●常见对象(正则表达式的概述和简单使用)

- A:正则表达式
  - 是指一个用来描述或者匹配一系列符合某个语法规则的字符串的单个字符串。其实就是一种规则。有自己特殊的应用。

作用：比如注册邮箱，邮箱用户名和密码，一般会对其进行限制长度，这个限制长度的事情就是正则表达式做的。

  - B:案例演示
  - 需求：校验 qq 号码.
    - 1:要求必须是 6-12 位数字
    - 2:0 不能开头
    - 3:必须都是数字
- a:非正则表达式实现

```
package com.sxt.homework;

public class Demo {
    public static void main(String[] args) {
        /*
           1）要求必须是6-12 位数字  :  qq.length
            2）0 不能开头    :qq.startWith("0")判读是否以0 开头
             3）必须都是数字 :  '7' >= 48 && '7' <= 57
         */
        String qq = "78978967823";
        String result =  check(qq)? "是正确的 qq 号" : "qq 不合法";
        System.out.println(result);
    }
    public static boolean check(String qq){
        boolean flag = true;//申明一个布尔值
      if(qq.length() >= 6 && qq.length() <= 12){
            //满足条件:长度在6-12 位之间
          if(!qq.startsWith("0")){
                //满足0 不能开头
            for (int i = 0; i <qq.length() ; i++) {
                    char c = qq.charAt(i);
                    if(c >=48 && c<= 57){
                        //满足都是数字

                    }else{
                            flag = false;
                        }
                    }
            }else{
                //这里0 开头进入这里
             flag = false;
            }
        }else{
            flag = false;
        }
        return flag;
```

```
    }
}
```

- b:正则表达式实现

```
package com.sxt.homework;

public class Demo {
    public static void main(String[] args) {

        String qq = "089780967831";
        String regex = "[1-9]\\d{5,11}";

        boolean result = qq.matches(regex);//matches 判断是否匹配,如果匹配返回 true
        System.out.println(result);
        /*
             [1-9]\\d{5,11}   正则表达式
             [1-9] :表示匹配 1~9 之间的任意一个数
             \\d  :第一个\是转义符; \d :表示 0~9 之间的数字
             {5,11}:表示\\d 最少重复出现 5 次,最多重复出现 11 次

         */
    }

}
```

●常见对象(预定义字符类演示)

- A:预定义字符类
  - . 任何字符(一个点代表一个任意字符)。
  - \d 数字：等同于[0-9]
  - \w 单词字符：等同于[a-zA-Z_0-9]
  - [abc] a、b 或 c（简单类）
  - [^abc] 任何字符，除了 a、b 或 c（否定）
  - [a-zA-Z] a 到 z 或 A 到 Z，两头的字母包括在内（范围）
  - [0-9] 0 到 9 的字符都包括

```
package com.sxt.homework;
public class Demo {
    public static void main(String[] args) {

      /*
    . 任何字符(一个点代表一个任意字符)。
   \d 数字：等同于[0-9]
   \w 单词字符：等同于[a-zA-Z_0-9]
   [abc] a、b 或 c（简单类）
```

```
    ▪ [^abc] 任何字符，除了 a、b 或 c（否定）
    ▪ [a-zA-Z] a 到 z 或 A 到 Z，两头的字母包括在内（范围）
    ▪ [0-9] 0 到9 的字符都包括
        */
        System.out.println("------匹配任何一个字符-------");
        String regex1 = ".";
        System.out.println("长".matches(regex1));
        System.out.println("------匹配\\d 匹配 0-9 之间任意一个字符-------");
        String regex2 = "\\d";
        System.out.println("3".matches(regex2));
        System.out.println("01".matches(regex2));
        System.out.println("------匹配\\w 匹配 a-zA-Z_0-9 之间任意一个字符-------");

        String regex3 = "\\w";
        System.out.println("2".matches(regex3));
        System.out.println("w".matches(regex3));
        System.out.println("E".matches(regex3));
        System.out.println("------匹配[abc]匹配 a 或 b 或 c 中任意一个字符-------");
        String regex4 = "[abc]";
        System.out.println("b".matches(regex4));
        System.out.println("d".matches(regex4));
        System.out.println("a".matches(regex4));
        System.out.println("ab".matches(regex4));
        System.out.println("------匹配[^abc]匹配除了 a 或 b 或 c 之外的任意一个字符-------");

        String regex5 = "[^abc]";
        System.out.println("e".matches(regex5));
        System.out.println("f".matches(regex5));
        System.out.println("你".matches(regex5));
        System.out.println("a".matches(regex5));
        String regex6 = "[changsha]";//匹配 changsha 这些字符中的任意一个字符
       System.out.println("z".matches(regex6));
        System.out.println("a".matches(regex6));
        System.out.println("------匹配[a-zA-Z]中任意一个字符-------");

        String regex7 = "[a-zA-Z]";
        System.out.println("b".matches(regex7));
        System.out.println("T".matches(regex7));
        System.out.println("9".matches(regex7));
        System.out.println("------匹配[0-9]中任意一个字符-------");
        String regex8 = "[0-9]";
        System.out.println("4".matches(regex8));
        System.out.println("2".matches(regex8));
        System.out.println("10".matches(regex8));

    }

}
```

注意:匹配的都是单个字符

●14.04_常见对象(数量词)

- A:Greedy 数量词
  - X? ： X 出现一次或零次
  - X* ： X 出现零次或多次
  - X+ ： X 出现一次或多次
  - X{n} ： X 出现恰好 n 次
  - X{n,} ：X 出现至少 n 次
  - X{n,m} ： X 出现至少 n 次，但是不超过 m 次

```
package com.sxt.homework;
public class Demo {
    public static void main(String[] args) {

      /*
          X 代表正则表达式
            ▪ X? ：X 出现一次或零次
            ▪ X* ： X 出现零次或多次
            ▪ X+ ： X 出现一次或多次
            ▪ X{n} ： X 出现恰好 n 次
            ▪ X{n,} ：X 出现至少 n 次
            ▪ X{n,m} ： X 出现至少 n 次，但是不超过 m 次
       */
        System.out.println("--------?--------");
        String regex1 = "[abc]?";//表示a 或b 或c 中任意一个出现一次或0 次
      System.out.println("a".matches(regex1));
        System.out.println("c".matches(regex1));
        System.out.println("d".matches(regex1));
        System.out.println("-------*-----------");
        String regex2 = "[^abc]*";//除了abc 以外的任意字符出现0 次或多次
      System.out.println("defe".matches(regex2));
        String regex3 = "[0-9]*";//0~9 之间的任意一个数出现0 次或多次
      System.out.println("324234".matches(regex3));
        System.out.println("ab".matches(regex3));
        System.out.println("-----+------");
        String regex4 = "[0-9]+";//0~9 之间的任意一个数出现,至少出现一次
      System.out.println("0989ad".matches(regex4));
        System.out.println("0989".matches(regex4));
        System.out.println("------{n}-------");
        String regex5 = "[a-zA-z]{4}";//表示a-zA-z 之间的字母出现正好四次
      System.out.println("abcd".matches(regex5));
        System.out.println("abcde".matches(regex5));
        System.out.println("abc1".matches(regex5));

        System.out.println("-----{n,}-------");
        String regex6 = "[nihao]{2,}";//表示nihao 中的任意一个字母最少出现2 次,最多无限次
      System.out.println("nihaowo hen hao ".matches(regex6));
        System.out.println("ninihaha".matches(regex6));
```

```
        System.out.println("-----{n,m}-------");
        String regex7 = "[ilovey]{2,5}";//表示 ilovey 中的任意一个字母最少出现 2 次,最多 5 次
       System.out.println("iiloveloveyou".matches(regex7));//false
        System.out.println("iiloveloveyoyo".matches(regex7));//false
        System.out.println("iilll".matches(regex7));//true
        System.out.println("iillli".matches(regex7));//false
        System.out.println("ilove".matches(regex7));//true
    }
}
```

●常见对象(正则表达式的分割功能)

- A:正则表达式的分割功能
  - String 类的功能： public String[] split(String regex)
- B:案例演示
  - 正则表达式的分割功能

```
String s = "我 a 爱 2 大 B 长 9 沙";
        //要求输出:我爱大长沙
```

```
package com.sxt.homework;
public class Demo {
    public static void main(String[] args) {

        String s = "我 a 爱 2 大 B 长 9 沙";
        //要求输出:我爱大长沙
       //方法一:
         String[]arr  =s.split("\\w");//     \\w 表示匹配 a-zA-Z0-9 之间的任意字符
        for (int i = 0; i <arr.length ; i++) {
            System.out.print(arr[i]);
        }
        //方法二:
         System.out.println();
        String newS = s.replaceAll("\\w","");
        System.out.println(newS);

    }

}
```

●常见对象(把给定字符串中的数字排序)

- A:案例演示
  - 需求：我有如下一个字符串:”91 27 46 38 50”， 请写代码实现最终输出结果是：”27 38 46 50 91”

```java
package com.sxt.homework;


import java.util.Arrays;

public class Demo {
    public static void main(String[] args) {
        String s = " 91 27 46 38 50";
        String s1 = s.trim();
        String[]arr = s1.split(" ");
        Arrays.sort(arr);
        for (String str:arr) {
            System.out.print(str+" ");
        }

    }

}
```

●常见对象(正则表达式的替换功能)

- A:正则表达式的替换功能
  - String 类的功能：public String replaceAll(String regex,String replacement)
- B:案例演示
  - 正则表达式的替换功能

  需求：用正则表达式去掉字符串"我爱 23 长沙 576ab 星 ec 城"里面的数字和字母

```java
String s="我爱23长沙576ab星ec城";
String r1="\\d";
String r2="\\w";
String s2=s.replaceAll(r1, replacement: "");
System.out.println(s2);
String s3=s2.replaceAll(r2, replacement: "");
System.out.println(s3);
```

●常见对象(正则表达式的分组功能)(选学)

- A:正则表达式的分组功能
  - 捕获组可以通过从左到右计算其开括号来编号。例如，在表达式 ((A)(B(C))) 中，存在四个这样的组：
-

1 ((A)(B(C)))
2 (A
3 (B(C))
4 (C)

组零始终代表整个表达式。

- B:案例演示

a:切割

需求：请按照叠词切割： "sdqqfgkkkhjppppkl"; 结果为: sd fg hj kl

```
package com.sxt.homework;
import java.util.Arrays;

public class Demo {
    public static void main(String[] args) {
        //匹配叠词的正则表达式,用分组
       String s1 = "快快乐乐";
        String regex1="(.)\\1(.)\\2";
        System.out.println(s1.matches(regex1));
        /*
           (.)代表一个字符作为一组整体; \\1 代表前面的组(是第一组)重新出现一次;
           (.)代表一个字符作为一组整体; \\2 代表前面的组(是第二组)重新出现一次;
          */
         String s2 = "高兴高兴";
        String regex2 = "(..)\\1";
        System.out.println(s2.matches(regex2));
        String s3 = "快乐啊快乐啊";
        String regex3 = "(...)\\1";
        System.out.println(s3.matches(regex3));



        String ss = "sdqqfgkkkhjppppkl";
        String regex4 = "(.)\\1+"; //任意字符作为一组重复出现一次或多次也就是叠词
       String[]arr  =ss.split(regex4);
        System.out.println(Arrays.toString(arr));

    }

}
```

b:替换

需求：我我....我...我.要...要要...要学....学学..学.编..编编.编.程.程.程..程

将字符串还原成:“我要学编程”。

```
package com.sxt.homework;
import java.util.Arrays;

public class Demo {
    public static void main(String[] args) {
        /*
         需求：我我....我...我.要...要要...要学....学学..学.编..编编.编.程.程.程..程
      将字符串还原成:“我要学编程”。
         */
```

```
        String s = "我我....我...我.要...要要...要学....学学..学.编..编编.编.程.程.程..程";
        String s2 = s.replaceAll("\\.",""); //把字符串中的.用空字符串替换
       System.out.println(s2);
        String regex = "(.)\\1+";
        String s3 = s2.replaceAll(regex,"$1");//$1 表示叠词中单个字符的本身
       System.out.println(s3);
    }
}
```

开发中对正则表达式的应用主要是匹配:电话号,身份证号,邮箱,网址

●常见对象(Pattern 和 Matcher 的概述)

- A:Pattern 和 Matcher 的概述
- B:模式和匹配器的典型调用顺序
  - 通过 JDK 提供的 API，查看 Pattern 类的说明
  - 典型的调用顺序是
  - Pattern p = Pattern.compile("a*b");
  - Matcher m = p.matcher("aaaaab");
  - boolean b = m.matches();

```
package com.sxt.homework;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Demo {
    public static void main(String[] args) {
        /*
          Pattern p = Pattern.compile("a*b");  获取到正则表达式
          Matcher m = p.matcher("aaaaab");     获取匹配器
          boolean b = m.matches();    能匹配 true  是否能匹配
          */
         //由正则表达式创建一个 Pattern 匹配器对象
       Pattern p = Pattern.compile("[abc]+");
        String s ="bbb";
        Matcher m = p.matcher(s);//让匹配器取匹配目标
       boolean result = m.matches();//返回匹配结果
       System.out.println(result);

        //上面的一堆,其实等同于:  s.matchers("[abc]+")

    }

}
```

●常见对象(正则表达式的获取功能)

- A:正则表达式的获取功能

- Pattern 和 Matcher 的结合使用(Matcher 类中 find()与 group())

```
package com.sxt.homework;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Demo {
    public static void main(String[] args) {
        /*
          Pattern p = Pattern.compile("a*b");
          Matcher m = p.matcher("aaaaab");
          boolean b = m.matches();
          */

         String s ="abc 我爱长沙 d 环境 ad 好天气";
        //要求提取 s 字符串中的字母
       String regex ="[abcd]+";
        Pattern p = Pattern.compile(regex);
        Matcher m = p.matcher(s);
      /*
        if(m.find()){  //如果 s 字符串中存在与 regex 匹配的内容,就返回 true
             String result = m.group();  //提取匹配上的内容
             System.out.println(result);
         }
         if(m.find()){  //如果 s 字符串中存在与 regex 匹配的内容,就返回 true
             String result = m.group();  //把匹配上的内容提取出来
             System.out.println(result);
         }*/
         while(m.find()){
            System.out.println(m.group());
        }
    }

}
```

- B:案例演示 1：
  - 需求：把一个字符串中的手机号码获取出来
    字符串为: "客户登记的手机号为：13647583392,曾用手机号 18547389921,第三个手机号 15598765432"

案例演示 2：取出下面字符串中的所有的邮箱：
"客户登记的邮箱为 13647@sina.com.cn,备用邮箱 abd89_91@abc.net,第三个邮箱 aa-b21@345.cn，第四个邮箱 xidada@vip.qq.com 第五的个邮箱 abdc@163.com"

```
package com.sxt.homework;
import java.util.Arrays;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Demo {
```

```
    public static void main(String[] args) {

        String s ="客户登记的手机号为：13647583392,曾用手机号 18547389921,第三个手机号
15598765432";
        String regex = "1[3-9]\\d{9}";
        Pattern p = Pattern.compile(regex);
        Matcher m  =p.matcher(s);
        while(m.find()){
            System.out.println(m.group());
        }
    }
}
```

邮箱的正则表达式:
网上查的样子:
email.matches(″^[a-z0-9A-Z]+[- | a-z0-9A-Z . _]+@([a-z0-9A-Z]+(-[a-z0-9A-Z]+)?\\.)+[a-z]{2,}$″)
这是在 javascript 中的正则表达式的格式

java 中使用这部分就可以
String regex =”[a-z0-9A-Z]+[- | a-z0-9A-Z . _]+@([a-z0-9A-Z]+(-[a-z0-9A-Z]+)?\\.)+[a-z]{2,}”

●常见对象(System 类的概述和方法使用)

- A:System 类的概述
  - System 类包含一些有用的类字段和方法。它不能被实例化。
- B:成员方法
  - public static void gc()
  - public static void exit(int status)
  - public static long currentTimeMillis()

- C:案例演示
  - System 类的成员方法使用

```
package com.sxt.homework;

public class Demo2 {
    public static void main(String[] args) {
        long currentTime = System.currentTimeMillis();
        System.out.println(currentTime);//返回当前时间的毫秒值
//         System.exit(0);//正常退出
//         System.exit(-1);//非正常退出
       for (int i = 0; i <200 ; i++) {
            new Person();
            System.gc();//通知垃圾回收机制,可以回收垃圾了
       }
    }
}
```

```
class Person{
    @Override
    protected void finalize() throws Throwable {
        System.out.println("垃圾正在回收");
    }
}
```

●常见对象(BigInteger 类的概述和方法使用)

- A:BigInteger 的概述
  - 可以让超过 Integer 范围内的数据进行运算
- B:构造方法
  - public BigInteger(String val)
- C:成员方法
  - public BigInteger add(BigInteger val)
  - public BigInteger subtract(BigInteger val)
  - public BigInteger multiply(BigInteger val)
  - public BigInteger divide(BigInteger val)
  - public BigInteger[] divideAndRemainder(BigInteger val)

```
package com.sxt.homework;
import java.math.BigInteger;

public class Demo2 {
    public static void main(String[] args) {
//        long a = 8979879879789798790790087L;//金融领域中 long 存储整数有些还是存不下
//        System.out.println(a);
         //解决?用 BigInteger
         BigInteger b1 = new BigInteger("798798798798798709867896986897698697 6");
        BigInteger b2 = new BigInteger("79878979797");
        BigInteger result1 =b1.add(b2);//加法计算
       System.out.println(result1);
        BigInteger result2 = b1.subtract(b2);//减法
       System.out.println(result2);
        BigInteger result3 = b1.divide(b2);//除法'
         System.out.println("除法:"+result3);
        //除法之后带小数部分
       BigInteger[] result33 = b1.divideAndRemainder(b2);
        System.out.println(result33.length);
        System.out.println("小数点之前:"+result33[0]+"-小数点之后:"+result33[1]);

        BigInteger result4 = b1.multiply(b2);//乘法
       System.out.println(result4);
    }
}
```

●常见对象(BigDecimal 类的概述和方法使用)

- A:BigDecimal 的概述
  - 由于在运算的时候，float 类型和 double 很容易丢失精度，演示案例。

    - 所以，为了能精确的表示、计算浮点数，Java 提供了 BigDecimal
    - 不可变的、任意精度的有符号十进制数。
- B:构造方法
    - public BigDecimal(String val)
- C:成员方法
    - public BigDecimal add(BigDecimal augend)
    - public BigDecimal subtract(BigDecimal subtrahend)
    - public BigDecimal multiply(BigDecimal multiplicand)
    - public BigDecimal divide(BigDecimal divisor)
- D:案例演示
    - BigDecimal 类的构造方法和成员方法使用

```java
package com.sxt.homework;

import java.math.BigDecimal;
import java.math.BigInteger;

public class Demo2 {
    public static void main(String[] args) {
        System.out.println(2.0-1.1);
        /*
        输出的结果是:0.8999999999999999
        浮点数在运算会存在误差,结果无限接近真实结果,但是就差那么一点点
       为了解决这个精确度的问题使用:BigDecimal
         */
        BigDecimal d1 = new BigDecimal("2.0");
        BigDecimal d2 = new BigDecimal("1.1");
        BigDecimal result1 = d1.subtract(d2);
        System.out.println(result1);


    }
}
```

```java
package com.sxt.homework;

import java.math.BigDecimal;
import java.math.BigInteger;

public class Demo2 {
    public static void main(String[] args) {
        System.out.println(2.0-1.1);
        /*
        输出的结果是:0.8999999999999999
        浮点数在运算会存在误差,结果会无限接近真实结果,但是就差那么一点点
       为了解决这个精确度的问题使用:BigDecimal
         */
        BigDecimal d1 = new BigDecimal("78979434348609867 86");
        BigDecimal d2 = new BigDecimal("2");
        //减法
```

```
        BigDecimal result1 = d1.subtract(d2);
          System.out.println("减法:"+result1);

          BigDecimal result2 = d1.add(d2);
          System.out.println("加法:"+result2);

          BigDecimal result3 = d1.multiply(d2);
          System.out.println("乘法:"+result3);

          BigDecimal result4 = d1.divide(d2);//因为存在相除之后可能出现无限循环的结果,所以会抛异常,除的尽就不抛异常
        System.out.println("除法:"+result4);
    }
}
```

●常见对象(Date 类的概述和方法使用)(掌握)

- A:Date 类的概述
  - 类 Date 表示特定的瞬间，精确到毫秒。
- B:构造方法
  - public Date()
  - public Date(long date)

```
package com.sxt.homework;
import java.util.Date;

public class Demo2 {
    public static void main(String[] args) {
        Date d = new Date();//有日期和时分秒的日期类
       System.out.println(d);//Date 类中重写了Object 的toString 方法



       long time = System.currentTimeMillis();
        Date d2 = new Date(time);
        System.out.println(d);
        System.out.println(d2.after(d));
        System.out.println(d2.before(d));
        System.out.println(d.getDay());//已过时,但可以用,不建议用
       System.out.println(d.getTime());

        //java.sql.Date
        //已经导入了import java.util.Date 包,那么同名类得使用全路径名
       java.sql.Date dd = new java.sql.Date(System.currentTimeMillis());//只有日期的类
       System.out.println(dd);



    }
}
```

●常见对象(SimpleDateFormat 类实现日期和字符串的相互转换)(掌握)

- A:DateFormat 类的概述
  - DateFormat 是日期/时间格式化子类的抽象类，它以与语言无关的方式格式化并解析日期或时间。是抽象类，所以使用其子类 SimpleDateFormat
- B:SimpleDateFormat 构造方法
  - public SimpleDateFormat()
  - public SimpleDateFormat(String pattern)

- C:成员方法
  - public final String format(Date date) //把 Date 对象转成字符串的表示形式
  - public Date parse(String source) //把字符串的时间格式转成 Date 对象

```
package com.sxt.fuxi;

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

public class Demo4 {
    public static void main(String[] args) throws ParseException {
        SimpleDateFormat sdf = new SimpleDateFormat();
        System.out.println(sdf);
        Date d = new Date();
        System.out.println(d);//  Mon May 11 09:05:41 CST 2020
         String s = sdf.format(d);  //转换成符合当前语言使用环境的日期格式
       System.out.println(s);  //   2020/5/11 上午9:06
         //
         SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");//指定时间格式创建SimpleDateFormat
//         Date d2 = sdf2.parse("1999-10-20 10:10:20");//把一个与指定格式的字符串转成Date 对象
//         System.out.println(d2);
         String sdate =  sdf2.format(new Date());
        System.out.println(sdate);

        SimpleDateFormat sdf3 = new SimpleDateFormat("yyyy 年 MM 月 dd 日");
        Date sdate2 = sdf3.parse("2000 年 9 月 10 日");//传入的字符串日期格式一定要和"yyyy 年MM 月dd 日"一致
       System.out.println(sdate2);
    }
}
        /*
         掌握从Date 转化成自己想要的字符串时间格式
        以自定义的时间字符串格式,传承Date 对象
          */
```

●常见对象(你来到这个世界多少天案例)(掌握)

- A:案例演示
  - 需求：算一下你来到这个世界多少天?

```
package com.sxt.fuxi;

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

public class Demo5 {
    public static void main(String[] args) throws ParseException {
            /*
            算一下你来到这个世界多少天?
             */
            String sDate = "1999-4-20";
            //需求,得到这个生日的当天到当前时间的毫秒值
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
          Date d = sdf.parse(sDate);
          long time1 = d.getTime();//获取生日当天到1970 年1 月1 日0 时0 分0 秒
        System.out.println(time1);

          //当前毫秒值
        long time2 = System.currentTimeMillis();//当前时间到1970 年1 月1 日0 时0 分0 秒
        long birthDay = time2 - time1;
          int day =  (int)(birthDay/1000/3600/24);
          System.out.println("出生到现在的天数:"+day);

    }
  }
```

功能封装之后:

```
package com.sxt.fuxi;

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

public class Demo5 {
    public static void main(String[] args) throws ParseException {
        int n = getDay("2003-4-30");
        System.out.println(n);

    }
    //方法封装(把一个功能封装起来,可以服用代码)
    public static int getDay(String sDate) throws ParseException {
        //需求,得到这个生日的当天到当前时间的毫秒值
       SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        Date d = sdf.parse(sDate);
```

```
        long time1 = d.getTime();//获取生日当天到1970 年1 月1 日0 时0 分0 秒
        System.out.println(time1);

        //当前毫秒值
        long time2 = System.currentTimeMillis();//当前时间到1970 年1 月1 日0 时0 分0 秒
        long birthDay = time2 - time1;
        int day =  (int)(birthDay/1000/3600/24);
        return day;
    }
}
```

●常见对象(Calendar 类的概述和获取日期的方法)(掌握)

- A:Calendar 类的概述
  - 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。
- B:成员方法
  - public static Calendar getInstance()
  - public int get(int field)

●常见对象(Calendar 类的 add()和 set()方法)(掌握)

- A:成员方法
  - public void add(int field,int amount)
  - public final void set(int year,int month,int date)
- B:案例演示
  - Calendar 类的成员方法使用

```
package com.sxt.fuxi;
import java.util.Calendar;

public class Demo6 {
    public static void main(String[] args) {
        Calendar c = Calendar.getInstance();//这里是多态,父类引用指向子类对象,底层是Calendar c =new
GregorianCalendar(zone, aLocale);
        System.out.println(c);
        int date = c.get(Calendar.DATE);
        System.out.println(date);
        int month = c.get(Calendar.MONTH);
        System.out.println(month);  //在格里高利万年历中月份是从0~11 之间,0 代表一月
       int day = c.get(Calendar.DAY_OF_MONTH);
        System.out.println("修改之前:"+day);
        c.add(Calendar.DAY_OF_MONTH,5);//往当前的日加5 天
       System.out.println("修改之后:"+c.get(Calendar.DAY_OF_MONTH));
        c.add(Calendar.DAY_OF_MONTH,-2);
        System.out.println("修改之后 2:"+c.get(Calendar.DAY_OF_MONTH));

        //查香港回归日期的那天是星期几
```

```
        Calendar c2= Calendar.getInstance();
        c2.set(1997,6,1);  //把日历设置为指定的年月日 ,月份 0~11
        int week = c2.get(Calendar.DAY_OF_WEEK);//星期是 0~6 之间表示,0 代表星期一
        System.out.println(week);
    }
}
```

●常见对象(如何获取任意年份是平年还是闰年)(掌握)

- A:案例演示
  - 需求：键盘录入任意一个年份，判断该年是闰年（2 月最后一天是 29）还是平年(2 月最后一天是 28)

```
package com.sxt.fuxi;

import java.util.Calendar;
import java.util.Scanner;

public class Demo6 {
    public static void main(String[] args) {

        Scanner sc =new Scanner(System.in);
        System.out.println("输入一个年份:");
        String result= getMsg(sc.nextInt());
        System.out.println(result);

    }
        //键盘录入任意一个年份，判断该年是闰年（2 月最后一天是 29）还是平年(2 月最后一天是 28)
     public static String getMsg(int year){
        Calendar c = Calendar.getInstance();
        c.set(year,2,1);
        c.add(Calendar.DAY_OF_MONTH,-1);//3 月 1 日的前一天,是二月的最后一天
       int day = c.get(Calendar.DAY_OF_MONTH);
        if(day ==29){
            return "闰年";
        }else{
            return "平年";
        }
    }
}
```

重点掌握格式转换

●包装类型(自动拆箱自动装箱)

基本数据类型对应的对象类型

byte:Byte

short:Short

int :Integer

long:Long

float:Float
double:Double
boolean:Boolean
char : Character

```java
package com.sxt.fuxi;

public class Demo7 {
    public static void main(String[] args) {
        Integer in = new Integer(10);
        Integer in2 = new Integer("10");
        Integer in3 = 10;  //底层直接把基本数据 int 类型 10 自动包装成了 Integer 对象
        Double d = 45.6;//自动把基本数据类型包装成 Double 类对象
        System.out.println(d+3);//从结果可以看出,对象的类型自动被拆装成基本数据类型,默认调用了 doubleValue()方法
        System.out.println(d.doubleValue());

        int num = in3;//把对象直接赋值给了一个基本数据类型,是自动拆装
        System.out.println(num*10);
        Float f = 34.5f;

        Boolean b = true;

        Character c = '中';  //自动包装成对象,等同于 Character c = new Character('中');
        char c2 = c;    //自动拆装
        System.out.println(c2);
    }
}
```