1. 日期时间类
   1. Date类

概述：java.util.Date 类表示特定的瞬间，精确到毫秒。

/\*  
Java.util.Date：表示日期和时间的类  
类 Date 表示特定的时间，精确到毫秒。  
毫秒：千万分之一 1000毫秒 == 1秒  
特定的时间：一个时间点，一刹那时间  
2020-5-26 08:35:333 （瞬间）  
...  
毫秒值的作用：可以对时间和日期进行计算。  
2099-01-03 到 2088-01-01 中间一个有多多好天。  
思路：可以把日期转换为毫秒进行计算，计算完毕，在把毫秒转换为日期。  
  
把日期转换为毫秒：  
 当前日期：2088-01-01  
 时间原点(0毫秒)：1970年1月1日 00:00:00（ IDK-APl中已经定义,英国格林威治）  
 就是计算当前日期到时间原点之间经历了多少毫秒。（1590454351727L）  
  
注意：  
 中国属于东八区，会把时间增加8个小时。  
 1970 年 1 月 1 日 08:00:00 （中国）  
  
把毫秒变为日期：  
 一天 = 24 x 60 x 60 = 86400秒 = 86400 x 1000 = 86400000毫秒  
 \*/  
public class Demo01Date {  
 public static void main(String[] args) {  
  
 // 获取当前系统时间到 1970年1月1日00:00:00 经历了多少毫秒。  
 System.out.println(System.currentTimeMillis()); // 1590454351727  
 }  
}

Date 的使用方法类型：

package demo02;  
  
import java.util.Date;  
  
public class Demo02Date {  
 public static void main(String[] args) {  
 demo01();  
 demo02();  
 demo03();  
 }  
  
 /\* 一、  
 Date类的常参数构造方法：  
 Date(long date):传递毫秒值，把 毫秒 转换为 Date 日期  
 \*/  
 private static void demo01(){  
 Date date01 = new Date(0L);  
 System.out.println(date01); // Thu Jan 01 08:00:00 CST(中国) 1970  
  
 Date d1 = new Date(200000L);  
 System.out.println(d1); // Thu Jan 01 08:03:20 CST 1970  
 }  
  
 /\* 二、  
 Date 类的空参数构造方法：  
 Date() 获取的就是当前系统的日期和时间。  
 \*/  
 private static void demo02(){  
 Date date02 = new Date(); //注意：要选择 java.util.Date  
 System.out.println(date02); // Tue May 26 09:09:24 CST 2020  
 }

/\* 三、  
 Date 类的成员方法  
 long getTIme() 把当前日期转换为毫秒（相当于System.currentTimeMillis())  
 返回 1970 年 1 月 1 日 00:00:00 GMT(英国格林维尔) 以来 Date 对象表示的毫秒数。  
 \*/  
 private static void demo03(){  
 Date date03 = new Date();  
 long time = date03.getTime();  
 System.out.println(time); // 1590457237193  
 }  
}

DateFormat 类

package demo02;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
/\*  
概述：java.text.DateFormat：是日期/时间格式化子类的抽象类，  
  
作用：格式化(日期 -> 文本) 、解析(文本 -> 日期)  
  
日期：Date类 文本：String类  
格式化（Format）：按照所指定的格式，从Date对象转换为String对象。  
解析：按照指定的格式，从String对象转换为Date对象。  
  
注意：DateFormat 是 Format 的子类，并且 DateFormat 为抽象类，  
 如要使用需要使用其子类 simpleDateFormat(已知子类)，或定义其他实现类。  
  
成员方法：  
 String format(Date date)：按照指定的模式，把Date日期，格式化为符合模式的字符串。  
 Date parse(String source)：把符合模式的字符串，解析为 Date 日期。  
DateFormat类是一个抽象类，无法直接创建对象使用，可以使用 DateFormat 的子类。(SimpleDateFormat)  
  
java.text.SimpleDateFormat extends DateFormat  
  
SimpleDateFormat构造方法:  
 SimpleDateFormat(String pattern):用给定的模式和默认语言环境的日期格式符号构造 SimpleDateFormat  
 参数：  
 String pattern：传递指定的模式  
 模式：（区分大小写） 在 JDK-APL 中可以查到。  
 y 年  
 M 月  
 d 日  
 H 时  
 m 分  
 s 秒  
 写对应的模式，会把模式替换为对应的日期和时间  
 "yyyy-MM-dd HH:mm:ss"  
 "yyyy年MM月dd日 HH时mm分ss秒"  
 注意：  
 模式中的字母不能更改，链接模式的符号可以改变。  
  
  
 \*/  
public class Demo03DateFormat {  
 public static void main(String[] args) throws ParseException {  
 demo01();  
 demo02();  
 }  
 /\*  
 使用DateFormat类中的方法 format，把日期格式化为文本。  
 String format(Date date) 按照指定的模式，把 Date日期，格式化为符合模式的字符串。  
 使用步骤：  
 1.创建 SimpleDateFormat 对象，构造方法中传递指定的模式。  
 2.调用 SimpleDateFormat 对象中的方法 format,按照构造方法中指定的模式，  
 把 Date日期格式转化为模式的字符串（文本）。  
 \*/  
 private static void demo01(){  
  
 // 1. 创建SimpleDateFormat对象，构造方法中传递指定的模式。  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
  
 // 2. 调用 SimpleDateFormat对象中的方法format，按照构造方法中指定的模式，  
 // 把Date日期格式化为符合模式的字符串（文本）  
 Date date = new Date(); // 一定要导入 java.util.Date();  
 String text = sdf.format(date);  
 System.out.println(date); // Tue May 26 16:41:32 CST 2020  
 System.out.println(text); // 2020-05-26 16:41:32  
 }  
 /\*  
 使用 DateFormat类中的方法parse，把文本解析为日期。  
 Date parse(String source) 把符合模式的字符串，解析为Date日期。  
 使用步骤：  
 1. 创建 SimpleDateFormat 对象，构造方法中传递指定的模式。  
 2. 调用 SimpleDateFormat 对象中的方法parse，把符合构方法模式中的字符串，解析为Date日期。  
 注意：  
 public Date parse(String source) throws ParseException  
 parse方法声明了一个异常叫 ParseException 解析异常。  
 如果字符串和构造方法中的模式不一样，那么程序就会抛出异常。  
 调用一个抛出了异常的方法，就必须处理这个异常，要么throws继续声明抛出这一个异常  
 要么 try...catch 自己处理这个异常。  
 快捷键方法：  
 鼠标选中 parse 按 Alt + Enter，  
 同样 demo02() 调用了这个方法，所以也需要进行 Alt + Enter 处理  
  
  
 \*/  
 private static void demo02() throws ParseException {  
  
 // 1. 创建一个 SimpleDateFormat 对象，构造方法中传递的指定模式。  
 SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
 // 2. 调用 SimpleDateFormat 对象中的方法parse，把符合构方法模式中的字符串，解析为Date日期。  
 Date date = sdf2.parse("2020-05-26 16:41:32");  
 System.out.println(date); // Tue May 26 16:41:32 CST 2020  
 }  
  
}

* 1. 练习

package demo02;  
  
/\*  
 练习：  
 请使用日期时间相关的API，计算一个人已经出生多少天。  
 分析：  
 1. 使用 Scanner类中的方法next，获取出生日期。  
 2. 使用DateFormat类中的方法parse，把字符串的出生日期解析为Date格式。  
 3. 把Date格式的出生日期转换为毫秒值。  
 4. 获取当前的日期，转换为毫秒值。  
 5. 使用当前日期的毫秒值-出生日期的毫秒值。  
 6. 把毫秒值的插值转换为天(s/1000/60/60/24)  
 \*/  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Scanner;  
  
public class Demo02Text {  
 public static void main(String[] args) throws ParseException {  
 // 1.使用Scanner类中的方法next，获取出生日期。  
 Scanner sc =new Scanner(System.in);  
 System.out.println("请输入您的出生日期，格式为yyyy-MM-dd");  
 String birthdayDateString = sc.next();  
  
 // 2.使用DateFormat类中的方法parse，把字符串的出生日期解析为Date格式。  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
 Date birthDate = sdf.parse(birthdayDateString);  
  
 // 3.把 Date格式的出生日期转换为毫秒值  
 long birthDateTime = birthDate.getTime();  
  
 // 4.获取当前的日期，转换为毫秒值。  
 long todayTime = new Date().getTime();  
  
 // 5.使用当前日期的毫秒值-出生日期的毫秒值。  
 long time = todayTime - birthDateTime;  
  
 // 6.把毫秒值的插值转换为天(s/1000/60/60/24)  
 System.out.println(time/1000/60/60/24);  
  
 }  
}

package demo02;  
  
/\*  
 练习：  
 请使用日期时间相关的API，计算一个人已经出生多少天。  
 分析：  
 1. 使用 Scanner类中的方法next，获取出生日期。  
 2. 使用DateFormat类中的方法parse，把字符串的出生日期解析为Date格式。  
 3. 把Date格式的出生日期转换为毫秒值。  
 4. 获取当前的日期，转换为毫秒值。  
 5. 使用当前日期的毫秒值-出生日期的毫秒值。  
 6. 把毫秒值的插值转换为天(s/1000/60/60/24)  
 \*/  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Scanner;  
  
public class Demo02Text {  
 public static void main(String[] args) throws ParseException {  
 // 1.使用Scanner类中的方法next，获取出生日期。  
 Scanner sc =new Scanner(System.in);  
 System.out.println("请输入您的出生日期，格式为yyyy-MM-dd");  
 String birthdayDateString = sc.next();  
  
 // 2.使用DateFormat类中的方法parse，把字符串的出生日期解析为Date格式。  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
 Date birthDate = sdf.parse(birthdayDateString);  
  
 // 3.把 Date格式的出生日期转换为毫秒值  
 long birthDateTime = birthDate.getTime();  
  
 // 4.获取当前的日期，转换为毫秒值。  
 long todayTime = new Date().getTime();  
  
 // 5.使用当前日期的毫秒值-出生日期的毫秒值。  
 long time = todayTime - birthDateTime;  
  
 // 6.把毫秒值的插值转换为天(s/1000/60/60/24)  
 System.out.println(time/1000/60/60/24);  
  
 }  
}

1. Calendar 类(日历类）

Calendar 中的默认子类 getInstance() （可以获得当前的日期）

import java.util.Calendar;  
  
/\*  
java.util.Calendar类：日历类  
Calendar 类是一个抽象类，里边提供了很多操作日历字段的方法(YEAR、MONTH、DAY\_OF\_MONTH、HOUR)。  
Calendar类无法直接创建对象使用，里边有一个静态方法叫getInstance()，该方法返回了 Calendar 类的子类对象。  
  
static Calendar getInstance() 使用默认时区和语言环境获得一个日历。  
注意：西方的月份从 0 到 11 月，中国的为 1 到 12 月。  
 \*/  
public class Demo03Calendar {  
 public static void main(String[] args) {  
 Calendar c = Calendar.getInstance(); // 多态  
 System.out.println(c);  
 }  
}

Calendar 类常用的方法

package demo03;  
  
import java.util.Calendar;  
import java.util.Date;  
  
/\*  
Calendar类的成员方法：  
 - public int get(int filed):返回给定日历字段的值。  
 - public void set(int filed,int value):将给定的日历字段设置为定值。  
 - public abstract void add(int filed,int amount):根据日历的规则，为给定的日历段添加或减去指定的时间量。  
 - public Date getTime():返回一个表示此Calendar时间值(从历元到现在的毫秒偏移量) 的 Date对象。  
成员方法的参数：  
 int filed:日历类的字段，可以使用 Calendar 类的静态成员变量获取。  
 public static final int YEAR = 1 ; 年  
 public static final int MONTH = 2 ; 月  
 public static final int DATE = 5 ; 月中的某一天  
 public static final int DAY\_oF\_MONTH = 5 ; 月中的某一天  
 public static final int HOUR = 10 ; 时  
 public static final int MINTER = 12 ; 分  
 public static final int SECOND = 13 ; 秒  
 \*/  
public class Demo03Calendar {  
 public static void main(String[] args) {  
 demo01();  
 demo02();  
 demo03();  
 demo04();  
 }  
 /\*  
 public int get(int filed):返回给定日历字段的值。  
 参数：传递指定的日历字段(YEAR,MONTH...)  
 返回值：日历字段的具体的值  
 \*/  
 private static void demo01(){  
  
 // 使用 getInstance 方法获取 Calendar 对象。即为当前的日期。  
 Calendar c = Calendar.getInstance();  
 System.out.println(c);  
  
 // 获取当前的年份  
 int year = c.get(Calendar.YEAR); // 静态的成员变量 直接对象名.方法即可调用  
 System.out.println(year); // 2020  
  
 int month = c.get(Calendar.MONTH);  
 // 注意：（西方：0~11，东方：1~12）  
 System.out.println(month); // 4 此时打印的为西方的  
 System.out.println(month+1); // 5 此时打印的为东方的  
  
 // int date = c.get(Calendar.DATE);  
 int date = c.get(Calendar.DAY\_OF\_MONTH);  
 System.out.println(date); // 2020 5 28  
 }  
 /\*  
 public void set(int filed,int value):将给定的日历字段设置为定值。  
 参数：  
 int filed:传递指定的日历字段(YEAR,MONTH...)  
 返回值：日历字段代表的具体值。  
 \*/  
 private static void demo02(){  
  
 // 使用 getInstance 方法获取 Calendar 对象。即为当前的日期。  
 Calendar c = Calendar.getInstance();  
  
 // 设置年份为 1999  
 c.set(Calendar.YEAR,1999); // 年份会从当前年份变为 1999  
  
 // 获取 当前的年份  
 int year = c.get(Calendar.YEAR); // 静态的成员变量 直接对象名.方法即可调用  
 System.out.println(year); // 1999  
  
 // 获取 当前的年份  
 int year01 = c.get(Calendar.YEAR); // 静态的成员变量 直接对象名.方法即可调用  
 System.out.println(year01); // 1999  
  
 // 方法重载：可以同时设置 年月日  
 c.set(1997,5,5); // 1997 5 5  
 }  
 /\*  
 public abstract void add(int filed,int amount):  
 根据日历的规则，为给定的日历段添加或减去指定的时间量。  
 把指定的字段增加/减少 指定的值。  
 参数：  
 int filed:传递指定的日历字段(YEAR,MONTH...)  
 int amount：增加/减少的值  
 注意：传递正数就增加，传递附属就减小。  
 \*/  
 private static void demo03(){  
  
 // 使用 getInstance 方法获取 Calendar 对象。即为当前的日期。  
 Calendar c = Calendar.getInstance();  
  
 // 把年增减两年  
 c.add(Calendar.YEAR,2);  
  
 // 获取增加后的年份  
 int year02 = c.get(Calendar.YEAR); // 静态的成员变量 直接对象名.方法即可调用  
 System.out.println(year02); // 2022  
  
 // 把月减少三个月  
 c.add(Calendar.MONTH,-3);  
  
 // 获取减少后的月份  
 int month01 = c.get(Calendar.MONTH);  
 System.out.println(month01+1); // 2 东方  
 }  
 /\*  
 public Date getTime():  
 返回一个表示此Calendar时间值(从历元到现在的毫秒偏移量) 的 Date对象。  
 把日历对象，转换为日期对象。  
 \*/  
 private static void demo04(){  
  
 // 使用 getInstance 方法获取 Calendar 对象。即为当前的日期。  
 Calendar c = Calendar.getInstance();  
  
 // 把日历对象，转换为日期对象。  
 Date date = c.getTime();  
 System.out.println(date); // Fri May 29 00:07:32 CST 2020  
 }  
}

3.3 日期相关类的总结

Date 类

A：构造方法

Date(): 根据当前系统时间创建日期对象

Date(long time)：根据传入的毫秒值时间创建日期对象

B：成员方法

long getTime() : 获取当前日期对象的毫秒时间。

String toLocaleString() : 根据本地格式转换日期对象

public class Test01 {  
 public static void main(String[] args) {  
  
 // 空参数构造方法  
 Date d = new Date(); // 一定要记住进行快捷键 Alt+Enter 进行导包。  
 System.out.println(d); // Fri May 29 13:28:16 CST 2020 （外国）  
 System.out.println(d.toLocaleString()); // 2020年5月29日 下午1:29:37(国内）  
 }  
}

DateFormat类&SimpleDateFormat类

A: 构造方法

SimpleDateFormat(String s) :根据指定的模板创建日期格式化对象。

B：成员方法

String format(Date d) :根据指定格式格式化日期对象

Date parse(String s) :根据指定格式解析字符串。

package demo02;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Test01 {  
 public static void main(String[] args) {  
  
 // 空参数构造方法  
 Date d = new Date(3000L); // 一定要记住进行快捷键 Alt+Enter 进行导包。  
 System.out.println(d.toLocaleString()); // 1970年1月1日 上午8:00:03  
  
 // 创建日期格式化对象  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss");  
 String format = sdf.format(d);  
 System.out.println(format); // 1970年01月01日 08:00:03

String str = "2018年08月08日 08:08:08";  
Date parse = sdf.parse(str);  
System.out.println(parse); // Wed Aug 08 08:08:08 CST 2018 (国外）

}  
}

Calendar 类

A：创建对象的方法

Calendar c = Calendar.newInstance() 获取日历对象

B：成员方法

int get(int n) : 获取指定日历字段信息。

void set(int n,int value) : 将指定日历字段设置为指定的值。

void add(int n,int value) : 将指定日历字段增减/减少指定的值。

package demo02;  
  
import java.util.Calendar;  
  
public class Text02 {  
 public static void main(String[] args) {  
 //获取当前日历类对象  
 Calendar c= Calendar.getInstance();  
  
 // get方法 获取当前年份  
 int year = c.get(Calendar.YEAR);  
 System.out.println(year); // 2020  
  
 // get 方法，获取当前月份  
 int month = c.get(Calendar.MONTH)+1; // 获取当前国内的月份 加1  
 System.out.println(month); // 5  
  
 // set方法，改变当前年份  
 c.set(Calendar.YEAR,2088);  
 int year01 = c.get(Calendar.YEAR);  
 System.out.println(year01); // 2088  
  
 // add方法，增加/减少年份  
 c.add(Calendar.YEAR,2);  
 System.out.println(c.get(Calendar.YEAR)); // 2090  
 }  
}

System 类

package demo4;  
  
import java.util.Arrays;  
  
/\*  
java.lang.System类中提供了大量的静态方法，可以获取与系统相关的信息或系统级操作；  
在System类的API文档中，常用方法有：  
 - public static long currentTimeMillis() : 返回以毫秒为单位的当前时间。  
 - public static void arraycopy(Object src,int srcPos,Object dest,int destPos,int length)：  
 将数组中指定的数据拷贝到另一个数组中。  
 \*/  
public class DemoSystem {  
 public static void main(String[] args) {  
 demo01();  
 demo02();  
 }  
 /\*  
 public static long currentTimeMillis() : 返回以毫秒为单位的当前时间。  
 作用：  
 用来测试程序的效率  
 练习：  
 验证for循环打印数字 1~9999所需要的时间（毫秒）  
 \*/  
 private static void demo01(){  
  
 // 程序执行时，获取一次毫秒值  
 long s = System.currentTimeMillis();  
  
 // 执行for循环  
 for (int i = 0; i < 9999; i++) {  
 System.out.println(i);  
 }  
  
 // 程序执行后，在获取一次毫秒值。  
 long c = System.currentTimeMillis();  
 System.out.println("程序供耗时：" + (c-s) +"毫秒"); // 程序供耗时：94毫秒  
 }  
  
 /\*

public static void arraycopy(Object src,int srcPos,Object dest,int destPos,int length)：  
 将数组中指定的数据拷贝到另一个数组中。  
  
 参数：  
 src - 源数组。  
 srcPos - 源数组中的起始位置。  
 dest - 目标数组。  
 destPos - 目标数据中的起始位置。  
 length - 要复制的数组元素的数量。  
 练习：  
 将src数组中的前3个元素，复制到dest数组的前3个位置上  
 复制元素前：src数组元素[1,2,3,4,5]，dest数组元素[6,7,8,9,10]  
 复制元素后：src数组元素[1,2,3,4,5]，dest数组元素[1,2,3,9,10]  
 \*/  
 private static void demo02(){  
  
 // 当以源数组  
 int[] src = {1,2,3,4,5};  
  
 // 定义目标数组  
 int[] dest = {6,7,8,9,10};  
 System.out.println("复制前：" + Arrays.toString(dest)); // 复制前：[6, 7, 8, 9, 10]  
  
 // 使用System类的方法将src数组中的前3个元素，复制到dest数组的前3个位置上  
 System.arraycopy(src,0,dest,0,3);  
 System.out.println("复制后：" + Arrays.toString(dest)); // 复制后：[1, 2, 3, 9, 10]  
 }  
}