常用类

回顾

今天任务

```
1.常用基础类
        1.1 Date类
        1.2 Calendar类
        1.3 SimpleDateFormat类
        1.4 Math类
        1.5 Random类
        1.6 Runtime类
        1.7 System类
2.枚举
        2.1 什么是枚举
        2.2 枚举结合switch的使用
```

教学目标

```
1.了解常用类的作用
2.掌握常用类中的常用方法
3.了解枚举的使用
```

第一节 常用基础类

1.1 Date类

> Date表示特定的瞬间，精确到毫秒。Date类中的大部分方法都已经被Calendar类中的方法所取代。

> Date类中的构造方法：

方法名	描述
Date()	分配Date对象并初始化此对象，以表示分配它的时间（精确到毫秒）。
Date(long date)	分配Date对象并初始化此对象，以表示自从标准基准时间（称为“历元（epoch）”，即 1970 年 1 月 1 日 00:00:00 GMT）以来的指定毫秒数。

> Date类中的成员方法：

> 1.判断两个日期对象的前后

```
        Date date1 = new Date();//获取当前系统时间
        Date date2 = new Date(10000);//获取从标准基准时间起10000毫秒的时间点
        boolean boo1 = date1.after(date2);//若date1在date2之后，则返回true，否则返回false
        boolean boo2 = date1.before(date2);//若date1在date2之前，则返回true，否则返回false
```

> 2.比较两个日期对象

```
Date date1 = new Date();
Date date2 = new Date(10000);
int i = date1.compareTo(date2);
```

> 3.获取时间

```
Date date = new Date();
long l = date.getTime();
```

> 4.修改时间

```
Date date = new Date();
date.setTime(1000);
```

1.2 Calendar类

> 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（例如获得下星期的日期）提供了一些方法。瞬间可用毫秒值来表示，它是距历元（即格林威治标准时间 1970 年 1 月 1 日的 00:00:00.000，格里高利历）的偏移量。

> Calendar类时抽象类不能创建对象，可以通过Calendar类中的静态getInstance方法获取对象（一个费抽象子类对象）.

> Calendar类中的常用静态常量属性：

属性	描述（这些都是特定与日历的值）
public static final int YEAR	指示年的字段数字
public static final int MONTH	指示月份的字段数字
public static final int DATE	指示一个月中的某天
public static final int DAY_OF_MONTH	指示一个月中的某天
public static final int DAY_OF_WEEK	指示一周中的某天
public static final int DAY_OF_WEEK_IN_MONTH	指示当月中的第几个星期
public static final int DAY_OF_YEAR	指示一年中的某天

> Calendar类中常用的成员方法：

> 1.获取一个Calendar对象：

```
Calendar c = Calendar.newInstance();//newInstance方法是一个静态的方法，直接通过类名调用
System.out.println(c);
```

2.获取某个日历对象的指定属性的值：

```
/*
get(int field)  参数就是指定的属性，可以使用静态常量属性名代替int值
*/
//注意：获取日期属性，不能直接用c.DATE，DATE属性时静态常量，所有Calendar类对象都共享并相同的值
Calendar c = Calendar.newInstance();
System.out.println(c.get(Calendar.DATE));//获取c对象所表示日历的日期属性值
```

3.修改某个日历对象的指定属性值：

```
/*
set(int field, int value) 第一个参数表示属性，第二个参数表示修改的值
*/
Calendar c = Calendar.newInstance();
c.set(Calendar.DATE, 2017);
```

4.获取某个属性所表示的最大、最小值

```
/*
getMaximum(int field)        获取属性的最大值
getMinimum(int field)        获取属性的最小值
*/
Calendar c = Calendar.newInstance();
int max = c.getMaximum(Calendar.DATE);//获取日期的最大值,无论c表示几月份,max的值都是31
int min = c.getMinimum(Calendar.DATE);//获取日期的最小值,无论c表示几月份,min的值都是1
```

5.获取指定Calendar对象的指定属性的最大、最小值

```
/*
getActualMaximum(int field) 获取指定日历的指定属性的最大值
getActualMinimum(int field) 获取指定日历的指定属性的最小值
*/
Calendar c = Calendar.newInstance();
int max = c.getActualMaximum(Calendar.DATE);
//若c表示的1月，则max值为31；若c表示的4月，则max值为30；
int min = c.getActualMimimum(Calendar.DATE);//无论c表示几月份，min值都是1
```

1.3 SimpleDateFormat类

SimpleDateFormat是一个以与语言环境有关的方式来格式化和解析日期的具体类。

它允许进行格式化（日期 -> 文本）、解析（文本 -> 日期）和规范化。

> 通过SimpleDateFormat类将字符串和日期相互转换时，需要使用一些时间模式字母，常用的时间模式字母：

字母	日期或时间元素	示例
y	年	1996;96
M	年中的月份	July;Jul;07
w	年中的周数	27
D	年中的天数	189
d	月份中的天数	10
a	Am/pm 标记	PM
H	一天中的小时数（0-23）	0
h	am/pm 中的小时数（1-12）	12
m	小时中的分钟数	30
s	分钟中的秒数	55
S	毫秒数	978

> 1.格式化日期：

```
/*
format(Date date)    将date对象格式化成指定格式的日期字符串
*/
String format = "yyyy-MM-dd a hh:mm:ss";
SimpleDateFormat sdf = new SimpleDateFormat(format);
Date date = new Date();
String dateStr = sdf.format(date);
```

> 2.解析日期：

```
/*
parse(String str)    将str对象解析成一个日期对象
*/
String dateStr = "2017-12-01 上午 10:10:10";
String format = "yyyy-MM-dd a hh:mm:ss";
SimpleDateFormat sdf = new SimpleDateFormat(format);
Date date = sdf.parse(dateStr);
```

1.4 Math类

> 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。

> Math类中的静态常量属性

属性	描述
public static final E	比任何其他值都更接近 *e*（即自然对数的底数）的double值。
public static final PI	比任何其他值都更接近 *pi*（即圆的周长与直径之比）的double值。

> Math类中常用的成员方法：

> 1.绝对值

```
int a = -5;
System.out.println(Math.abs(a));//输出的结果是5
```

> 2.最大、最小值

```
int a = 5;
int b = 10;
System.out.println(Math.max(a,b));//输出结果为10
System.out.println(Math.min(a,b));//输出结果为5
```

> 3.幂运算

```
int a = 3;
int b = 4;
System.out.println(Math.pow(a,b));//结果为3*3*3*3的结果
```

> 4.平方根、立方根

```
int a = 8;
System.out.println(Math.sqrt(a));//结果为8的正平方跟
System.out.println(Math.cbrt(a));//结果为8的立方根
```

> 5.四舍五入

```
double a = 3.6;
System.out.println(Math.round(a));//结果为4
```

> 6.随机数

```
//产生一个3~9之间的随机数
int a = (int)(Math.random()*(9-3+1)+3);
```

1.5 Random类

> 此类的实例用于生成伪随机数流。此类使用 48 位的种子，使用线性同余公式 (linear congruential form) 对其进行了修改所得。

> 如果用相同的种子创建两个Random实例，则对每个实例进行相同的方法调用序列，它们将生成并返回相同的数字序列。

> Random类中的构造方法：

方法名	描述
Random()	创建一个新的随机数生成器。此构造方法将随机数生成器的种子设置为某个值，该值与此构造方法的所有其他调用所用的值完全不同。
Random(long seed)	使用单个 long 种子创建一个新的随机数生成器。该种子是伪随机数生成器的内部状态的初始值.

注意：若long种子确定，则在不同程序中，相同次数产生的随机数是相同的。

> Random类中的常用方法：
>
> 1.产生随机数

```
Random random = new Random(10);//以10为种子，使用线性同余公式产生伪随机数
int i1 = random.nextInt();//产生一个随机整数
int i2 = random.nextInt(10);//产生一个10以内的随机整数
double d = random.nextDouble();//产生一个随机double值
boolean b = random.nextBoolean();//产生一个随机boolean值
```

> 2.修改种子

```
Random random = new Random(10);
random.setSeed(20);//将随机数种子设置为20
```

1.6 Runtime类

> 每个 Java 应用程序都有一个Runtime类实例，使应用程序能够与其运行的环境相连接。可以通过getRuntime方法获取当前运行时。

1.7 System类

> System类包含一些有用的类字段和方法。它不能被实例化。

> System类中的常见属性：

属性	描述
public static fianl PrintStream err	标准错误输出流
public static final InputStream in	标准输入流
public static final PrintStream out	标准输出流

> System类中的常见方法：
>
> 1.获取系统时间

```
        long time1 = System.currentTimeMillis();//获取当前时间，毫秒数
        long time2 = System.nanoTime();//获取当前时间，毫微秒
```

> 2.强制退出虚拟机

```
        System.exit();//强制退出当前正在执行的虚拟机
```

> 3.垃圾回收处理机制：
>
> gc()： 运行垃圾回收器。

第二节 枚举

2.1 什么是枚举

> 枚举是一个引用类型，枚举就是一个规定了取值范围的变量类型

> 声明枚举：
>
> 格式：
>
> public enum 枚举名{
>
> //枚举的取值范围
>
> //枚举中可以生命方法
>
> }

```
        //声明一个表示季节的枚举
        public enum Season{
            SPRING,
            SUMMER,
            FULL,
            WINTER;
            public void fun(){}
        }
```

注意：

- 枚举中的成员是此枚举的取值范围;
- 所有的值之间使用逗号分隔，最后一个值后可以加分号也可以不加分号;
- 在枚举中可以声明方法，但是必须在所有的取值之后声明，而且最后一个值后必须使用分号隔开.

2.2 枚举结合switch结构使用

代码实现：

```
public class TestSeasonEnmu{
        public static void main(String[]args){
            //声明一个枚举变量
            Season season = Season.SPRING;
            //switch小括号中的表达式类型可以是枚举类型
            switch(season){
              //每个case后的常量直接写枚举的取值范围
              case SPRING:
                System.out.println("春天");
                break;
              case SUMMER:
                System.out.println("夏天");
                break;
              case FULL:
                System.out.println("秋天");
                break;
              case WINTER:
                System.out.println("冬天");
                break;
            }
        }
    }
```

总结

课前默写

> 1.设计一个方法，分别计算出一个字符串中字母，数字，下划线，空格和其他字符的个数
>
> 2.设计一个方法，模拟一个indexOf(int ch)方法

作业

> 使用Calendar完成一个万年历
>
> 要求：键盘输入一个年份和月份，控制台输出当月的日历

面试题

> 使用Math类产生一个随机数的方法