Date

步骤一：时间原点的概念

步骤二：创建日期对象

步骤三：getTime 的使用

步骤四：System.currentTimeMills()

步骤一：时间原点的概念

八种基本数据类型包括字符串最终都可以以数字的方式表现出来（数据就是二进制，而二进制是可以转换成 10 进制的）

日期类型类似，2022 年 1,1，也可以用一个数字来代替

在日期中最特殊的数字其实也是 0，0 这个日期代表的就是 java 中时间的原点——》1970,1,1 8,0,0 →1969 全球发布了第一个 unix 系统，综合考虑就认为 1970

凡是在这个 0 点的基础上，过一毫秒，意味着+1

步骤二：创建日期对象

需求：打印出当前时间，再打印一下从原点过了 100 秒的时间

```
        //当前时间
        Date date=new Date();
        System.out.println(date);//Wed Aug 28
14:39:16 CST 2019
        //从原点过了100秒的时间
        //1970 01 01 08:00:00  100
        //1970 01 01 08:01:40  100
        //1000毫秒=1秒  100000=100s
        Date date2=new Date(100000);
        System.out.println(date2);//Thu Jan 01
```

```
08:01:40 CST 1970
```

步骤三：getTime 的使用

getTime（）返回一个 long 整型

这个数代表从 197011800 开始每过一毫秒就增加 1

直接打印对象会出现这样的日期格式：Wed Aug 28 14:39:16 CST 2019
可是阅读性差，需要转换成数字，方便阅读，至于格式化后面会讲

```
        Date now=new Date();
        System.out.println(now);
        //当前时间距离原点时间一共过了多少毫秒
        System.out.println(now.getTime());
        //1566974943085
        //如果要得到原点
        Date zero=new Date(0);
        System.out.println(zero.getTime());
```

用来计算一段程序使用多少时间

```
        new TestDate01().forTest();
    }
    //观察一个程序执行多少时间
    public long startTime() {
        return new Date().getTime();
    }

    //10万次的for循环
    public void forTest() {
        long startTime=startTime();
        int sum=0;
        for(int i=1;i<100000;i++) {
            sum+=i;
            new DateTest("name:"+i);
        }
        System.out.println(sum);
        long endTime=endTime();
        long result=endTime-startTime;
```

```
            System.out.println(result);
        }

        public long endTime() {
            return new Date().getTime();
        }
}
class DateTest{
        private String name;

        public DateTest(String name) {
            super();
            this.name = name;
        }

        @Override
        public String toString() {
            return "DateTest [name=" + name + "]";
        }
}
```

步骤四：System.currentTimeMills()

当前日期的毫秒数

理论上说 getTime，currentTimeMills 两个值应该是一样的，但是由于机器性能不相同，彼此相差几十毫秒，因为每一行代码实际上都需要时间来运行

```
            Date now=new Date();
            //打印当前日期的毫秒数
            System.out.println(now.getTime());
            //通过system来获取

        System.out.println(System.currentTimeMillis());
```

日期格式化

日期格式化成字符串

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
Y	Week year	Year	2009; 09
M	Month in year (context sensitive)	Month	July; Jul; 07
L	Month in year (standalone form)	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day name in week	Text	Tuesday; Tue
u	Day number of week (1 = Monday, ..., 7 = Sunday)	Number	1
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800
X	Time zone	ISO 8601 time	-08; -0800; -

	zone	08:00

显示模式

Date and Time Pattern	Result
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 AD at 12:08:56 PDT
"EEE, MMM d, ''yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o''clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700
"yyyy-MM-dd'T'HH:mm:ss.SSSZ"	2001-07-04T12:08:56.235-0700
"yyyy-MM-dd'T'HH:mm:ss.SSSXXX"	2001-07-04T12:08:56.235-07:00
"YYYY-'W'ww-u"	2001-W27-3

```
            //y year 年
            //m month 月
            //d day 日
            //h hour 小时
            //m minute 分钟
            //s seconds 秒
            //s milliseconds 毫秒
            SimpleDateFormat sdf=new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss SS");
            Date now=new Date();
            String nowStr=sdf.format(now);
            System.out.println(nowStr);//2019-08-28
15:25:01 546
```

字符串格式化成日期

```
          //y year 年
          //m month 月
          //d day 日
          //h hour 小时
          //m minute 分钟
          //s seconds 秒
          //s milliseconds 毫秒
          SimpleDateFormat sdf=new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

          String str="2019/8/28 11:12:01";
          try {
               Date
date=sdf.parse(str);//DateForamt.parset(str)
               System.out.println(date);//Unparseable
date: "2019/8/28 11:12:01"
               //Wed Aug 28 11:12:01 CST 2019
          } catch (ParseException e) {
               e.printStackTrace();
          }
```

字符串转日期类的时候需要注意格式必须匹配到一起，否则可能会报异常

日历

通过日历类管理时间

```
          //java.util.GregorianCalendar
```

```
    //[time=1566979638106,areFieldsSet=true,areAllFie
ldsSet=true,lenient=true,zone=sun.util.calendar.ZoneI
nfo

    //[id="Asia/Shanghai",offset=28800000,dstSavings=
0,useDaylight=false,transitions=19,lastRule=null],

    //firstDayOfWeek=1,minimalDaysInFirstWeek=1,ERA=1
,YEAR=2019,MONTH=7,WEEK_OF_YEAR=35,WEEK_OF_MONTH=5,

    //DAY_OF_MONTH=28,DAY_OF_YEAR=240,DAY_OF_WEEK=4,D
AY_OF_WEEK_IN_MONTH=4,AM_PM=1,HOUR=4,HOUR_OF_DAY=16,

    //MINUTE=7,SECOND=18,MILLISECOND=106,ZONE_OFFSET=
28800000,DST_OFFSET=0]
```

```
        //单例模式构建一个日期对象（calendar-【年:
2011，月：1 日：1 时：8分：0秒:1】）
        Calendar c=Calendar.getInstance();
        System.out.println(c);
        //通过日历来获取日期
        Date d=c.getTime();
        System.out.println(d);
        Date d2=new Date();
        System.out.println(d2);
        //需求：我需要把日历的时间调回到时间的原点
        Date d3=new Date(0);//设置1970原点时间
        c.setTime(d3);//把时间设置进日历类中
        d=c.getTime();//通过日历类在显示出原点的时间类
        System.out.println(d);
```

日历最佳实践案例：翻日历

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;//util 工具
//需求：打印当前时间，下个月的今天，上个月的今天，去年的今
```

```
天，上个月的第三天
public class TestDate01 {
    //设定打印的时间格式
    //simpledateformat
    private static SimpleDateFormat sdf=new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

    public static void main(String[] args) {
        //搞一个日历耍耍
        Calendar c=Calendar.getInstance();

        //获取今天
        Date now=c.getTime();
        print(now);//2019-08-28 16:26:54
        //下个月的今天—每个月有固定的天数28 29 30 31
        //我要把今天加入日历
        c.setTime(now);
        //翻日历
        c.add(Calendar.MONTH,1);//下个月的今天 1 就是
一个月后多一天
        Date nextMonthNow=c.getTime();
        print(nextMonthNow);
        //获取去年的今天
        c.setTime(now);
        c.add(Calendar.YEAR,-1);
        print(c.getTime());
        //上个月的第三天
        c.setTime(now);
        c.add(Calendar.MONTH,-1);
        //不改变大字段，改变小字段--不改年月改日
        c.set(Calendar.DATE,3);
        print(c.getTime());

    }

    //打印格式
```

```
    private static void print(Date date) {
        System.out.println(sdf.format(date));
    }

}
```