目录

[Day08.    Java](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896688)

[1     回顾](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896689)

[2     Object](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896690)

[练习1  Object](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896691)

[3     String](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896692)

[3.1       字符串创建](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896693)

[3.2       字符串的常量池](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896694)

[练习2   字符串](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896695)

[3.3       字符串不可变](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896696)

[练习3   字符串连接效率](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896697)

[3.4       字符串的常用方法](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896698)

[练习4   二进制十六进制格式判断](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896699)

[4     StringBuilder](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896700)

[练习5  StringBuilder](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896701)

[4.1   StringBuilder 的方法](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896702)

[4.2   StringBuilder 和 StringBuffer](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896703)

[5     正则表达式 Regex](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896704)

[练习6   正则表达式](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896705)

[6     基本类型的包装类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896706)

[6.1       数字父类 Number](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896707)

[6.2   Integer 类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896708)

[练习7   基本类型的包装类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896709)

[6.3   Integer 的方法](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896710)

[6.4       自动装箱、自动拆箱](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896711)

[6.4.1   自动装箱](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896712)

[6.4.2   自动拆箱](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896713)

[练习8   双色球](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896714)

[7     作业](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day08.htm#_Toc5896715)

**Day08.  Java**

**1   回顾**

  类，模板、图纸

  对象，实例

  引用，遥控器

  局部变量，栈

  成员变量，堆

  构造方法

  不定义，有默认

  作用：常用来为成员变量赋值

  this

  this.xxxx

  引用当前对象的地址

  this(...)

  构造方法之间调用

  首行代码

  方法重载 Overload

  同名不同参

a()

a(int)

a(double)

a(String)

  继承

  作用：代码重用、复用

  单继承

  如果不继承其他类，默认继承 Object

  子类对象

  由父类对象和子类对象绑定，整体作为一个对象

  调用成员时，先找子类，再找父类

  父类的构造方法

  默认执行无参构造 super()

  手动调用有参构造 super(参数)

  方法重写 Override

  继承的方法，重新定义重新编写

  理解成覆盖、改写

  super

  super.xxx()

  重写时，调用父类同一个方法

  super(...)

  调用父类构造方法

**2   Object**

所有类的顶层父类，一个类如果不继承其他类，默认从 Object 继承

class A /\*extends Object\*/ {

}

Object 的方法

  **toString()**

获得一个对象的字符串表示

Object中的默认实现是：

"day0801.Point@a19c832c"

"类名@地址"

如果需要，可以在子类中重写 toString()

  **equals(Object obj)**

当前对象，与参数对象 obj 比较是否相等

a.equals(b)

Object中的默认实现是比较内存地址

this == obj

   如果需要，在子类中，可以重写这个方法

**练习1  Object**

项目：day0801\_Object

类：day0801.Test1

             Point

Point

**package** day0801;

**public** **class** Point /\*extends Object\*/ {

**int** x;

**int** y;

//无参构造、全参构造

**public** Point() {

}

**public** Point(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

@Override

**public** String toString() {

**return** "Point [x=" + x + ", y=" + y + "]";

}

/\*

 \* a = {x:3, y:4}

 \* b = {x:3, y:4}

 \*

 \* a.equals(b)

 \*/

@Override

**public** **boolean** equals(Object obj) {

    Point p = (Point) obj;

    //当前对象的x,y，和参数对象p的x,y比较

**return** **this**.x == p.x && **this**.y == p.y;

}

}

Test1

**package** day0801;

**public** **class** Test1 {

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

    Point p1 = **new** Point(3, 4);

    Point p2 = **new** Point(3, 4);

    System.***out***.println(p1.toString());

    System.***out***.println(p2.toString());

    System.***out***.println(p1 == p2);

    /\*

     \* equals() 方法比较什么，

     \* 要进入方法，看方法代码比较的是什么

     \* Object中比较内存地址

     \*/

    System.***out***.println(p1.equals(p2));

}

}

**3   String**

String是封装 char[] 数组的对象

s = "abcd"

s = {

    -value: ['a', 'b', 'c', 'd']

}

**3.1     字符串创建**

char[] a = {'a', 'b', 'c', 'd'};

String s = new String(a);

上面创建方式的简化语法

String s = "abcd";

**3.2     字符串的常量池**

第一次使用一个字符串字面值时，会在“字符串常量池”中新分配内存，再次使用相同字面值时，直接访问常量池中存在的对象，而不重复创建

String s1 = "abcd";

String s2 = "abcd";

**练习2  字符串**

项目：day0802\_字符串

类：day0802.Test1

**package** day0802;

**public** **class** Test1 {

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

**char**[] a = {'a', 'b', 'c', 'd'};

    String s1 = **new** String(a);//新分配内存

    String s2 = "abcd";//在常量池新分配内存

    String s3 = "abcd";//访问常量池存在的对象

    System.***out***.println(s1);

    System.***out***.println(s2);

    System.***out***.println(s3);

    System.***out***.println(s1 == s2);//false，地址不同

    System.***out***.println(s2 == s3);//true，地址相同

    /\*

     \* equals()比较什么，

     \* 进入equals()方法看里面的代码在比较什么

     \*/

    System.***out***.println(s1.equals(s2));

}

}

**3.3     字符串不可变**

String s1 = "aaa"

String s2 = "bbb"

String s3 = "ccc"

String s4 = s1 + s2 + s3

字符串连接，效率低，每次连接都新建字符串对象

**练习3  字符串连接效率**

Test2

**package** day0802;

**public** **class** Test2 {

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

    String s0 = "abcdefghijklmnopqrstuvwxyz";

    String s = "";

    //连接前，取系统当前时间的时间点毫秒值

    //毫秒值：1970-1-1 0点开始的毫秒值

**long** t = System.*currentTimeMillis*();

**for** (**int** i = 0; i < 100000; i++) {

        s += s0;

    }

    //求时间差

    t = System.*currentTimeMillis*()-t;

    System.***out***.println(t);

}

}

**3.4     字符串的常用方法**

**charAt(i)**

获取指定位置的字符

**length()**

字符串长度，字符的数量

**indexOf(子串)**

找第一个子串出现的起始位置，找不到返回 -1

**indexOf(子串, start)**

从执行位置向后查找

**lastIndexOf(子串)**

从后向前找

**substring(start)**

截取 start 到末尾

**substring(start, end)**

截取 [start, end) 范围

**trim()**

去除两端的空白字符

**练习4  二进制十六进制格式判断**

**package** day0402;

**import** java.util.Scanner;

**public** **class** Test1 {

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

    //死循环

    outer:

**while**(**true**) {

        System.***out***.println(

         "输入0b开头二进制，或0x开头十六进制，q退出");

        String s = **new** Scanner(System.***in***).nextLine();

        //字符串比较是否相等，不能用 ==

        //要用 equals() 方法

**if**(s.equals("q")) {

**break**;

        }

        //前缀不是0b，也不是0x

**if** (! (s.startsWith("0b") || s.startsWith("0x"))) {

           System.***out***.println("前缀错误");

**continue**;

        }

        //字符串长度至少3

**if**(s.length() < 3) {

           System.***out***.println("格式错误");

**continue**;

        }

**if**(s.startsWith("0b")) {

           // 0b10x0111101

**for** (**int** i = 2; i < s.length(); i++) {

**char** c = s.charAt(i);

**if** (! (c=='1' || c=='0')) {

                 System.***out***.println("二进制格式错误");

**continue** outer;

              }

           }

        }

**if**(s.startsWith("0x")) {

           // 0x9af9ce

**for** (**int** i = 2; i < s.length(); i++) {

**char** c = s.charAt(i);

**if** (! ((c>='0' && c<='9') || (c>='a'&&c<='f'))) {

                 System.***out***.println("十六进制格式错误");

**continue** outer;

              }

           }

        }

        //把s传递到 f() 方法，转成10进制整数

**int** r = *f*(s);

        System.***out***.println("10进制："+r);

        System.***out***.println("\n---------------------");

    }

}

**private** **static** **int** f(String s) {

    /\*

     \* 参数字符串 s

     \* "0B1101011"

     \* "0xf9aac5e" --> "f9aac5e"

     \* 如果是大写，先变成小写

     \* s.toLowerCase()

     \*

     \* 获取0b或0x后面的数字部分

     \* s.substring(2)

     \*

     \* 判断前缀是0b还是0x

     \* s.startsWith("0b")

     \*

     \* 转成10进制

     \* Integer.parseInt("1101011", 2)

     \* Integer.parseInt("f9aac5e", 16)

     \*/

    //转成小写 "0B1101" --> "0b1101"

    s = s.toLowerCase();

    //截取0b、0x后面的数字部分 "0b1101"--> "1101"

    String n = s.substring(2);

    //准备一个保存结果的变量 r

**int** r = 0;

    //判断前缀是0b还是0x

**if** (s.startsWith("0b")) {

        r = Integer.*parseInt*(n, 2);//"1101"-->13

    } **else** **if** (s.startsWith("0x")) {

        r = Integer.*parseInt*(n, 16);//"d" -->13

    }

**return** r;

}

}

**4   StringBuilder**

可变的字符序列

封装 char[] 数组，提供了一组方法，可以对内部封装的字符进行修改

常用来代替字符串，做高效率的字符串连接

append() 方法，追加字符内容

  内部数组默认初始容量 16

  放满后，容量翻倍+2

**练习5  StringBuilder**

项目：day0803\_StringBuilder

类：day0803.Test1

**package** day0803;

**public** **class** Test1 {

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

    String s0 = "abcdefghijklmnopqrstuvwxyz";

    StringBuilder sb = **new** StringBuilder();

**long** t = System.*currentTimeMillis*();

**for** (**int** i = 0; i < 100000; i++) {

        sb.append(s0);

    }

    t = System.*currentTimeMillis*() - t;

    System.***out***.println(t);

}

}

**4.1     StringBuilder 的方法**

**append()**

追加

**delete(start, end)**

删除区间 [start, end)

**deleteCharAt(i)**

删除指定位置的单个字符

**insert(i, 内容)**

在指定位置插入

**setCharAt(i, 字符)**

替换指定位置的单个字符

**replace(start, end, 内容)**

替换指定范围的内容

...

**4.2     StringBuilder 和 StringBuffer**

StringBuffer 是一个旧版本的类，JDK1.0中就存在StringBuffer

StringBuilder 是一个新版本的类，JDK1.5中新添加的类

StringBuffer 线程安全

StringBuilder 线程不安全，效率高

**5   正则表达式 Regex**

Regex - Regular Expression

规则表达式

正确的字符串格式规则

一般用来判断用户输入的内容是否符合格式要求

例如，email，用户名，密码，生日，网址，手机号

百度“正则表达式大全”

|  |  |
| --- | --- |
| 正则表达式 | 匹配的字符串 |
| k | k |
| abc | abc |
| [abc] | a, b, c |
| [abc][123] | a1, a2, a3, b1, b2, b3, c1, c2, c3 |
| [a-z] | a, z, b, k |
| [a-zA-Z\_0-9] | a,A,9,\_ |
| [\u4e00-\u9fa5] | 中文范围 |
| [^a-zA-Z] | 排除英文  9,\*,中 |
| \d | 数字[0-9] |
| \D | 排除数字[^0-9] |
| \w | 单词字符[a-zA-Z\_0-9] |
| \W | 排除单词字符 |
| \s | 空白字符 |
| \S | 排除空白字符 |
| . | 任意字符 |
| [abc]? | ? 0或1个  a,b,,c |
| [abc]?[123] | a1,b3,a2,1,2,3 |
| [abc]\* | \* 0到多个  a,ab,,abccbbccaacbc |
| [abc]+ | + 1到多个  a,ab,abcbcbcaa |
| [abc]{3} | 3个  aaa,bbb,abc,ccb,caa |
| [abc]{3,5} | 3到5个  abc,abca,abcab |
| [abc]{3,} | 3到多个，至少3个  abc,abca,abcccbbacb |
| | |  |
| ^ | 匹配起始位置 |
| $ | 匹配结束位置 |

字符串的 matches() 方法，用来做匹配判断

if(s.matches(regex)) {

}

**split(正则)**

用匹配的子串，来拆分字符串

String s = "aaa,bbb,ccc";

String[] a = s.split(",");

**replaceAll(正则, 子串)**

替换所有匹配的子串

String s = "sdf754sdfsfd44767sdf7575";

s = s.replaceAll("\\d+", "\*\*\*");

**练习6  正则表达式**

项目：day0804\_正则表达式

类：day0804.Test1

**package** day0804;

**import** java.util.Scanner;

**public** **class** Test1 {

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

    System.***out***.println("\n-----------------");

*f1*();

    System.***out***.println("\n-----------------");

*f2*();

}

**private** **static** **void** f1() {

    System.***out***.println("身份证号：");

    String s = **new** Scanner(System.***in***).nextLine();

    /\*

     \* 123456789012345

     \* 123456789012345678

     \* 12345678901234567x

     \* 12345678901234567X

     \*

     \* |

     \* \d{15}|

     \* \d{15}|\d{17}

     \* \d{15}|\d{17}[\dxX]   正则表达式语法

     \*

     \* "\\d{15}|\\d{17}[\\dxX]"

     \*/

    String regex = "\\d{15}|\\d{17}[\\dxX]";

    //判断字符串s能否匹配指定的正则表达式

**if** (s.matches(regex)) {

        System.***out***.println("格式正确");

    }**else** {

        System.***out***.println("格式错误");

    }

}

**private** **static** **void** f2() {

    System.***out***.println("固定电话：");

    String s = **new** Scanner(System.***in***).nextLine();

    /\*

     \* 1234567

     \* 12345678

     \* 010-1234567

     \* 0101-12345678

     \* (010)1234567

     \* (0101)12345678

     \*

     \* (区号表达式)?\d{7,8}

     \* (|)?\d{7,8}

     \* (\d{3,4}-|)?\d{7,8}

     \* (\d{3,4}-|())?\d{7,8}

     \* (\d{3,4}-|\(\d{3,4}\))?\d{7,8}

     \*/

    String regex = "(\\d{3,4}-|\\(\\d{3,4}\\))?\\d{7,8}";

    //判断字符串s能否匹配指定的正则表达式

**if** (s.matches(regex)) {

        System.***out***.println("格式正确");

    }**else** {

        System.***out***.println("格式错误");

    }

}

}

**6   基本类型的包装类**

有时需要把基本类型，当做一个对象来使用

void f(Object obj) {

}

f(new Integer(6))

|  |  |
| --- | --- |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| char | Character |
| boolean | Boolean |

**6.1     数字父类 Number**

子类：

Byte,Short,Integer,Long

Float,Double

BigDecimal,BigInteger

取出基本类型值的方法：

byteValue()

shortValue()

intValue()

longValue()

floatValue()

doubleValue()

**6.2     Integer 类**

创建Integer对象

a = {

   value: 6

}

Integer a = new Integer(6);

Integer a = Integer.valueOf(6);

Integer类中，存在 256 个Integer 的缓存对象，封装 -128 到 127

[{-128}, {-127}, {} ... {6} ... {127}]

如果指定范围内的值，访问缓存对象

如果指定范围外的值，直接新建对象

**练习7  基本类型的包装类**

项目：day0805\_基本类型的包装类

类：day0805.Test1

**package** day0805;

**public** **class** Test1 {

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

    Integer a = **new** Integer(6);

    Integer b = Integer.*valueOf*(6);

    Integer c = Integer.*valueOf*(6);

    System.***out***.println(a == b);//false，地址不同

    System.***out***.println(b == c);

    System.***out***.println(a.equals(b));//

    System.***out***.println(a.byteValue());

    System.***out***.println(a.shortValue());

    System.***out***.println(a.intValue());

    System.***out***.println(a.longValue());

    System.***out***.println(a.floatValue());

    System.***out***.println(a.doubleValue());

}

}

**6.3     Integer 的方法**

字符串解析成 int

Integer.parseInt("255")  255

Integer.parseInt("11111111", 2)    255

Integer.parseInt("377", 8)   255

Integer.parseInt("ff", 16)   255

   Byte.parseByte()

   Short.parseShort()

   Integer.parseInt()

   Long.parseLong()

   Float.parseFloat()

   Double.parseDouble()

   Boolean.parseBoolean()

进制转换

Integer.toBinaryString(255)  "11111111"

Integer.toOctalString(255)   "377"

Integer.toHexString(255)     "ff"

**6.4     自动装箱、自动拆箱**

**6.4.1 自动装箱**

基本类型值，自动包装成包装对象

Integer a = 6;

编译器编译成：

Integer a = Integer.valueOf(6);

**6.4.2 自动拆箱**

int i = a;

编译器编译成：

int i = a.intValue();

a = a + 1;

编译器编译成：

a = Integer.valueOf(a.intValue() + 1);

  自动拆箱，要当心 null 值

**练习8  双色球**

红球 33 选 6

蓝球 16 选 1

项目：day0806\_双色球

类：day0806.Test1

**package** day0806;

**import** java.util.Arrays;

**import** java.util.Random;

**public** **class** Test1 {

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

    /\*

     \* r = [1,2,3,4,5,6,7,8,9,10....33]

     \* b = [1,2,3,4,5...16]

     \*/

    //准备两个号码数组

**int**[] r = *zbsz*(33);

**int**[] b = *zbsz*(16);

    System.***out***.println(Arrays.*toString*(r));

    System.***out***.println(Arrays.*toString*(b));

**int**[] red = *xuanRed*(r);//选6个红球

**int** blue = *xuanBlue*(b);//选1个蓝球

    System.***out***.println("红球：");

    System.***out***.println(Arrays.*toString*(red));

    System.***out***.println("蓝球：");

    System.***out***.println(blue);

}

**private** **static** **int**[] xuanRed(**int**[] r) {

    /\*

     \*                        j

     \* [12,15,3,1,17,10,7,8,9,6,11,4,13,14,2,16,5,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33]

     \*        i

     \*

     \* [12,15,3,1,17,10]

     \*/

    //i循环从0到<6，随机交换6个值到前面

**for** (**int** i = 0; i < 6; i++) {

        //j随机定位 i+ [0, 33-i) --> [i, 33)

**int** j = i+ **new** Random().nextInt(33-i);

        //i,j交换

**int** t = r[i];

        r[i] = r[j];

        r[j] = t;

    }

    //复制前6个位置，生成一个新的数组

**int**[] arr = Arrays.*copyOf*(r, 6);

**return** arr;

}

**private** **static** **int** xuanBlue(**int**[] b) {

**return** b[**new** Random().nextInt(16)];

}

**private** **static** **int**[] zbsz(**int** n) {

    //新建int[]数组a，长度n

**int**[] a = **new** **int**[n];

    //遍历，填入1到n

**for** (**int** i = 0; i < a.length; i++) {

        a[i] = i+1;

    }

    //返回数组a

**return** a;

}

}

**7   作业**

  重写

  day0703\_电子宠物

  字符串

  输入 email 地址，提取名字部分

abc@def.com ---> abc

  输入字符串，判断是否对称

abcdedcba