day12 常用类 Object类String类

- 学习目标
  - Object类的特点
  - Object类中的方法
    - toString()方法
    - equals()方法
  - String类的使用频率
  - String类的特点
  - String类的构造方法
  - String类的其它的方法

1. API的概念

应用程序编程接口 : 每一个技术,官方都会定义出许多的功能,开发人员可以直接拿来使用(拿来主义). API可以理解为Sun公司已经开发好的类和方法. API文档就是我们开发的帮手

2. Object类

Object类 : 所有类的父类,一切类都直接或者是间接继承Object. Object类中的所有功能,子类都可以使用.福利 :

```
class XX {} //自己的构造方法,继承下来11个方法
```

Object类定义在了java.lang包, lang包是核心包,此包中的任何类,在使用的时候不需要import 导入

2.1 Object类的本地方法

本地方法 : 方法的定义上使用关键字,是修饰符 `native` ,这个方法就是本地方法.

本地方法的特点 :

- 方法没有方法体
- 方法是C++语言编写的,Sun公司不开源
- 方法运行的时候,是一个独立的内存 (本地方法栈)
- 作用 : 凡是遇到本地方法,方法的作用是和本机的操作系统交互的

2.2 Object类的方法toString()

自己定义类Person类,默认的继承Object,Object类定义定义了方法 :

```
public String toString(); //结果是字符串,就是对象内地地址
```

> 输出语句中System.out.println(对象) 调用对象的toString()
>
> System.out.println(对象) == System.out.println(对象.toString())

toString方法的结果,和开发没有任何的关系. 我们需要的是重写父类的方法toStirng(),建立我们对象自己的字符串表现形式

2.2.1 重写父类的方法toString()

```
public class Person {
    private String name;
    private int age;
    public Person(){}
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    /**
     * 重写了父类的方法toString()
     * 返回字符串
     * @return
     * 重写方法目标 : 方法中,返回类中成员变量的值
     */
    public String toString(){
        return name + "::" + age;
    }
}
```

2.3 Object类的方法equals()

Java技术认为任何对象都具备比较性,Object类定义了方法equals(),作用就是用来比较对象的.方法结果是boolean值,对象相等就是true

自己进行了对象的比较

```
boolean b = p1.equals(p2); // false 对象不相等
```

- Object类的方法源码equals

```
public boolean equals(Object obj){
    return this == obj ;
}
```

- 引用数据类型 == 就是比较对象的地址是不是相同的
- Object类的方法 equals默认比较对象的内存地址

> 思考 : 对象的地址有可比性吗 北京海淀区中关村南大街1号, 北京昌平区北大产业园路2号. 不能比较对象的地址,**我们要重写equals方法,建立我们对象Person自己的比较形式**

```
public class Person {
    private String name;
    private int age;
    public Person(){}

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    /**
```

```
     * 重写equals,建立Person对象自己的比较方式
     * 比较对象的age年龄,年龄相同返回true
     */

    public boolean equals(Object obj){
        //健壮性判断,如果obj对象的值是null,比较的另一个对象不存在
        if (obj == null){
            return false;
        }
        //判断this和参数obj是不是一个对象 "p1" 如果是,返回true
        //怎么确定this和obj是不是一个对象,内地地址要是一样
        if ( this == obj ) {
            return true;
        }
        //比较对象的年龄,this和obj
        //obj向下转型为Person,安全性判断
        if (obj instanceof  Person) {//obj是Person对象
            Person p = (Person) obj;
            return this.age == p.age;
        }
        //不是Person,没有可比性
        return false;
    }
}
```

3. String字符串类

字符串对象,程序中定义""都是字符串对象,这个对象的使用频率最高.

字符串类 java.lang.String类,继承Object类,实现了三个接口.

程序中只要你写 "里面任意" 都是String类的对象.

字符串对象是常量,一旦创建不能修改.

3.1 字符串对象创建

```
public static void main(String[] args) {
    //字符串创建,2个方式
    //直接=创建
    String s = "abc";
    //使用构造方法创建
    String str = new String("aa");
}
```

- 直接 = 创建方式,代码少,书写简单,推荐使用
- new String() 使用了构造方法的创建形式,代码大,不推荐使用

3.2 字符串的实现原理

字符串这个数据类,在Java中是不存在的,字符串的实现原理是用char[]数组表示.

"abc",使用数组char[] ch = {'a','b','c'} ;来表示.

JDK9版本之后,节约内存,char数组改变为了byte数组

JDK8版本以前都是char数组

```
private final char value[]; //JDK中String类的源码
```

> 数组的前面的修饰符final, 最终的数组,数组一旦建立,数组的地址就被锁死(常量)使用常量的原因,为了线程安全

3.3 字符串创建对象的区别

```
String str = "abc";
String str = new String("abc");
```

```
public class StringTest {
    public static void main(String[] args) {
        String s1 = "abc";
        String s2 = new String("abc");
        System.out.println(s1 == s2); //false

        System.out.println("==========");
        /**
         *  s3 = hello 内存中出现String对象,里面是char数组
         *  s3保存的是String对象
         *
         *  s4 = "hello" 和s3中的字符串在内存中的数组表现是一样的
         *  共用
         *  s3的内存地址,赋值给s4
         */
        String s3 = "hello";
        String s4 = "hello";
        System.out.println(s3 == s4); //true

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

        String s5 = "how";
        String s6 = "you";

        String s7 = "howyou";
        /**
         *   s7 == (s5+s6)   s5和s6是变量
         *   变量在编译的时候,javac不确定变量的计算结果是什么
         *   运行的时候,JVM会为 s5+s6的结果,新开内存空间
         */
        System.out.println(s7 == (s5+s6)); //false

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

        /**
         *  "how"+"you" 是常量,值在编译期间就已经确定
         *  运行,不会建立新的内存空间
         */
        System.out.println(s7 == ("how"+"you"));// true

        String s8 = "a"+"b"+"c";

```

```
    }

    public static void print(){
        //字符串的不变
        //abc内存是不会改变
        String s = "abc";
        System.out.println(s);
        //变量s,指向了新的字符串对象
        s = "bbc";
        System.out.println(s);
    }
}

```

3.4 String类的构造方法

讲过字符编码, ASCII, 小写字母a的值97

- `String(byte[] b)` 字节数组转成字符串,使用平台的默认字符集
- `String(byte[] b,int off,int len)` 字节数组转成字符串,使用平台的默认字符集,参数off数组的开始索引,len要转的个数
- `String(byte[] b,int off,int,len,String,charsetName)` 字节数组转成字符串,使用平台的默认字符集,参数off数组的开始索引,len要转的个数,charsetName参数是你自己可以指定编码表

```
public class StringTest {
    public static void main(String[] args) throws UnsupportedEncodingException {
        stringConsByte3();
    }

    //String类构造方法相关,和字节,汉字相关
    public static void stringConsByte3() throws UnsupportedEncodingException {
        // String(byte[] bytes)通过使用平台的默认字符集解码指定的 byte 数组，构造一个新的 String。
        //平台是操作系统, 默认字符集是GBK
        //强制指定为GBK编码
        byte[] bytes ={-28, -67, -96, -27};
        String str = new String(bytes,"gbk");
        System.out.println(str);
    }

    //String类构造方法相关,和字节,汉字相关
    public static void stringConsByte2(){
        // String(byte[] bytes)通过使用平台的默认字符集解码指定的 byte 数组，构造一个新的 String。
        //平台是操作系统, 默认字符集是GBK
        //IDEA 启动的时候,为JVM添加启动参数,默认字符集改成UTF-8

        byte[] bytes ={-28, -67, -96, -27, -91, -67}; // 6字节的数组,转为字符串后是2个汉字
        String str = new String(bytes);
        System.out.println(str);
    }

```

```
26
27      //String类构造方法相关,和字节
28      public static void stringConsByte(){
29          // String(byte[] bytes)通过使用平台的默认字符集解码指定的 byte 数组，构造一个新的 String。
30          //平台是操作系统，默认字符集是GBK
31          byte[] bytes = {97,98,99,100};
32          String str = new String(bytes);
33          System.out.println(str);
34
35          //数组的一部分转成字符串
36          String str1 = new String(bytes,1,2);//从1索引开始,要2个
37          System.out.println(str1);
38      }
39  }
```

- `String(char[] b)` 字节数组转成字符串
- `String(char[] b,int off,int,len)` 字节数组转成字符串,参数off数组的开始索引,len要转的个数

```
1   //String类的构造方法,new String(char[])
2   public static void stringConsChar(){
3       char[] ch = {'a','b','c','d','e'};
4       //构造方法,数组转成字符串
5       String s = new String(ch);
6       System.out.println(s);
7
8       //构造方法,数组转成字符串,转一部分
9       String s1 = new String(ch,1,3);//从1索引开始,转3个
10      System.out.println(s1);
11  }
```

3.5 String类的常用方法

> String类的判断类型的方法, 返回都是布尔类型

- boolean equals(Object obj) 字符串之间的比较,两个字符串相同,返回true
- boolean equalsIgnoreCase(String str) 字符串之间的比较,两个字符串相同,返回true,忽略大小写
- boolean startWith(String str)判断字符串是否以另一个字符串开头,是开头就返回true
- boolean endsWith(String str)判断字符串是否以另一个字符串结尾,是结尾就返回true
- boolean contains(String str) 判断字符串中是否包含另一个字符串,完全包含返回true
- boolean isEmpty()判断字符串的长度是不是0,如果是0返回true

```
1   public static void  stringMethod(){
2           //boolean equals(Object obj) 字符串之间的比较,两个字符串相同,返回true
3           //String类继承Object,重写父类方法,比较的是字符串的实际内容
4           String s1 = new String("abc");
5           String s2 = new String("abc");
6           boolean b = s1.equals(s2);
7           System.out.println(b);
8
9           System.out.println("====================");
```

```
        //boolean equalsIgnoreCase(String str ) 字符串之间的比较,两个字符串相同,返回true,忽略大小写
        b = "abcdef".equalsIgnoreCase("ABCDEF");
        System.out.println(b);
        System.out.println("==============");
        //boolean startWith(String str)判断字符串是否以另一个字符串开头,是开头就返回true
        b = "HelloWorld.Java".startsWith("Hello");

        System.out.println(b);
        b = "HelloWorld.Java".endsWith(".Java");
        System.out.println("==============");
        System.out.println(b);

        //boolean contains(String str) 判断字符串中是否包含另一个字符串,完全包含返回true
       b = "how are you".contains("are  ");
        System.out.println("==============");
        System.out.println(b);

        //boolean isEmpty()判断字符串的长度是不是0,如果是0返回true
        b = "".isEmpty();
        System.out.println("==============");
        System.out.println(b);
    }
```

> String类的获取方法,返回值不一定

- int length() 返回字符串长度,字符串中字符的个数
- char charAt(int index) 返回指定索引上的单个字符
- int indexOf(String str) 返回指定的字符串,在当前字符串中第一次出现的索引
- int lastIndexOf(String str) 返回指定的字符串,在当前字符串中最后一次出现的索引
- String substring(int start,int end)截取字符串,参数表示开始索引和结束索引,包含开头索引,不包含结束索引

```
  /**
     * String类的获取方法
     */
    public static void stringMethod2(){
        //int length() 返回字符串长度,字符串中字符的个数
        int length = "abcdef".length();
        System.out.println("length = " + length);

        //char charAt(int index) 返回指定索引上的单个字符
        char ch = "abcdef".charAt(3);
        System.out.println("ch = " + ch);

        //int indexOf(String str) 返回指定的字符串,在当前字符串中第一次出现的索引
        //找不到指定的字符串,返回-1  负数不能作为索引出现
        int index = "how do you do".indexOf(" ");
        System.out.println("index = " + index);

        //int lastIndexOf(String str) 返回指定的字符串,在当前字符串中最后一次出现的索引
        index = "how do you do".lastIndexOf(" ");
```

```
        System.out.println("index = " + index);

        //String substring(int start,int end)截取字符串,参数表示开始索引和结束索引,
        // 包含开头索引,不包含结束索引
        String str = "HelloWorld";
        str =  str.substring(2,6);// 返回新的字符串
        System.out.println("str = " + str);

        //substring具有重载写法
        String str1 = "你好我好大家好";
        str1 = str1.substring(3);//从3索引开始,截取到最后
        System.out.println("str1 = " + str1);
    }
```

> String类的转换方法

- String toLowerCase() 字符串中的所有内容转成小写
- String toUpperCase() 字符串中的所有内容转成大写
- char[] toCharArray() 字符串转成字符数组
- byte[] getBytes() 字符串转成字节数组 (查询编码表),平台默认字符集
- byte[] getBytes(String charsetName) 字符串转成字节数组 (查询编码表),指定编码表
- static String valueOf(任意类型参数) 参数转成字符串对象

```
   /**
     * String类的转换方法
     */
    public static void stringMethod3() throws UnsupportedEncodingException {
        //- String toLowerCase() 字符串中的所有内容转成小写
        // - String toUpperCase() 字符串中的所有内容转成大写
        String str = "abCDefGHJktyM";
        String lower = str.toLowerCase();
        String upper = str.toUpperCase();
        System.out.println("lower = " + lower);
        System.out.println("upper = " + upper);

        //char[] toCharArray() 字符串转成字符数组
        char[] ch = str.toCharArray();
        System.out.println(ch);

        //byte[]  getBytes() 字符串转成字节数组 (查询编码表),平台默认字符集
        String s = "呵呵你好";
        byte[] bytes = s.getBytes("gbk");
        for (int i = 0; i < bytes.length; i++) {
            System.out.println(bytes[i]);
        }

       // static String valueOf(任意类型参数) 参数转成字符串对象
        int i = 1;
        String strI = String.valueOf(i);
        System.out.println(strI+1);
    }
```

> String类的比较方法

int compareTo(String str) 字符串之间的比较,谁大谁小,按照字典顺序(自然顺序)

```
 /**
     * String类的字符串的比较方法,字典顺序
     */
    public static void stringMethod4() throws UnsupportedEncodingException {
        //int compareTo(String str) 字符串之间的比较,谁大谁小,按照字典顺序(自然顺序)
        String str1 = "bcm";
        String str2 = "baz";
        //对象str1调用方法compareTo,参数传递str2
        /**
         *  返回值是int
         *  返回的是 负数,调用者小
         *  返回的是 正数,调用者大
         *  返回是0  一样大
         */
        int i = str1.compareTo(str2);
        System.out.println("i = " + i);
    }
```

> String类的方法 去空格,替换,切割

- String trim() 去掉字符串两边空格,中间空格不去掉
- String replace(String oldString,String newString)替换字符串
- String[] split("规则字符串") 对字符串进行切割

```
    /**
     * String类的方法 去空格,替换,切割
     */
    public static void stringMethod5() throws UnsupportedEncodingException {
        //String trim() 去掉字符串两边空格,中间空格不去掉
        String str = "  abc  def       ";
        System.out.println(str);
        str = str.trim();
        System.out.println("str = " + str);

        //String[] split("规则字符串") 对字符串进行切割
        String splitStr = "aa,bb,cc,dd,ee"; // 逗号,进行切割
        String[] strs = splitStr.split(",");
        for (int i = 0; i < strs.length; i++) {
            System.out.println(strs[i]);
        }

        //String replace(String oldString,String newString)替换字符串
        String repStr = "how do you do";
        repStr = repStr.replace("o","N");
        System.out.println("repStr = " + repStr);
    }
```

> String类正则表达式相关的功能

正则表达式 : 专门用于处理字符串的技术 (正则大神)

- 字符类 :

    - `[abc]` 字符串的这个位置只能是abc
    - `[^abc]` 字符串的这个位置不能是abc
    - `[a-zA-Z]` 字符串的这个位置必须是字母,52个
    - `[^a-zA-Z]` 字符串的这个位置必须不能是字母,52个
- 数字类:
    - `[0-9]` 字符串的这个位置只能是数字
    - `[^0-9]` 字符串的这个位置不能是数字
    - `[\d]` 等同于 `[0-9]`
    - `[\D]` 等同于 `[^0-9]`
- 预定义字符 :
    - . 匹配所有的字符
    - `[\d]` 等同于 `[0-9]`
    - `[\D]` 等同于 `[^0-9]`
    - `[\w]` 文字字符,包含数字,字母,下划线 `[a-zA-Z0-9_]`
    - `[\W]` 文字字符,不能包含数字,字母,下划线 `[^a-zA-Z0-9_]`
- 数量词 :
    - X{m} X这个字符只能出现m次 a{3}
    - X{m,} X这个字符至少出现m次
    - X{m,n} X这个字符至少出现m次,不超过n次
    - X? X这个字符出现一次,或者一次也没有
    - X* X这个字符出现零次或者多次
    - X+ X这个字符出现至少一次

> 正则表达式的匹配功能,String类的方法matches()

```
/**
     *  检查邮箱
     *  规则 :
     * @ 前面 : 可以是数组,字母,混合,_  位数放下
     * @ 后面 : 数组,字母  sina qq 126 1393 yahoo gmail 位数放下
     * . 固定 : com  cn org  edu gov 字母  位数放下
     */
public static void stringMethod2(){
    String email = "shihehe@sina.com";
    String reg = "[\\w]+@[a-z0-9]+(\\.[a-z]+)+";
    boolean b = email.matches(reg);
    System.out.println(b);

}
/**
     *  正则表达式检查手机号是否合法
     *  开头必须是1,长度固定11
     *  第二位3 4 5 6 7 8 9
     *  第三位 必须是都是数字
     */
public static void stringMethod(){
    String tel = "13800138000";
    //定义正则的规则,也是字符串
    String regex = "1[3459678][0-9]{9}";
    //正则规则,和字符串校验
    //String类的方法 matches()
```

```
    boolean b = tel.matches(regex);
    System.out.println(b);
}
```

> String类的方法split

```
   public static void stringMethod3(){
        String str = "as123d387654w5465fasfr234567sa";
        String[] strings = str.split("\\d+");
        for (int i = 0; i < strings.length; i++) {
            System.out.println(strings[i]);
        }
        System.out.println("================");
        String ip = "192.....168.....35.121";
        String[] ipArray = ip.split("\\.+");
        for (int i = 0; i < ipArray.length; i++) {
            System.out.println(ipArray[i]);
        }
    }
```

> String类的方法replaceAll

```
public static void stringMethod4(){
       String str = "as123d387654w5465fasfr234567sa";
       //字符串中的所有数组,换成#
       String repString = str.replaceAll("\\d+","#");
       System.out.println(repString);

       String first =  str.replaceFirst("\\d+","#");
       System.out.println(first);
   }
```

4. StringBuilder

StringBuilder是字符串对象的缓冲区对象, 缓冲区(出现目的,为了高效)提供String类的效率.

```
String str = "a"; //字符数组
String str2 = "b";//字符数组
String str3 = str + str2;//字符数组
```

4.1 StringBuilder类的实现原理

一个可变的字符序列,字符序列就是字符数组

```
String 类中 : private final char[] value;
StringBuilder : char[] value;
```

字符序列是数组,Java数组的是定长的,一旦创建,长度固定!

创建对象的时候,StringBuilder中的数组的初始化长度为16个字符

StringBuilder自动的进行数组的扩容,新数组实现,原来数组的中元素复制到新的数组.

> 结论 : 无论怎么做字符串的操作,StringBuilder内部永远只有一个数组

StringBuilder类是线程不安全的类,运行速度快 , 推荐使用StringBuilder

StringBuffer是线程安全的类,运行速度慢,多线程的程序,使用

两个类的构造方法,和其他的方法,一模一样.