1.System静态方法

1.1 System类提供的 public static long currentTimeMillis 用来返回当前时间与 1970 年 1 月 1 日 0 时 0 分 0 秒之间以毫秒为单位的时差。

1.2 此方法适于计算时间差。

```
// System类中的currentTimeMillis()
@Test
public void test1(){
long time = System.currentTimeMillis();
//返回当前时间与1970年1月1日0时0分0秒之间以毫秒为单位的时间差。
//称为时间戳
System.out.println(time);
}
```

1.3 计算世界时间的主要标准有

- UTC(Coordinated Universal Time)
- GMT(Greenwich Mean Time)
- CST(Central Standard Time)

2.Date类

2.1 java.util.Date类

|---java.sql.Date类

2.2 两个构造器的使用

>构造器一：Date()：创建一个对应当前时间的Date对象

>构造器二：Date(long date) 创建指定毫秒数的Date对象

2.3 两个方法的使用

>toString():显示当前的年、月、日、时、分、秒

>getTime():获取当前Date对象对应的毫秒数。（时间戳）

```
@Test
public void test2(){
//构造器一：Date()：创建一个对应当前时间的Date对象
Date date1 = new Date();
System.out.println(date1.toString());//Sat Feb 16 16:35:31 GMT+
08:00 2019

System.out.println(date1.getTime());//1550306204104

//构造器二：创建指定毫秒数的Date对象
 Date date2 = new Date(155030620410L);
 System.out.println(date2.toString());

 //创建java.sql.Date对象
 java.sql.Date date3 = new java.sql.Date(35235325345L);
 System.out.println(date3);//1971-02-13
 }
```

3.Calendar类

1.Calendar 是一个抽象基类，主用用于完成日期字段之间相互操作的功能。

2.获取 Calendar 实例的方法

方式一：创建其子类（GregorianCalendar）的对象

方式二：调用其静态方法getInstance()

3.一个 Calendar 的实例是系统时间的抽象表示，通过 get(int field)方法来取得想要的时间信息。比如 YEAR 、 MONTH 、 DAY_OF_WEEK 、 HOUR_OF_DAY 、
MINUTE 、 SECOND

- public void set(int field,int value)

- public void add(int field,int amount)
- public final Date getTime()
- public final void setTime (Date date)

4.注意

获取月份 时： 一月 是 0 ， 二月是 1 ， 以此类推 12 月是 11

获取星期时： 周日是 1 ， 周二是 2 。 。。。周六是 7

```
/*
Calendar日历类(抽象类）的使用

*/
@Test
public void testCalendar(){
//1.实例化
//方式一：创建其子类（GregorianCalendar）的对象
//方式二：调用其静态方法getInstance()
 Calendar calendar = Calendar.getInstance();
// System.out.println(calendar.getClass());

 //2.常用方法
 //get()
 int days = calendar.get(Calendar.DAY_OF_MONTH);
 System.out.println(days);
 System.out.println(calendar.get(Calendar.DAY_OF_YEAR));

 //set()
 //calendar可变性
 calendar.set(Calendar.DAY_OF_MONTH,22);
 days = calendar.get(Calendar.DAY_OF_MONTH);
 System.out.println(days);

 //add()
 calendar.add(Calendar.DAY_OF_MONTH,-3);
 days = calendar.get(Calendar.DAY_OF_MONTH);
 System.out.println(days);
```

```
29
30  //getTime():日历类---> Date
31  Date date = calendar.getTime();
32  System.out.println(date);
33
34  //setTime():Date ---> 日历类
35  Date date1 = new Date();
36  calendar.setTime(date1);
37  days = calendar.get(Calendar.DAY_OF_MONTH);
38  System.out.println(days);
39
40  }
```

4.SimpleDateFormat类

SimpleDateFormat的使用：SimpleDateFormat对日期Date类的格式化和解析

4.1 格式化 ： 日期 --->字符串

SimpleDateFormat () ：默认的模式和语言环境创建对象

public SimpleDateFormat (String pattern)该构造方法可以用参数pattern指定的格式创建一个对象，该对象调用：

public String format(Date date) 方法格式化时间对象 date

4.2 解析 ： 格式化的逆过程，字符串 ---> 日期

public Date parse(String source) 从给定字符串的开始解析文本，以生成一个日期

字母	日期或时间元素	表示	示例
G	Era 标志符	Text	AD
y	年	Year	1996; 96
M	年中的月份	Month	July; Jul; 07
w	年中的周数	Number	27
W	月份中的周数	Number	2
D	年中的天数	Number	189
d	月份中的天数	Number	10
F	月份中的星期	Number	2
E	星期中的天数	Text	Tuesday; Tue
a	Am/pm 标记	Text	PM
H	一天中的小时数（0-23）	Number	0
k	一天中的小时数（1-24）	Number	24
K	am/pm 中的小时数（0-11）	Number	0
h	am/pm 中的小时数（1-12）	Number	12
m	小时中的分钟数	Number	30
s	分钟中的秒数	Number	55
S	毫秒数	Number	978
z	时区	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	时区	RFC 822 time zone	-0800

```java
@Test
public void testSimpleDateFormat() throws ParseException {
//实例化SimpleDateFormat:使用默认的构造器
SimpleDateFormat sdf = new SimpleDateFormat();

//格式化：日期 --->字符串
Date date = new Date();
System.out.println(date);

 String format = sdf.format(date);
 System.out.println(format);

 //解析：格式化的逆过程，字符串 ---> 日期
 String str = "19-12-18 上午11:43";
 Date date1 = sdf.parse(str);
 System.out.println(date1);

 //**********按照指定的方式格式化和解析：调用带参的构造器*******
**********
```

```
// SimpleDateFormat sdf1 = new SimpleDateFormat("yyyyy.MMMMM.dd GGG hh:mm aaa");
SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
//格式化
String format1 = sdf1.format(date);
System.out.println(format1);//2019-02-18 11:48:27
//解析:要求字符串必须是符合SimpleDateFormat识别的格式(通过构造器参数体现),
//否则，抛异常
Date date2 = sdf1.parse("2020-02-18 11:48:27");
System.out.println(date2);
}
```

```
练习一：字符串"2020-09-08"转换为java.sql.Date
@Test
public void testExer() throws ParseException {
String birth = "2020-09-08";

SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-MM-dd");
Date date = sdf1.parse(birth);
// System.out.println(date);

 java.sql.Date birthDate = new java.sql.Date(date.getTime());
System.out.println(birthDate);
}
```