空格、换行符的单个字符串。
可以看到，nextLine()自动读取了被next()去掉的Enter作为他的结束符，所以没办法给s2从键盘输入值。

经过验证，我发现其他的next的方法，如double nextDouble() ， float nextFloat() ， int nextInt() 等与nextLine()连用时都存在这个问题，解决的办法是：在每一个 next()、nextDouble()、 nextFloat()、nextInt() 等语句之后加一个nextLine()语句，将被next()去掉的Enter结束符过滤掉
*/
```

集合概述

```
1. 集合是与数组类似，也是一种容器，用于装数据的
   数组定义完成并启动后，类型确定、长度固定。
问题：在个数不能确定，且要进行增删数据操作的时候，数组是不太合适的
2. 集合的大小不固定，启动后可以动态变化，类型也（可以）选择不固定。
3. 集合非常适合做元素个数不确定，且要进行增删操作的业务场景
```

ArrayList快速入门

方法名	说明
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

```
    @Test
```

```
    public void test1() {
        List list = new ArrayList();
        //可以添加任意类型元素，但不建议，所以有泛型这种东西
        list.add("交易延迟 毫无疑问");
        list.add('我');
        list.add(100);
        list.add(100L);
        list.add(1.0);
        list.add(1.0F);
        list.add(false);//存什么就什么，printWriter
        list.add(1, "deal late no doubt");//指定索引，插入数据
        System.out.println(list);//集合重写了toString方法，会直接输出内容
    }
==========================================
[交易延迟 毫无疑问, deal late no doubt, 我, 100, 100, 1.0, 1.0, false]
```

ArrayList对于泛型的支持

```
ArrayList<E>：其实就是一个泛型类可以在编译阶段约束集合对象只能操作某种数据类型
            ArrayList<String> ：此集合只能操作字符串类型的元素。
            ArrayList<Integer>：此集合只能操作整数类型的元素
注意：集合中只能存储引用类型，不支持基本数据类型 泛型也不支持基本数据类型(所以出现了包装类)
```

ArrayList常用API

ArrayList集合常用方法

方法名称	说明
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
public E remove(int index)	删除指定索引处的元素，返回被删除的元素
public boolean remove(Object o)	删除指定的元素，返回删除是否成功
public E set(int index,E element)	修改指定索引处的元素，返回被修改的元素

```
增:boolean add(E e)
删:boolean remove(Object o);E remove(int index);
查:E get(int index)
改:set(int index,E e)
长度:size()
```

- 常用api代码

```
    /**
     * ClassName: ArrayListDemo
     * Description:ArrayList的常用API
     * date:2022/3/6
     */
```

```
    @Test
    public void test2() {
        final List<String> list = new ArrayList<>();
        list.add("java");
        list.add("java");
        list.add("oracle");
        list.add("phthon");

        final List list2 = list;//绕过泛型添加其他类型的变量，另外一种方法是通过反射绕过编
译阶段
        list2.add(1);

        final Object o1 = list.get(1);
        //集合中第二个位置的是字符串类型
        //字符串强转必须判断类型
        Object a1 = o1 instanceof String ? o1 : null;
        //String类重写了toString方法，多态，对于方法而言，运行看右边，编译看左边 跑子类的方
法
        System.out.println(a1);

        final Object o = list.get(4);
        //集合中第二个位置的是整形的包装类
        Object a = o instanceof Integer ? o : null;
        //包装类重写了toString方法，多态，对于方法而言，运行看右边，编译看左边 跑子类的方
        System.out.println(a);法
        //println方法，在输出前先使用String.valueOf,这个在判断不为空是就会调用该类的
toString方法 [java, java,
oracle, phthon, 1]
        System.out.println(list);
        System.out.println(list.size());//5
        System.out.println(list.remove(2));//删除集合中指定索引的元素，返回元素
        System.out.println(list.remove("java"));//删除集合中指定名字的元素（由前往后
找，第一个），返回是否成功
        list.set(0, "123");//修改指定索引下的元素值，返回原来的元素值    java
    }
```

ArrayList遍历

- arraylist遍历删除元素

```
    /**
     * ClassName: ArrayListDemo2
     * Description:集合遍历删除，第一种方法 正向删
     * date:2022/3/6
     */
    @Test
    public void test1() {
        final List<Integer> list = new ArrayList<>();
        Collections.addAll(list, 98, 77, 66, 89, 79, 50, 100);
        System.out.println(list);
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i) < 80) {
                list.remove(i);
                i--;//每删除一位就往前移一位，防止漏删
            }
        }
```

```
        System.out.println(list);
    }


    /**
     * ClassName: ArrayListDemo2
     * Description:集合遍历删除，第一种方法 逆向删
     * date:2022/3/6
     */

    @Test
    public void test2() {
        final List<Integer> list = new ArrayList<>();
        Collections.addAll(list, 98, 77, 66, 89, 79, 50, 100);
        System.out.println(list);
        for (int i = list.size() - 1; i >= 0; i--) {
            if (list.get(i) <= 80) {
                list.remove(i);
            }
        }
        System.out.println(list);
    }
```

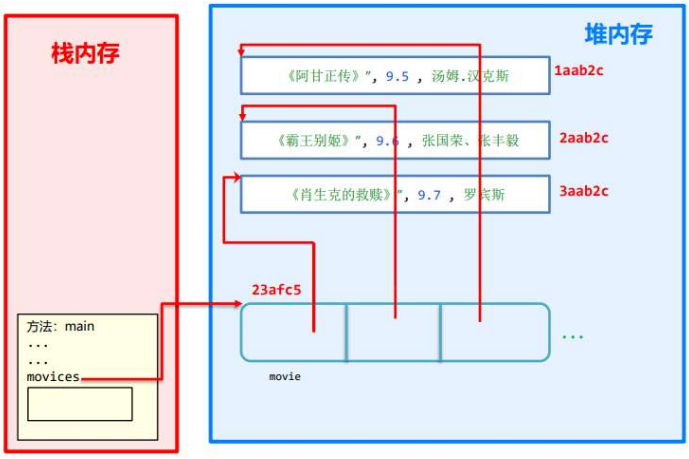
- ArrayList存储自定义类型

```
public class Movie {
    private String name;
    private double score;
    private String acotr;

    public Movie(String name, double score, String acotr) {
        this.name = name;
        this.score = score;
        this.acotr = acotr;
    }
    // ... getter + setter
}
```

```
public class SystemDemo {
    public static void main(String[] args) {
        ArrayList <Movie> movies = new ArrayList<>();
        movies.add(new Movie("《肖生克的救赎》", 9.7 , "罗宾斯"));
        movies.add(new Movie("《霸王别姬》", 9.6 , "张国荣、张丰毅"));
        movies.add(new Movie("《阿甘正传》", 9.5 , "汤姆.汉克斯"));
        System.out.println(movies);

        for (int i = 0; i < movies.size(); i++ ) {
            Movie movie = movies.get(i);
            System.out.println("片名：" + movie.getName());
            System.out.println("评分：" + movie.getScore());
            System.out.println("主演：" + movie.getAcotr());
        }
    }
}
```



结论：集合中存储的元素并不是对象本身，而是对象的地址。