常用类

包装类

- boolean-Boolean
- char-Character
- int-Interger
- short-Short
- long-Long
- byte-Byte
- float-Float
- double-Doube

均属于number的子类

均实现了单行化接口和可比较大小的接口，均是object的子类

相当于对基本数据类型的扩展

包装类与基本数据类型的转换

- 手动装箱拆箱
  - 装箱
    - 见笔记
  - 拆箱
    - 见笔记
- 自动装箱拆箱
  - 装箱
    - 见笔记
  - 拆箱
    - 见笔记
- 补充
  - ==号只要有基本数据类型，就是比较数值

包装类与String类型转换

- 包装类转String
  - 见笔记
- String转包装类
  - 见笔记

Integer和Character的常用方法

- Integer
  - MAX_VALUE
  - MIN_VALUE
- Character
  - isDigit(')
  - isletter()
  - islowercase()
  - isuppercase()
  - iswhitespace()
  - toUppercase()
  - tolowercase()

String类

- 特点
  - String对象用于保存字符串
  - 字符串字符使用Unicode编码，每个字符占两个字节
  - String类有很多构造器
  - 该类是final类，不可以被继承
  - String有属性 private final char【】value；用于存放字符串内容
    - 里面的值可以修改，但是不能改变value地址的指向
- String对象的创建方式
  - 直接创建 String str="1222";
    - 特点
      - 先看常量池有无该字符串，有的话直接把地址传给变量接受，没有的话就创建新的字符串在返回地址给变量
    - 规则
      - 先在池中开空间，在堆外value属性，先看常量池有无该字符串，有的话直接把地址传给value接受，没有的话就创建新的字符串在返回地址给value
      - 常量相加，看的是池，变量相加只要有变量就执行堆的内容，看的是堆
      - 变量相加用的是StringBuffer，创建他的新对象，然后再调用他的append方法，最后返回这个String对象
  - 构造器创建 String s=new String ("阿斯顿撒")
    - 特点
      - 两个对象，一个堆一个池
    - 用途
      - 可以用于数组转换成字符串，去掉括号等麻烦的东西
- String类的常用方法
  - equals
    - 区分大小写，判断字符串的内容是否相等
  - equalsignorecase
    - 不区分大小写，判断字符串内容是否相等
  - length
    - 返回字符串的长度
  - tolowercase
    - 把字符串转换成小写的字符串
  - touppercase
    - 把字符串转换成大写的字符串
  - indexof
    - 获取字符串在字符串中第一次出现的位置
  - lastindexof
    - 获取字符串在字符串中最后一次出现的位置
  - Substring
    - 获取指定位置的字符串，前闭后开，理解为后面的数字要减去一
  - charat
    - 获取字符串某个索引处的字符，不可以使用字符串对象【index】这个方式，因为他不是一个数组
  - concat
    - 拼接字符串，可以链式使用
  - replace
    - 替换字符串的内容，使用方法xx.replace('zzzz','sss')
  - split
    - 以指定的字符分割字符串，返回分割后的字符串数组，注意转义字符,////
  - compareto
    - 比较两个字符串，先按照两个字符的位置一一对比，如果发现不同就返回不同的字符那个字符的差值，如果全部相同且长度一致，就返回0，如果前面比较的相同，某一个字符串更长，返回长度的差值。
  - tochararry
    - 将字符串转化成单个的字符串数组，以char【】数组接受
  - format
    - 使用方法xxx.format("模板字符串"，变量) %s---String %c---char %d---int %.2f ----浮点型（保留两位小数）

Stringbuffer类

- 特点
  - 该类是一个final类，不可以被继承
  - 父类是abstractStringbuffer，且有属性char【】存放字符，不是final，存放在堆中
  - 不用每次都更换地址，所以效率更高
  - 实现了Serializeable接口，可以串行化
- Stringbuffer对象的创建方式
  - StringBuffer sub=new StringBuffer（）
  - StringBuffer sub=new StringBuffer（100）
  - StringBuffer sub=new StringBuffer（"asdaf"）
- Stringbuffer和String的转换
  - String转Stringbuffer
    - 使用构造器
      - String str="hello"；Stringbuffer sb=new Stringbuffer（str）
    - 使用append方法
      - Stringbuffer sb=new Stringbuffer（）；sb=sb.append(str)
  - Stringbuffer转String
    - 使用toString方法
      - String str="hello"；Stringbuffer sb=new Stringbuffer（str）；String s=sb.toString()
    - 使用构造器
      - String s= new Stringbuffer（Stringbuffer对象）
- Stringb类的常用方法
  - append
    - 可以链式使用
    - 如果添加的对象为空，则添加null字符进去
  - delete
    - 删除某个范围内的字符串，左闭右开
  - replace
    - 替换某个范围的字符串 repla（9,10,"ssss"）
  - insert
    - 在某个指定的位置插入指定字符串，在那个索引前面

Stringbuilder类

- 特点
  - 效率最高，但是线程不安全，被单个线程使用的时候，推荐使用
  - 直接父类是abstraStringbuilder，有属性char【】value，没有final，存在堆中
  - 实现了Serialize接口，可以串行化
  - 没有做互斥处理，没有synchronized
- 方法和Stringbuffer一致

Math类

- 常用方法
  - abs
  - pow
  - ceil
  - floor
  - round
  - sqrt
  - random 任意随机数（int）（a+math。random（）*（b-a+1））
  - max，返回两个数中的最大值
  - min 返回两个数中的最小值

Arrays类

- 常用方法
  - tostring
    - 返回数组的字符串形式，避免了写for循环的麻烦，返回样式【sss，111,2】
  - sort
    - 自然排序，从小到大排序
    - 定制排序，传入comparetor接口（以内部类的方式传入）返回一个整形，重写compare方法，在方法里面一般需要进行向下转型，适用进行定制排序（以首指排序为例子来理解就是确定交换的条件）
  - binarySearch
    - 二分查找排序，要求被查找的数组按照从小到大排好序，找到就返回index，找不到就返回-（low+1），low就是那个数应该在的位置
  - copyof
    - 使用方法，先声明一个数组来接受，然后Arrys.copy(被复制数组，复制长度)，如果复制长度大于被复制数组，就添加null，如果长度小于0，则返回NegaArrysizeException。该方法的底层使用的是System.arrycopy()方法
  - fill
    - 指定用一个元素去替换原先数组的所有值
  - equals
    - 返回boolean值，如果两个数组完全一样，就返回true
  - alist
    - 使用方法 List list=Arrys.aslist(2,3,45)，转换成数组，实际运行内存是Arrays类的静态内部类

System类

- exit
  - 退出程序，里面输入状态码，0表示正常退出，1表示不正常退出
- arraycopy
  - 复制数组元素，比较适合底层调用，使用方法arraycopy（源数组，源数组复制起始位置，复制目标位置，复制目标起始位置，复制长度），一般使用Arrays.copyof完成复制
- currentTimeMills
  - 返回从1970年1月1日到现今为止的毫秒数
- gc
  - 运行垃圾回收机制，特点是不会阻碍进程

bigInteger

- 特点，专门处理大数据，将大数转成字符串进行处理后再转成数字，适合保存大的整形
  - 声明方式
    - BigInteger bigInteger =new BigInteger（"212342342354543"）
  - 加
    - 先声明另一个操作大数，使用add
  - 减
    - 先声明另一个操作大数，使用subtract
  - 乘
    - 先声明另一个操作大数，使用multiply
  - 除
    - 先声明另一个操作大数，使用devide

BigDecimal

- 特点，专门处理大数据，将大数转成字符串进行处理后再转成数字，适合保存大的整形
  - 声明方式
    - Biginteger bigdecimal =new Bigdecimal（"0.212342342354543"）
  - 加
    - 先声明另一个操作大数，使用add
  - 减
    - 先声明另一个操作大数，使用subtract
  - 乘
    - 先声明另一个操作大数，使用multiply
  - 除
    - 先声明另一个操作大数，使用devide
    - 如果除以一个数是无限循环小数，则会报错无限小数
      - 解决方案
        - 在方法后面加上BigDecimal.ROUND_CEILING

Date

- SimpleDateFormat
  - 格式和解析日期的类，返回字符串
  - 使用方法
    - SimpleDateFormat sdf=new SimpleDateFormat(格式)，"yyyy-MM-dd HH-mm-ss"
    - 然后调用这个对象的format方法
    - 特点
      - 可以把给定的字符串转换成日期对象，调用该对象的parse方法，如果格式不对，则会抛出转换异常，用try-catch处理
- 使用方法
  - Date d1=new Date（）
    - 获取当前系统的时间，默认输出的格式是国外的方式
  - Date d1=new Date（毫秒数）
    - 指定毫秒数获取时间，返回自1970 年1月1日以来的时间

第二代日期类Calendar

- 特点：是一个抽象类，故不能实例化，提供了大量方法和字段如顺序员，他没有提供对应的格式化类
  - 可以通过getInstance（）来获取实例
    - 方法：Calendar c=Calandar.getInstance()
  - 获取年
    - c.get(Calandar.YEAR)
  - 获取月
    - c.get(Calandar.MONTH)+1
      - 从0开始
  - 获取日
    - c.get(Calandar.DATE_OF_MONTH)
  - 获取小时
    - c.get(Calandar.HOUR)
  - 获取分钟
    - c.get(Calandar.MINUTE)
  - 获取秒
    - c.get(Calandar.SECOND)

第三代日期类

- 前两代日期类的缺点
  - 可变性
    - 时间和日期这样的类应该是不可变得
  - 偏移性
    - Date中的年份是从1900年开始的，而月份都从0开始
  - 格式化
    - 格式化只对Date有用，Calendar则不行
  - 线程不安全，不能处理闰秒，每隔两天，多出一秒
- LocalDate(获取年月日)，LocalTime（获取时分秒），LocalDateTime（返回年月日，时分秒）
  - 使用
    - 使用类名.now（）获取当前时间对象
      - LocalDateTime dd=LocalDateTime.now()
    - 可以使用DateTimeFormatter ne=DateTimeFormatter.ofPattern()来指定格式，里面的格式跟SimpleDateFormat一致，然后调用他的format方法，返回字符串
    - 获取年
      - dd.getYear()
    - 获取月
      - dd.getMonth()
        - 返回月的英文缩写
    - 获取月（数字）
      - dd.getMonthValue()
    - 获取日
      - dd.getDayofMonth()
    - 获取时
      - dd.getHour()
    - 获取分
      - dd.getMinute()
    - 获取秒
      - dd.getSecond()
    - plus
      - 可以在当前时间对象上加天，年月，日等，返回一个新的时间对象
    - minus
      - 可以在当前时间对象上减天，年月，日等，返回一个新的时间对象，记得用类型去接受

时间戳

- 类似Date
  - 使用方法
    - Instant now=Instant.now()
  - Instant转Date
    - Date date=Date.form(now)
  - Date转Instant
    - Date date=new Date()　Instant instant=data.toInstant()