实用类

- 枚举 Enum

是一种特殊的数据类型，是一个类同时又比普通类多了一些约束，因为有约束，所以枚举具有简洁，安全，方便等特点。枚举的值被约束到一个特定的范围之中，只能取该范围以内的值。

在实际开发中，如果需要描述值有其特定的范围的数据时，比如性别，一年四季，周一到周天，可以使用枚举类型来描述。

枚举是由一组常量组成的类型，指定了一个区间，我们只能从该区间内取值。

枚举的定义和类很相似，使用enum关键字来描述，基本语法：

public enum 枚举名{

值1，值2，值3...

}

```
package com.southwind.test;

public enum WeekEnum {
    MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY,SUNDAY;
}
```

Java在编译期间会自动生成一个WeekEnum，并且是继承java.lang.Enum，同时被final修饰，表示该类不可被继承。

同时还生成了7个WeekEnum的实例对象分别对应枚举中定义的7个日期。

MONDAY的值是一个WeekEnum对应的抽象类的实例对象

TUESDAY的值也是一个WeekEnum对应的抽象类的实例对象

...

```
final class WeekEnum extends Enum{
    public static final WeekEnum MONDAY;
    public static final WeekEnum TUESDAY;
    public static final WeekEnum WEDNESDAY;
    public static final WeekEnum THURSDAY;
    public static final WeekEnum FRIDAY;
    public static final WeekEnum SATURDAY;
    public static final WeekEnum SUNDAY;
    private static final WeekEnum $VALUES[];

    public WeekEnum(String s,int i){
        super(s,i);
    }
```

```
    static{
        MONDAY = new WeekEnum("MONDAY",0);
        TUESDAY = new WeekEnum("TUESDAY",1);
        WEDNESDAY = new WeekEnum("WEDNESDAY",2);
        THURSDAY = new WeekEnum("THURSDAY",3);
        FRIDAY = new WeekEnum("FRIDAY",4);
        SATURDAY = new WeekEnum("SATURDAY",5);
        SUNDAY = new WeekEnum("SUNDAY",6);
        $VALUES = (new WeekEnum[]
{MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY,SUNDAY});
    }

    public static WeekEnum[] values(){
        return (WeekEnum[])$VALUES.clone();
    }

    public static WeekEnum valueOf(String s){
        return (WeekEnum)Enum.valueOf("com/southwind/test/WeekEnum",s);
    }

}
```

- Math

Math类为开发者提供了一系列的数学相关操作的静态方法，同时还提供了两个静态常量E（自然对数的底数）和PI（圆周率）。

```
package com.southwind.test;

public class MathTest {
    public static void main(String[] args) {
        /*
         * 常量E和PI
         */
        System.out.println("常量E: "+Math.E);
        System.out.println("常量PI: "+Math.PI);

        /*
         * 求平方根
         */
        System.out.println("5的平方根: "+Math.sqrt(9));

        /*
         * 求立方根
         */
        System.out.println("8的立方根: "+Math.cbrt(8));
```

```
        /*
         * 求x的y次方
         */
        System.out.println("2的3次方："+Math.pow(2, 3));

        /*
         * 求num1和num2中的较大值
         */
        System.out.println(Math.max(6, 5.5));

        /*
         * 求num1和num2中的较小值
         */
        System.out.println(Math.min(3, 2));

        /*
         * 求绝对值
         */
        System.out.println(Math.abs(-6));

        /*
         * 求大于num的最小整数
         */
        System.out.println(Math.ceil(4.1));

        /*
         * 求小于num的最大整数
         */
        System.out.println(Math.floor(4.999));

        /*
         * 四舍五入
         */
        System.out.println(Math.rint(5.6));

        /*
         * 四舍五入
         * num为float类型时返回int类型
         * num为double类型时返回long类型
         */
        System.out.println(Math.round(5.6f));
        System.out.println(Math.round(5.6));

        /*
         * 生成一个大于等于0.0且小于1.0的随机数
         */
        System.out.println(Math.random());
    }
}
```

- Random

```
package com.southwind.test;

import java.util.Random;

public class RandomTest {
    public static void main(String[] args) {
        /*
         * 无参构造
         */
        Random random = new Random();
        for (int i = 1; i <=10; i++) {
            /*
             * 随机生成一个boolean的值
             */
            boolean flag = random.nextBoolean();
            System.out.println(flag);
        }
        for (int i = 1; i <= 10; i++) {
            double num = random.nextDouble();
            System.out.println("第"+i+"个随机数是："+num);
        }
        for (int i = 1; i <= 10; i++) {
            float num = random.nextFloat();
            System.out.println("第"+i+"个随机数是："+num);
        }
        for (int i = 1; i <= 10; i++) {
            int num = random.nextInt();
            System.out.println("第"+i+"个随机数是："+num);
        }
        for (int i = 1; i <= 10; i++) {
            long num = random.nextLong();
            System.out.println("第"+i+"个随机数是："+num);
        }
    }
}
```

- String

String的实例化

1.直接赋值的方式 String str = "Hello";

2.通过构造函数创建：String() String(String str) ...

== 用法：如果操作数是基本数据类型，==判断值是否相等

如果操作数是引用数据类型，==判断内存地址是否相等

栈内存：变量都在栈内存中

堆内存：对象都在堆内存中

Student stu = new Student();

==判断栈内存中的值是否相等

Java在堆内存中提供了一个字符串常量池，专门用来存储String类型的对象，字符串常量池有一个特点，在实例化一个String对象时，会首先在字符串常量池中查找，如果该字符串已经在常量池中创建，则直接返回该字符的内存地址，如果常量池中没有该对象，则创建然后返回内存地址。

字符串常量池中只适用于通过直接赋值方式创建的字符串对象，如果是通过构造函数创建的字符串对象，不会保存到字符串常量池中，与其他对象创建是一样的。

基于上述的原因，实际开发中比较两个字符串是否相等，不能使用==来判断，而应该使用equals方法来判断。

```
int num = 1;

int num2 = 1;

num == num2; true

String str = new String("abc");

String str2 = new String("abc");

str == str2; false

String str3 = "abc";

String str4 = "abc";

str3 == str4; true
```

equals方法是定义在Object类中的方法，说明所有的对象都可以调用该方法，该方法的定义如下：

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

String常用的方法

- public String() 创建一个值为空的对象
- public String(String value) 创建一个值为value的对象
- public String(char value[]) 将一个char类型的数组转为字符串对象
- public String(char value[],int offset,int count) 将一个指定范围内的char型数组转为字符串对象
- public String(byte[] bytes) 将一个byte型数组转为字符串对象

- public String(byte bytes[],int offset,int count) 将一个指定范围内的byte型数组转为字符串对象
- public int length() 返回字符串长度
- public boolean isEmpty() 判断字符串是否为空
- public char charAt(int index) 返回字符串中指定位置的字符
- public byte[] getBytes() 将字符串对象转为byte类型的数组
- public boolean equals(Object anObject) 判断两个字符串的值是否相等
- public boolean equalsIgnoreCase(String anotherString) 判断两个字符串的值是否相等并且忽略大小写
- public int compareTo(String anotherString) 对两个字符串进行排序
- public boolean startsWith(String prefix) 判断字符串是否以指定的值开头
- public boolean endsWith(String suffix) 判断字符串是否以知道的值结尾
- public int hashCode() 获取字符串的哈希值
- public int indexOf(String str) 从头开始查找指定的字符位置
- public int indexOf(String str,int fromIndex) 从指定的位置开始查找指定的字符位置
- public String substring(int beginIndex) 截取字符串从指定的位置开始到结尾
- public String substring(int beginIndex,int endIndex) 截取字符串从指定位置开始到指定位置结束
- public String concat(String str) 追加字符串
- public String replacAll(String regex, String replacement) 替换字符串
- public String[] split(String regex) 用指定字符串对目标字符串进行分割，返回数组
- public String toLowerCase() 将字符串转为小写
- public String toUpperCase() 将字符串转为大写
- public char[] toCharArray() 将字符串转为char类型数组

```
package com.southwind.test;

public class Test {
    public static void main(String[] args) {
        char[] chars = {'a','b','c','d','e','f'};
        String str = new String(chars,2,3);
        System.out.println(str);
        byte[] bytes = {96,97,98};
        str = new String(bytes,1,2);
        System.out.println(str.length());
        String name = "张三";
        System.out.println(name.length());
        System.out.println(name.isEmpty());
        str = new String();
        System.out.println(str.isEmpty());
        str = "abcdefg";
        char cha = str.charAt(3);
        System.out.println(cha);
        byte[] by = str.getBytes();
        String str2 = "ABCDEFG";
        System.out.println(str.equals(str2));
        System.out.println(str.equalsIgnoreCase(str2));
        str = "a";
        str2 = "A";
```

```
        System.out.println(str.compareToIgnoreCase(str2));
        str = "helloworld";
        System.out.println(str.startsWith("hell"));
        System.out.println(str.endsWith("ld"));
        System.out.println(str.hashCode());
        System.out.println(str.indexOf("l"));
        System.out.println(str.indexOf("l", 5));
        System.out.println(str.substring(3,6));
        System.out.println(str.concat("java"));
        System.out.println(str.replaceAll("l", "o"));
        String[] array = str.split("l");
        System.out.println(array.length);
        for (String string : array) {
            System.out.println(string);
        }
        char[] chares = str.toCharArray();
        for (char item : chares) {
            System.out.println(item);
        }
    }
}
```

StringBuffer

当开发中字符串对象需要频繁修改时，如果使用String，需要频繁开辟内存空间，使用StringBuffer，修改之后不会开辟新的内存空间，而是在原有的基础上进行追加，所以使用StringBuffer可以避免重复开辟新的内存空间。

StringBuffer数组默认长度为16，即一个空的StringBuffer对象长度为16，如果通过构造函数创建一个有值的对象，长度为值的长度+16，16就是为修改对象而预留的空间。如果需要追加的值长度超过了16，会在原有的基础上对数组进行动态扩容，不会开辟新的内存空间来存值。

StringBuffer常用方法

public StringBuffer() 创建一个空的StringBuffer，长度为16

public StringBuffer(String str) 创建有值的StringBuffer，长度为str.length()+16

public synchronized int length() 返回StringBuffer的长度

public synchronized char charAt() 返回字符串中指定位置的字符

public synchronized StringBuffer append(String str) 追加字符串

public synchronized StringBuffer delete(int start,int end) 删除指定区间的字符

public synchronized StringBuffer deleteCharAt(int index) 删除指定位置的字符

public synchronized StringBuffer replace(int start,int end,String str) 将指定区间内的值替换为str

public synchronized String substring(int start) 截取字符串从指定位置开始到结尾

public synchronized String substring(int start,int end) 截取字符串从指定的位置开始到指定的位置结束

public synchronized StringBuffer insert(int offset,String str) 向指定的位置插入str

public int indexOf(String str) 从头开始查找指定字符的位置

public int indexOf(String str,int fromIndex) 从指定的位置开始查找指定字符的位置

public synchronized StringBuffer reverse() 进行反转

public synchronized String toString() 返回StringBuffer对应的String

```
package com.southwind.test;

public class Test2 {
    public static void main(String[] args) {
        StringBuffer stringBuffer = new StringBuffer("abc");
        stringBuffer = stringBuffer.append("defghi");
        System.out.println(stringBuffer);
        System.out.println(stringBuffer.length());
        System.out.println(stringBuffer.charAt(2));
        System.out.println(stringBuffer.delete(3, 6));
        System.out.println(stringBuffer.deleteCharAt(2));
        System.out.println(stringBuffer.delete(2, 3));
        stringBuffer = new StringBuffer("abcdefghi");
        System.out.println(stringBuffer.replace(2, 6, "he"));
        String str = stringBuffer.substring(3);
        System.out.println(str);
        str = stringBuffer.substring(3, 4);
        System.out.println(str);
        System.out.println(stringBuffer.insert(3, "java"));
        System.out.println(stringBuffer.indexOf("a"));
        System.out.println(stringBuffer.indexOf("a", 5));
        System.out.println(stringBuffer.reverse());
        System.out.println(stringBuffer.toString());
    }
}
```

日期

java.util.Date java.util.Calendar

Date

实例化对象表示当前系统的日期+时间

```java
package com.southwind.date;

import java.util.Date;

public class Test {
    public static void main(String[] args) {
        Date date = new Date();
        System.out.println(date);
    }
}
```

运行结果：

这种表示日期的格式不是我们中国人所习惯的形式，我们需要将日期的表示转为我们所习惯的方式，通过java.text.SimpleDateFormat类对Date对象进行格式化处理，并且格式化的规则我们可以自定义。

使用SimpleDateFormat时可以自定义pattern，pattern就是转换的规则，SimpleDateFormat提供了模版标记，用来拼接pattern。

y 年，yyyy表示4位数的年份信息

M 月，MM表示2位数的月份信息

m 分钟，mm表示2位数的分钟信息

d 天，dd表示2位数的天信息

H 小时，HH表示2位数的24小时制下的小时信息

h 小时，hh表示2位数的12小时制下的小时信息

s 秒，ss表示2位数的秒信息

S 毫秒，SSS表示3位数的毫秒信息

2018-10-23 20:25:30 - parttern yyyy-MM-dd HH:mm:ss

```java
package com.southwind.date;

import java.text.SimpleDateFormat;
import java.util.Date;

public class Test {
    public static void main(String[] args) {
        Date date = new Date();
        System.out.println(date);
        String pattern = "yyyy年MM月dd日 hh:mm:ss.SSS";
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
        String time = simpleDateFormat.format(date);
        System.out.println(time);
    }
}
```

Calendar

使用Date只能表示当前的系统时间，无法进行日期数据的逻辑运算，使用Calendar类可以完成日期数据的逻辑运算。

使用Calendar进行日期运算的基本思路：先将日期数据赋给Calendar，再调用Calendar的方法来完成相关的运算。

Calendar类提供了很多静态常量，用来记录日期数据

public static final int YEAR 年

public static final int MONTH 月

public static final int DAY_OF_MONTH 天，以月为单位，即当天是该月中的第几天

public static final int DAY_OF_YEAR 天，以年为单位，即当天是该年中的第几天

public static final int HOUR_OF_DAY 小时

public static final int MINUTE 分钟

public static final int SECOND 秒

public static final int MILLISECOND 毫秒

Calendar类常用的方法

public static Calendar getInstance() 获取Calendar实例化对象

public void set(int field,int value) 给静态常量所映射的值赋值

public int get(int field) 取出静态常量所映射的值

public final Date getTime() 获取Calendar对应的Date对象

```java
package com.southwind.date;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class CalendarTest {
    public static void main(String[] args) {
        //计算从今天算起15天之后的日期
        //1.将今天的日期赋给Calendar
        Date date = new Date();
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-
dd");
        String dateStr = simpleDateFormat.format(date);
        Calendar calendar = Calendar.getInstance();
        String[] dateArray = dateStr.split("-");
        calendar.set(Calendar.YEAR, Integer.parseInt(dateArray[0]));
        //1月为0，2月为1，...month为month-1
        calendar.set(Calendar.MONTH,Integer.parseInt(dateArray[1])-1);
        calendar.set(Calendar.DAY_OF_MONTH,
Integer.parseInt(dateArray[2]));
        //2.计算15天之后的日期
        int days = calendar.get(Calendar.DAY_OF_YEAR);
        int resultDay = days+9;
        calendar.set(Calendar.DAY_OF_YEAR, resultDay);
        Date date2 = calendar.getTime();
        String dateStr2 = simpleDateFormat.format(date2);
        System.out.println(dateStr2);

        //计算2018年8月6日往后推21天的日期
        calendar.set(Calendar.YEAR, 2018);
        calendar.set(Calendar.MONTH, 7);
        calendar.set(Calendar.DAY_OF_MONTH, 6);
        calendar.set(Calendar.DAY_OF_YEAR,
calendar.get(Calendar.DAY_OF_YEAR)+21);
        String laterDateStr = simpleDateFormat.format(calendar.getTime());
        System.out.println("2018年8月6日21天之后的日期："+laterDateStr);

        //计算2018年8月6日往前推21天的日期
        calendar.set(Calendar.YEAR, 2018);
        calendar.set(Calendar.MONTH, 7);
        calendar.set(Calendar.DAY_OF_MONTH, 6);
        calendar.set(Calendar.DAY_OF_YEAR,
calendar.get(Calendar.DAY_OF_YEAR)-21);
        String fontDateStr = simpleDateFormat.format(calendar.getTime());
        System.out.println("2018年8月6日21天之前的日期："+fontDateStr);
```

```
        //计算2018年8月6日所在的周是2018年的第几周
        calendar.set(Calendar.YEAR, 2018);
        calendar.set(Calendar.MONTH, 7);
        calendar.set(Calendar.DAY_OF_MONTH, 6);
        int week = calendar.get(Calendar.WEEK_OF_YEAR);
        int week2 = calendar.get(Calendar.WEEK_OF_MONTH);
        System.out.println("2018年8月6日所在的周是2018年的第"+week+"周");
        System.out.println("2018年8月6日所在的周是2018年8月的第"+week2+"周");
    }
}
```

```
2018-11-01
2018年8月6日21天之后的日期：2018-08-27
2018年8月6日21天之前的日期：2018-07-16
2018年8月6日所在的周是2018年的第32周
2018年8月6日所在的周是2018年8月的第2周
```