day01

1.文档注释

文档注释是功能级注释，用来说明一个类，一个方法或一个常量的，因此只在上述三个地方使用。

文档注释可以使用java自带的命令javadoc来对这个类生成手册。

```
package api;

/**
 * 文档注释是功能级注释，用来说明一个类，一个方法或一个常量的，因此只在上述三个地方使用。
 * 文档注释可以使用java自带的命令javadoc来对这个类生成手册。
 *
 * 在类上使用时用来说明当前类的整体功能。
 * @author FAN
 */
public class dd.Demo {
    /**
     * sayHello中使用的问候语
     */
    public static final String INFO = "Hello!";

    /**
     * 为给定的用户添加一个问候语
     * @param name 指定的用户的名字
     * @return  返回了含有问候语的字符串
     */
    public String sayHello(String name){
        return "Hello!"+ name;
    }
}
```

2.String类

String用来表示一个字符串。具有以下特点:

•java.lang.String使用了final修饰，不能 被继承;

•字符串底层封装了字符数组及针对字符数组的操作算法;

•字符串一旦创建，对象永远无法改变，但字符串引用可以重新赋值;

•Java字符串在内存中采用Unicode编码方式，任何一个字符对应两个字节的定长编码。

字符串常量池

java在堆内存中开辟了一段空间用于缓存所有使用字面量形式创建的字符串对象，并在后期再次使用该字面量创建字符串时重用对象，避免内存中堆积大量内容一样的字符串对象来减小内存开销。

对于重复出现的字符串直接量，JVM会首先在常量池中查找，如果存在即返回该对象地址。

```
package string;
/**
 *  字符串String
 *  内部使用一个char数组保存所有字符，每个字符为2字节，存的是该字符unicode编码。
 *  字符串是不变对象，一旦创建内容不可改变，若改变会创建新对象
 */
public class StringDemo {
    public static void main(String[] args) {
        /*
            字符串常量池
            java在堆内存中开辟了一段空间用于缓存所有使用字面量形式创建的字符串对象，
            并在后期再次使用该字面量创建字符串时重用对象，避免内存中堆积大量内容一样
            的字符串对象来减小内存开销。
         */
        String s1 = "123abc";//字面量
        String s2 = "123abc";//与s1字面量相同，重用对象
        //地址相同，说明s2重用了s1对象
        System.out.println(s1==s2);//true
        String s3 = "123abc";
        System.out.println(s1==s3);//true

        String s4 = new String("123abc");//new会产生新对象
        System.out.println("s4:"+s4);
        System.out.println(s1==s4);//false
        /*
            通常我们判断字符串都是比较内容，因此应当使用字符串的equals方法
         */
        System.out.println(s1.equals(s4));//true
        /*
            由于字符串是不变对象，改变内容会产生新对象
         */
        s1 = s1 + "!";//生成一个新的字符串对象123abc!.
        System.out.println("s1:"+s1);//123abc!
        System.out.println("s2:"+s2);//123abc
        System.out.println(s1==s2);//false s1,s2已经不再指向同一个对象了

        /*
            这里触发了一个编译器的特性:
            编译器在编译期间若遇到几个计算表达式，发现在编译期可以确定结果时就会进行计算
            并将结果编译到class文件中，这样以来JVM每次执行字节码文件就无需再计算了。
            下面的代码会被编译器改为:
            String s5 = "123abc";
            也因此s5会重用常量池中的对象，所以地址与s2相同
         */
        String s5 = "123" + "abc";
        System.out.println("s5:"+s5);
        System.out.println(s2==s5);
```

```
        String s = "123";
        String s6 = s + "abc";
        System.out.println("s6:"+s6);
        System.out.println(s2==s6);

        String s7 = 1+2+3+"abc";//6abc
        System.out.println(s2==s7);//false


        String s8 = 1+'2'+3+"abc";
        System.out.println(s2==s8);//false

        String s9 = 1+"2"+3+"abc";
        System.out.println(s2==s9);//true

    }
}
```

字符串常用方法

int length()

返回当前字符串的长度(字符个数)

```
package string;

public class LengthDemo {
    public static void main(String[] args) {
        String str = "我爱java!";
        int len = str.length();
        System.out.println("len:"+len);
    }
}
```

indexOf()

检索给定字符串在当前字符串中的位置，若当前字符串不含有给定内容则返回值为-1

```
package string;

public class IndexOfDemo {
    public static void main(String[] args) {
        //            0123456789012345
        String str = "thinking in java";

        int index = str.indexOf("in");//2
        System.out.println(index);

        //重载的方法可以从指定位置开始检索第一次出现给定字符串的位置
        index = str.indexOf("in",3);//5
        System.out.println(index);

        //检索最后一次出现in的位置
        index = str.lastIndexOf("in");
        System.out.println(index);
```

```
    }
}
```

substring()

截取当前字符串中指定范围内的字符串。两个参数分别为开始位置的下标和结束位置的下标。

```
package string;

/**
 * String substring(int start,int end)
 * 截取当前字符串中指定范围内的字符串。两个参数分别为开始位置的下标和结束位置的下标。
 * 注:在JAVA API中通常使用两个数字表示范围时是"含头不含尾"的。
 */
public class SubstringDemo {
    public static void main(String[] args) {
        //             01234567890
        String line = "www.tedu.cn";
        //截取域名tedu
        String str = line.substring(4,8);
        System.out.println(str);

        //重载的方法是从指定位置开始截取到字符串末尾
        str = line.substring(4);
        System.out.println(str);

    }
}
```

trim()

去除一个字符串两边的空白字符

```
package string;

/**
 * String trim()
 * 去除一个字符串两边的空白字符
 */
public class TrimDemo {
    public static void main(String[] args) {
        String line = "   hello         ";
        System.out.println(line);

        String trim = line.trim();
        System.out.println(trim);
    }
}
```

charAt()

返回当前字符串中指定位置上的字符

```
package string;

/**
 * char charAt(int index)
 * 返回当前字符串中指定位置上的字符
 */
public class CharAtDemo {
    public static void main(String[] args) {
        //            0123456789012345
        String str = "thinking in java";
        //获取第10个字符
        char c = str.charAt(9);
        System.out.println(c);

    }
}
```

startsWith()和endsWith()

判断当前字符串是否是以给定的字符串开始或结束的。

```
package string;

/**
 * boolean startsWith(String str)
 * boolean endsWith(String str)
 * 判断当前字符串是否是以给定的字符串开始或结束的。
 */
public class StartsWithDemo {
    public static void main(String[] args) {
        String line = "http://www.tedu.com";

        boolean starts = line.startsWith("http");
        System.out.println("starts:"+starts);

        boolean ends = line.endsWith(".com");
        System.out.println("ends:"+ends);
    }
}
```

toLowerCase()和toUpperCase()

```
package string;

/**
 * String toUpperCase()
 * String toLowerCase()
 * 将当前字符串中的英文部分转换为全大写或全小写
 */
public class ToUpperCaseDemo {
    public static void main(String[] args) {
```

```
        String line = "我爱Java";

        String lower = line.toLowerCase();
        System.out.println(lower);

        String upper = line.toUpperCase();
        System.out.println(upper);

    }
}
```

valueOf()

String提供了一组重载的静态方法:valueOf,作用是将其他类型转换为String

```
package string;

/**
 * String提供了一组重载的静态方法:valueOf
 * 作用是将其他类型转换为String
 */
public class ValueOfDemo {
    public static void main(String[] args) {
        int a = 123;
        String s1 = String.valueOf(a);
        System.out.println("s1:"+s1);

        double d = 123.123;
        String s2 = String.valueOf(d);
        System.out.println("s2:"+s2);

        String s3 = a+"";//任何内容和字符串链接结果都是字符串
        System.out.println("s3:"+s3);

    }
}
```