一、包装类

1、简介

> 包装类是将基本数据类型的值包装为Java中的对象，Java语言为8种基本数据类型分别提供了包装类。

2、分类

基本数据类型	包装类
boolean	Boolean
char	Character
byte	Byte（继承自Number）
short	Short（继承自Number）
int	Integer（继承自Number）
long	Long（继承自Number）
float	Float（继承自Number）
double	Double（继承自Number）

3、包装类和基本数据的转换

```
3.1、jdk5前的手动装箱和拆箱方式，装箱：基本类型->包装类型；反之，拆箱
3.2、jdk5后的自动装箱和拆箱方式是自动装箱
3.3、自动装箱底层调用的是valueOf方法，如Integer.valueOf()
3.4、自动拆箱底层调用的是xxValue方法，如integer.intValue()
```

4、Integer类和Character类常用方法

```
System.out.println(Integer.MIN_VALUE);             // 返回最小值
System.out.println(Integer.MAX_VALUE);             // 返回最大值

System.out.println(Character.isDigit('a'));        // 判断是不是数字
System.out.println(Character.isLetter('a'));       // 判断是不是字母
System.out.println(Character.isUpperCase('a'));    // 判断是不是大写
System.out.println(Character.isLowerCase('a'));    // 判断是不是小写
System.out.println(Character.isWhitespace('a'));   // 判断是不是空格
System.out.println(Character.toUpperCase('a'));    // 转成大写
System.out.println(Character.toLowerCase('a'));    // 转成小写
```

5、Integer ==运算符

Integer中有缓存数组 -128 ~ 127，也就是说，当使用 == 进行比较时，如果两边的属性值都在范围内，直接从数组中拿来进行比较。如果属性值不在范围内，就会创建新的对象，再进行比较

二、***String类

1、String类理解

1.1、String对象用于保存字符串，也就是一组字符序列
1.2、字符串常量对象使用双引号括起的字符序列。如："你好"、"12，2"、"hello"等
1.3、字符串的字符使用Unicode字符编码，一个字符（不区分字母还是汉字）占两个字节
1.4、String类常用构造方法
- String s1 = new String();
- String s2 = new String(String original);
- String s3 = new String(char[] a);
- String s4 = new String(char[] a, int startIndex, int count);

1.5、String类实现了多个接口
- Serializable：String 可以串行化，也就是可以再网络中传输
- Comparable：String 对象可以比较大小

1.6、String是final类，不能被其他的类继承
1.7、String有属性 private final char value[]; 用于存放字符串内容
1.8、vulue[] 是一个final类型，也就是String对象的地址不可修改

2、String类创建对象

方式一：直接赋值 String s = "xxx";
- 先从常量池查看是否有"xxx"数据空间，如果有，直接指向；如果没有则重新创建，然后指向。s最终指向的是常量池的空间地址。

方式二：调用构造器 String s2 = new String("xxx");
- 先在堆中创建空间，里面维护了value属性，指向常量池的xxx空间。如果常量池没有"xxx"，重新创建，如果有，直接通过value指向。最终指向的是队中的空间地址。

3、String类特性

字符串常量相加，比较时只看常量池；字符串变量相加，比较时看对象

4、String类常见方法

equals	区分大小写，判断内容是否相等
equalsIgnoreCase	忽略大小写，判断内容是否相等
length	获取字符的个数，字符串的长度
indexOf	获取字符在字符串中第1次出现的索引，索引从0开始，如果找不到，返回-1
lastIndexOf	获取字符在字符串中最后1次出现的索引，索引从0开始，如果找不到，返回-1
substring	截取指定范围的字符串
trim	去前后空格
charAt	获取其索引处的字符，注意不能使用Str[index]这种方式
toUpperCase	将字符串转换成大写
toLowerCase	将字符串转换成小写

```
    concat          拼接字符串
    compareTo       比较两个字符串的大小
    replace         替换字符串中的字符
    split           分割字符串
    toCharArray      转换成字符数组
    format          格式化字符串，%s 字符串；%c 字符；%d 整型；%.2f 浮点型
```

三、***StringBuffer和StringBuilder类

1、StringBuffer类-简介

```
    StringBuffer是一个容器，是基于String的可变长度的类。java.lang.StringBuffer代表可变的字符序列，可以对字符串内容进行增删。
```

2、String VS StringBuffer

```
    2.1、String保存的是字符串常量，里面的值不能更改，每次String类的更新实际上就是更改地址，效率较低。
    2.2、StringBuffer保存的是字符串变量，里面的值可以更改，每次StringBuffer的更新实际上可以更新内容，不用更新地址，效率较高。
```

3、StringBuffer类-构造器

```
    StringBuffer()
        构造一个其中不带字符的字符串缓冲区，其初始容量为16字符
    StringBuffer(CharSequence seq)
        public java.lang.StringBuilder(CharSequence seq) 构造一个字符串缓冲区，它包含与指定的CharSequence相同的字符
    StringBuffer(int capacity)
        构造一个不带字符，但具有初始容量的字符串缓冲区。即对char[]大小进行指定
    StringBuffer(String str)
        构造一个字符串缓冲区，并将其内容初始化为指定的字符串内容
```

4、String和StringBuffer转换

```
# String -> StringBuffer
    String str = "Hello"
    方式1：构造器
    StringBuffer stringBuffer = new StringBuffer(Str);
    方式2：append方法
    StringBuffer stringBuffer1 = new StringBuffer();
    stringBuffer1 = stringBuffer.append(str);
```

```
# StringBuffer -> String
    StringBuffer stringBuffer3 = new StringBuffer("xxx");
    方式1：StringBuffer的toString方法
    String s = stringBuffer3.toString();
    方式2：构造器
    String s1 = new String(stringBuffer3);
```

5、StringBuffer类-常见方法

```
    5.1、append()                       增加
    5.2、delete(start,end)               删除
    5.3、replace(start,end,string)      将start-end之间的内容替换掉，不含end
    5.4、indexOf()                      查找
    5.5、insert()                       插入
    5.6、length                         获取长度
```

6、StringBuilder类-简介

```
    StringBuilder 和 StringBuffer 均代表可变的字符序列，方法是一样的，所以使用和StringBuffer一样。
```

7、StringBuilder类-常见方法

```
    7.1、append()                       增加
    7.2、delete(start,end)               删除
    7.3、replace(start,end,string)      将start-end之间的内容替换掉，不含end
    7.4、indexOf()                      查找
    7.5、insert()                       插入
    7.6、length                         获取长度
```

8、StringBuilder类-细节

```
    8.1、StringBuilder 继承 AbstractStringBuilder类
    8.2、实现了 Serializable，说明StringBuilder对象是可以串行化（对象可以网络传输，可以保存到文件）
    8.3、StringBuilder 是final类，不能被继承
    8.4、StringBuilder 对象字符序列仍然是存放在其父类 AbstractStringBuilder的 char[] value;
    8.5、StringBuilder 的方法，没有做互斥的处理，即没有synchronized 关键字，因此在单线程的情况下使用StringBuilder
```

9、String、StringBuffer、StringBuilder比较

```
    9.1、StringBuilder 和 StringBuffer 非常类似，均代表可变的字符序列，而且方法一样
    9.2、String：不可变字符序列，效率低，但是复用率高
    9.3、StringBuffer：可变字符序列，效率较高(增删)，线程安全
    9.4、StringBuilder：可变字符序列，效率最高，线程不安全
    9.5、如果需要对String做大量修改，建议不要使用String
```

10、String、StringBuffer、StringBuilder选择

```
    10.1、如果字符串存在大量的修改操作，一般使用 StringBuffer 或 StringBuilder
    12.2、如果字符串存在大量的修改操作，并在单线程的情况，使用StringBuilder
    12.3、如果字符串存在大量的修改操作，并在多线程的情况，使用StringBuffer
    12.4、如果字符串很少修改，被多个对象引用，使用String，比如配置信息等
```

四、Math类

1、简介

Math类包含用于执行基本数学运算的方法，如初等指数、对数、平方根和三角函数。

2、常见方法

```
abs()       绝对值
pow()       求幂
ceil()      向上取整
floor()     向下取整
round()     四舍五入
sqrt()      求开方
random()     求随机数
max()       求两个数的最大值
min()       求两个数的最小值
```

五、Date日期类、Calendar日历类以及新的日期类

1、Date日期类-简介

Date：精确到毫秒，代表特定的瞬间
SimpleDateFormat：格式和解析日期的类

2、Date日期类-用法

```
2.1、获取当前系统时间
    new Date()
2.2、当前系统时间格式化
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 hh:mm:ss E");
    String format = sdf.format(date);
2.3、时间戳转换为系统时间
    new Date(64655434)
2.4、格式化转换为系统时间
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 hh:mm:ss E");
    sdf.parse("2022年9月1日 10:19:22 星期四")
```

3、Calendar日历类-简介

Calendar类是一个抽象类，它为特定瞬间与一组诸如 YEAR、MONTH、DAY_OF_MONTH、HOUR等日历字段之间的转换提供了一些方法，并为操作日历字段（如获取星期几）提供了一些方法。

4、Calendar日历类-用法

```
Calendar c = Calendar.getInstance();

System.out.println("年：" + c.get(Calendar.YEAR));
System.out.println("月：" + c.get(Calendar.MONTH) + 1);
System.out.println("日：" + c.get(Calendar.DAY_OF_MONTH));
System.out.println("小时：" + c.get(Calendar.HOUR));
System.out.println("分钟：" + c.get(Calendar.MINUTE));
System.out.println("秒：" + c.get(Calendar.SECOND));
```

5、Calendar不足

```
5.1、可变性：像日期和时间这样的类是不可变的
5.2、偏移性：Date中的年份是从1900开始的，而月份都从0开始
5.3、格式化：格式化只对Date有用，Calendar则不行
5.4、线程不安全；不能处理闰秒（每隔2天，多出1秒）
```

6、第三代日期类-常见方法

```
LocalDate           日期/年月日
LocalTime           时间/时分秒
LocalDateTime        日期时间/年月日时分秒
DateTimeFormatter    格式化日期类
```

7、时间戳 Instant

```
Instant -> Date
    Date date = Date.from(instant);
Date -> Instant
    Instant instant = date.toInstant();
```

六、System类

1、简介

```
    System类位于java.lang包，代表当前java程序的运行平台，系统级的很多属性和控制方法都放置在该类的内部。
```

2、常见方法

```
exit()              退出当前程序
arrcopy()           复制数组元素，比较适合底层调用，一般使用Arrays.copyOf复制数组
currentTimeMillens()     返回当前时间距离1970-1-1的毫秒数
gc()                     运行垃圾回收机制
```

七、Arrays类

1、简介

> Arrays里面包含了一系列静态方法，用于管理或操作数组（比如排序和搜索）。

2、常见方法

```
toString()       返回数组的字符串形式
    - Arrays.toString(arr)
sort()           排序
    - Arrays.sort(arr)
binarySearch()    通过二分搜索法进行查找，要求必须排好序
    - Arrays.binarySearch(arr, 3)
copyOf()         数组元素的复制
    - Arrays.copyOf(arr, arr.length)
fill()           数组元素的填充
    - Arrays.fill(num, 99)
equals()         比较两个数组元素内容是否完全一致
    - Arrays.equals(arr, arr2)
asList()         将一组值，转换成list
    - Arrays.asList(2,3,4,5,1,8)
```

八、BigInteger类和BigDecimal类

1、简介

> BigInteger用于保存比较大的整型，BigDecimal用于保存精度更高的浮点型。

2、BigInteger-常见方法

```
add          加
subtract     减
multiply     乘
divide       除
```

3、BigDecimal-常见方法

```
add          加
subtract     减
multiply     乘
divide       除
    - 使用divide时，当余数为无穷数时，在参数列表加上 xx.ROUND_CEILING即可
```