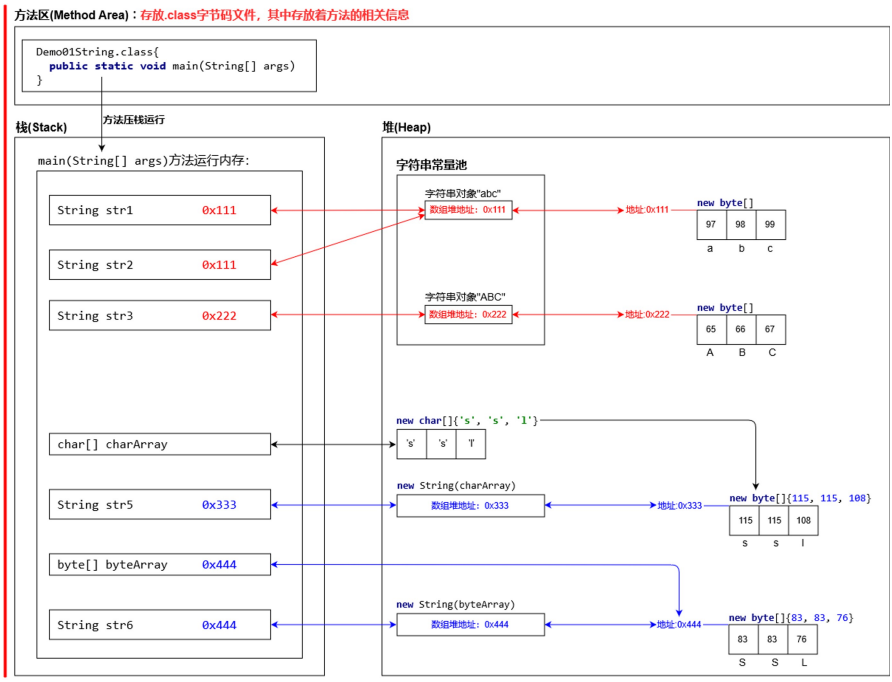
day08_String类、static、Arrays类、Math类

一.String类

java.lang.String 类代表字符串，Java程序中的所有字符串字面值（如 "abc" 等）都作为此类的实例实现。即程序当中带有双引号的字符串常量，都是String类的对象。（就算没有new，也照样是）

1.字符串对象的创建

```java
package demo01;

/*
创建字符串常见的"3+1"种方式：

(1).直接创建：直接写上双引号，就是字符串对象。
String str = "Hello";

(2).利用三种构造方法创建：
public String(): 创建一个空白字符串，不含有任何内容。
public String(char[] array): 根据字符数组的内容，来创建对应的字符串。
public String(byte[] array): 根据字节数组的内容，来创建对应的字符串。

 */
public class Demo01String {
    public static void main(String[] args) {

        // (1).直接创建
        String str1 = "abc";
        String str2 = "abc";
        String str3 = "ABC";

        // (2).根据空参构造方法创建一个空字符串
        String str4 = new String();

        // (3).根据字符数组创建字符串
        char[] charArray = new char[]{'s', 's', 'l'};
        String str5 = new String(charArray);

        // (4).根据字节数组创建字符串
        byte[] byteArray = new byte[]{115, 115, 108};
        String str6 = new String(byteArray);

        // 直接输出字符串的引用类型变量，结果是字符串内容，而不是地址值
        System.out.println(str1); // abc

        //对于基本类型来说，==是进行数值的比较。对于引用类型来说，==是进行【地址值】的比较
        System.out.println(str1 == str2); // true
        System.out.println(str1 == str3); // false

    }
}
```



(1).所有字符串对象都是以字节数组的形式存储在堆内存中的，并且字符串对象当中保存的是字节数组的堆地址。(即String类的底层原理是字节数组)

a.对于直接创建的字符串对象：

- 凡是带有双引号的字符串，就是在"字符串常量池"中创建一个字符串对象，并返回该字符串对象保存的数组堆地址，交由相应的引用类型变量存储。(如：String str1 = "abc")
- 先判断该字符串是否在字符串常量池中。若存在，则直接返回已存在字符串中的数组堆地址；否则，就创建一个新的字符串对象到字符串常量池中。(如：String str2 = "abc"、String str3 = "ABC")

b.对于通过构造方法new出来的字符串对象：

- new出来的字符串对象不在常量池中。
- 使用字符数组char[]创建的字符串对象，将字符数组char[]先转化为相应的字节数组byte[]，将该字符串以字节数组的形式存储。

(2).由字符串对象的存储方式可知，字符串主要有以下特点：

a.字符串的内容永不可变。

我们对字符串进行重新赋值：str2 = "ABC" ； 字符串的内容不是已经被改变了吗？我们说字符串的内容永不可变，是指在常量池中的字符串对象永不可变。str2只是字符串对象的引用，str2 = "ABC" 只是将引用str2指向了另一个字符串对象"ABC"，位于常量池中的字符串对象本身并不会改变。这就是字符串的内容永不可变的含义。

b.正是因为字符串内容不可改变，所以字符串可以共享使用。

位于字符串常量池中的字符串对象可以被多个引用类型变量共享使用。如：String str1 = "abc"、String str2 = "abc" ，字符串常量池中只有一个"abc"对象被创建，同时被str1和str2共享使用。

2.String类的常用成员方法

(1).字符串的比较相关方法：

== 比较运算符，对于基本数据类型比较的是值，对于引用数据类型比较的是对象的地址值。如果确实需要字符串的内容比较，可以使用以下方法：

- public boolean equals(Object obj)：与指定字符串进行内容比较
- public boolean equalsIgnoreCase(String str)：与指定字符串进行内容比较，忽略大小写

#.注意：

a.任何对象传入方法，都能使用Object类进行接收

b.equals方法具有对称性，也就是a.equals(b)和b.equals(a)效果一样

c.如果比较双方一个常量一个变量，推荐把常量字符串写在前面。因为当"str = null"时，前者不会报错，后者会发生对象空指针异常。

推荐：`"abc".equals(str)`

不推荐：`str.equals("abc")`

(2).字符串的获取相关方法：

- `public int length()`：获取当前字符串的字符个数，即字符串长度
- `public String concat(String str)`：将当前字符串的尾部拼接上参数字符串，并返回新的字符串
- `public char charAt(int index)`：获取指定索引位置的单个字符（索引从0开始）
- `public int indexOf(String str)`：查找参数字符串在当前字符串中首次出现的索引位置，如果没有返回-1值

(3).字符串的截取相关方法：

- `public String subString(int index)`：从参数索引位置开始截取，一直到字符串末尾，并返回新的字符串
- `public String subString(int begin, int end)`：从左闭右开区间[begin, end)截取，并返回新的字符串

#.注意：被截取的字符串并不会改变，截取出来的字符串是在常量池中创建了一个新字符串。

(4).字符串的转换相关方法：

- `public char[] toCharArray()`：将当前字符串拆分成为字符数组并返回
- `public byte[] getBytes()`：获得当前字符串底层的字节数组并返回
- `public String replace(CharSequence oldString, CharSequence newString)`：将所有出现的oldString替换成为newString，并返回替换后的字符串
  CharSequence是接口，意思是可以接受字符串类型

(5).字符串的分割相关方法：

- `public String[] split(String regex)`：按照参数的规则，将字符串切分成为若干部分，并放入到String[]数组里面返回

#.注意：

a.split方法的参数其实是一个 "正则表达式"，今后学习

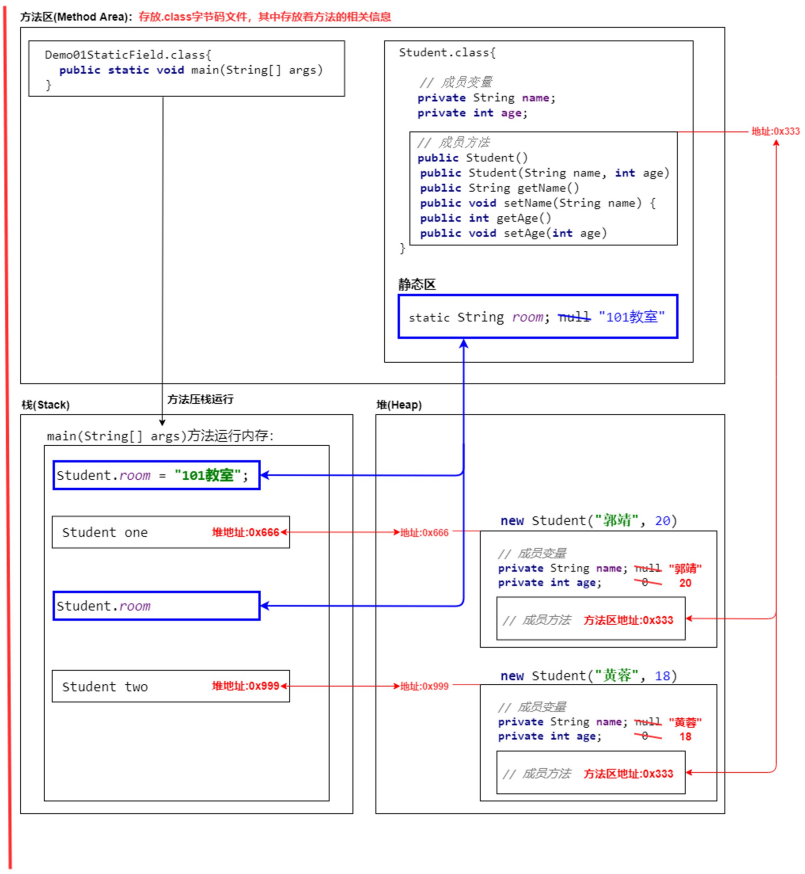
b.如果按照英文句点"."进行切分，必须写成 "\\."

二.static关键字

1.static关键字修饰成员变量

如果一个成员变量使用了static关键字修饰，那么该变量不再属于对象，而是属于所在的类，称为"类的静态成员变量"。静态成员变量跟对象没有关系，因此在使用时无需创建对象，使用类名即可调用。既可以在本类的所有成员方法中使用，也可以在其它类中使用。

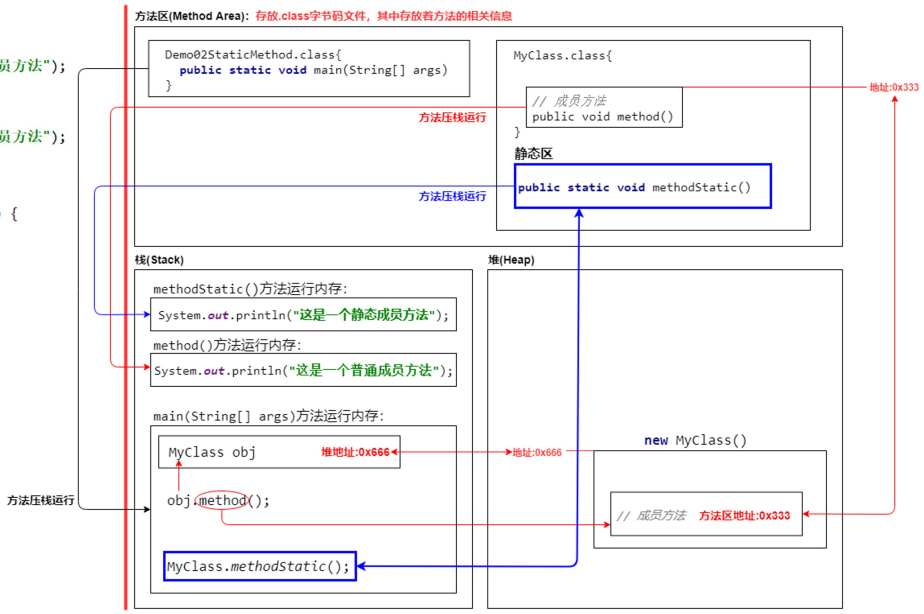
```java
public class Student {

    private String name; // 姓名
    private int age; // 年龄

    // 静态成员变量
    static String room; // 教室

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
public class Demo01StaticField {
    public static void main(String[] args) {

        // 使用类名调用静态成员变量
        Student.room = "101教室";

        Student one = new Student( name: "郭靖",  age: 20);
        System.out.println("姓名：" + one.getName());
        System.out.println("年龄：" + one.getAge());
        System.out.println("教室：" + Student.room);

        Student two = new Student( name: "黄蓉",  age: 18);
        System.out.println("姓名：" + two.getName());
        System.out.println("年龄：" + two.getAge());
        System.out.println("教室：" + Student.room);
    }
}
```



2.static关键字修饰成员方法

如果一个成员方法使用了static关键字修饰，那么该方法不再属于对象，而是属于所在的类，称为"类的静态成员方法"。静态成员方法跟对象没有关系，因此在使用时无需创建对象，使用类名即可调用。既可以在本类的所有成员方法中使用，也可以在其它类中使用。

```
public class MyClass {

    public void method(){
        System.out.println("这是一个普通成员方法");
    }

    public static void methodStatic(){
        System.out.println("这是一个静态成员方法");
    }
}
public class Demo02StaticMethod {
    public static void main(String[] args) {

        // 普通成员方法需要创建对象才能调用
        MyClass obj = new MyClass();
        obj.method();

        // 静态成员方法直接使用类名即可调用
        MyClass.methodStatic();

    }
}
```

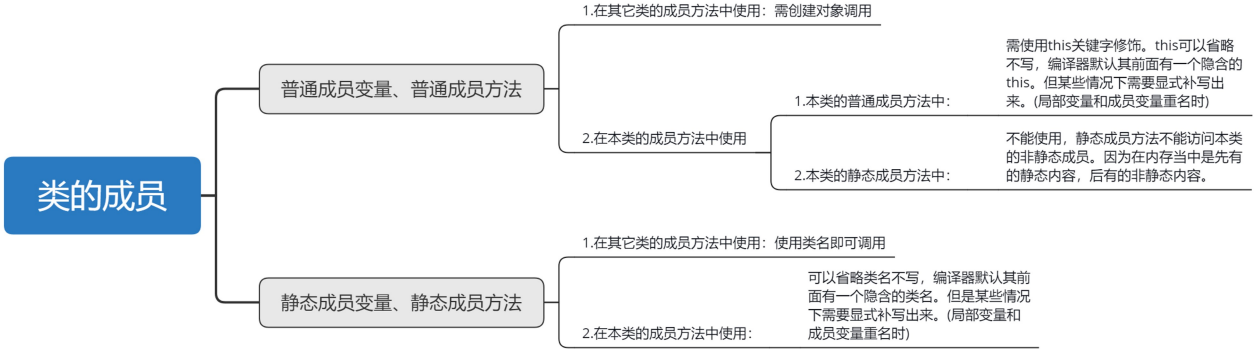


#.注意：

a.静态成员变量也可以使用private关键字进行封装。

```
public class Student {

    // 使用private关键字封装静态成员变量，使其变为"私有的静态成员变量"，不能直接访问，只能通过get、set方法来间接访问
    private static String room;

    // 相应的get、set方法也为静态成员方法
    public static String getRoom() {
        return room;
    }

    public static void setRoom(String room) {
        Student.room = room; // 局部变量和静态成员变量重名，需将类名显式补写出来
    }
}
public class PrivateStaticField {
    public static void main(String[] args) {

        // Student.room = "101教室";  不能直接访问

        // 只能通过"静态的get、set方法"间接访问
        Student.setRoom("101教室");
        System.out.println(Student.getRoom());

    }
}
```

b.成员变量、成员方法的使用规则：



c.静态成员方法中不能使用this关键字。

3.静态代码块

定义在成员位置，使用static关键字修饰的代码块 { … } 叫做静态代码块，语法格式如下：

```
public class 类名称 {

    static {
        静态代码块内容
    }

}
```

- 第一次用到本类时，静态代码块执行唯一的一次（一般是使用构造方法创建该类对象时）
- 静态内容总是优先于非静态，所以静态代码块比构造方法先执行
- 静态代码块的典型用途：用来一次性地对静态成员变量赋初值

static 关键字，可以修饰成员变量、成员方法和代码块。在使用过程中，其主要目的还是想在不创建对象的情况下，直接使用类名调用静态方法。下面将介绍两个工具类Arrays、Math，其中提供了大量的静态方法，使用起来非常方便。

三.Arrays类

java.util.Arrays 类是一个与数组相关的工具类，里面提供了大量静态方法，用来实现数组常见的操作。

- public static String toString(数组)：将参数数组变成字符串格式："[元素1，元素2，元素3]"
- public static void sort(数组)：对参数数组元素进行升序排序

四.Math类

java.lang.Math 类是数学相关的工具类，里面提供了大量的静态方法，用来完成与数学运算相关的操作。

- public static double abs(double num)：获取绝对值，有多种重载形式
- public static double ceil(double num)：向上取整
- public static double floor(double num)：向下取整
- public static long round(double num)：四舍五入
- Math.PI是Math类的静态常量，为近似的圆周率(double类型)