目录

[Day15.    Java](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847045)

[练习1   查找图片文件](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847046)

[练习2   集合排序](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847047)

[练习3   文本文件中的日期字符串，排序后保存成新文件](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847048)

[1     线程](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847049)

[1.1       进程](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847050)

[1.2       线程](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847051)

[1.3       创建线程（两种方式）](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847052)

[1.4       继承 Thread](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847053)

[练习4   线程](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847054)

[1.5       实现 Runnable](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847055)

[练习5  Runnable](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847056)

[1.6       线程的状态](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847057)

[1.7       线程的方法](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847058)

[练习6   循环打印时间](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847059)

[练习7  join](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847060)

[2     作业](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day15.htm#_Toc6847061)

**Day15.  Java**

**练习1  查找图片文件**

项目：day1501\_查找图片文件

类：day1501.Test1

**package** day1501;

**import** java.io.File;

**import** java.io.FileFilter;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.LinkedList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** Test1 {

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

    System.***out***.println("输入文件夹：");

    String s = **new** Scanner(System.***in***).nextLine();

    File dir = **new** File(s);

**if**(! dir.isDirectory()) {

        System.***out***.println("请输入正确的文件夹");

**return**;

    }

    List<File> list = *f*(dir);

**for** (File f : list) {

        System.***out***.println(f.getAbsolutePath());

    }

}

**private** **static** List<File> f(File dir) {

    ArrayList<File> pic = **new** ArrayList<>();

    LinkedList<File> stack = **new** LinkedList<>();

    stack.push(dir);//初始状态，dir先压入

**while** (! stack.isEmpty()) {//栈中还有数据

        File f = stack.pop();//弹出

**if** (f.isFile()) { //f是文件

           //if(isPic(f)) {//f是图片文件

              pic.add(f);//图片添加到pic集合

           //}

        } **else** { //f是文件夹

           FileFilter filter = **new** FileFilter() {

              @Override

**public** **boolean** accept(File f) {

**if** (f.isDirectory()) {

**return** **true**;//接受文件夹

                 }

                 //也接受图片文件

                 String n = f.getName().toLowerCase();

**return** n.matches(".+\\.(jpg|png|gif|bmp)");

              }

           };

           File[] list = f.listFiles(filter);

**if**(list == **null**) {

**continue**;//列不出来不处理，继续弹出下一项

           }

**for** (File f2 : list) {//遍历，全部压入栈中

              stack.push(f2);

           }

        }

    }

**return** pic;

}

**private** **static** **boolean** isPic(File f) {

    // sdf.sdfs.sdf.jpg, a.JPG, .Jpg

    String n = f.getName().toLowerCase();

**return** n.matches(".+\\.(jpg|png|gif|bmp)");

}

}

**练习2  集合排序**

ArrayList，存放日期格式的字符串，排序

"2019-4-1"

"2019-4-10"

"2019-4-11"

"2019-4-12"

"2019-4-2"

"2019-4-20"

"2019-4-21"

"2019-4-22"

项目：day1502\_集合排序

类：day1502.Test1

**package** day1502;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.Date;

**public** **class** Test1 {

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

    ArrayList<String> list = **new** ArrayList<>();

    Collections.*addAll*(

           list,

           "2019-4-2","2019-4-10",

           //"dfhgfghsdfgsd",

           "2019-4-21","2019-4-30","2019-4-1",

           "2019-4-20","2019-4-12","2019-4-3",

           "2019-4-11","2019-4-22");

    System.***out***.println(list);

    //排序

    Collections.*sort*(list);

    System.***out***.println(list);

    //排序

    /\*

     \* sort()方法，可以使用一个接口，接入一个比较器对象

     \* sort(list, 比较器)

     \* sort()方法，排序运算过程中，

     \* 对数据比大小时，调用比较器来比较

     \*/

    Collections.*sort*(list, **new** Comparator<String>() {

        /\*

         \* o1和o2比较大小

         \* o1大，正数

         \* o1小，负数

         \* 相同，0

         \*

         \* 重写父类的方法，异常管道，不能比父类多

         \*/

        @Override

**public** **int** compare(String o1, String o2) {

           //日期字符串，解析成Date

           SimpleDateFormat f =

**new** SimpleDateFormat("yyyy-MM-dd");

**try** {

              Date a = f.parse(o1);

              Date b = f.parse(o2);

**return** a.compareTo(b);

           } **catch** (Exception e) {

              //包装成能抛的，再抛出

**throw** **new** IllegalArgumentException(e);

           }

        }

    });

    System.***out***.println(list);

}

}

**练习3  文本文件中的日期字符串，排序后保存成新文件**

d:/abc/f8

----------------

2019年4月2

2019年4月10

2019年4月21

2019年4月30

2019年4月1

2019年4月20

2019年4月12

2019年4月3

2019年4月11

2019年4月22

项目：day1503\_文本文件按行读写

类：day1503.Test1

**BufferedReader**，readLine()

读取一行字符串，读完再读，返回null

**PrintWriter**, println()

打印输出一行字符串，末尾补换行

**package** day1503;

**import** java.io.BufferedReader;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.InputStreamReader;

**import** java.io.OutputStreamWriter;

**import** java.io.PrintWriter;

**import** java.io.UnsupportedEncodingException;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.Date;

**import** java.util.List;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) **throws** Exception {

    //读取 f8 文件，所有日期字符串存入集合

    List<String> list = *f1*();

    //对list中的日期字符串排序

*f2*(list);

    //保存到 f9 文件

*f3*(list);

}

**private** **static** List<String> f1() **throws** Exception {

    /\*

     \* BR--ISR--FIS--f8

     \*

     \* FIS -  插在文件上

     \* ISR -  处理编码转换

     \* BR  -  按行读取

     \*/

    BufferedReader in =

**new** BufferedReader(//处理换行，按行读取

**new** InputStreamReader(//编码转换流

**new** FileInputStream("d:/abc/f8"),"GBK"));

    ArrayList<String> list = **new** ArrayList<>();

    String line = **null**;

**while**((line = in.readLine()) != **null**) {

        // trim() 去除两端的空白字符

        String s = line.trim();

**if** (s.length() == 0) {

**continue**;

        }

        list.add(s);

    }

    in.close();

    System.***out***.println(list);

**return** list;

}

**private** **static** **void** f2(List<String> list) {

    Collections.*sort*(list, **new** Comparator<String>() {

        @Override

**public** **int** compare(String o1, String o2) {

           SimpleDateFormat f =

**new** SimpleDateFormat("yyyy年MM月dd日");

**try** {

              Date a = f.parse(o1);

              Date b = f.parse(o2);

**return** a.compareTo(b);

           } **catch** (Exception e) {

**throw** **new** IllegalArgumentException(e);

           }

        }

    });

    System.***out***.println(list);

}

**private** **static** **void** f3(List<String> list) **throws** Exception {

    /\*

     \* PW--OSW--FOS--f9

     \*

     \* FOS    - 插在文件上

     \* OSW    - 编码转换

     \* PW     - println()自动加换行

     \*/

    PrintWriter out =

**new** PrintWriter(

**new** OutputStreamWriter(

**new** FileOutputStream("d:/abc/f9"),"GBK"));

**for** (String s : list) {

        out.println(s);

    }

    out.close();

}

}

**1   线程**

**1.1     进程**

在操作系统中，并行执行的任务

**1.2     线程**

在进程内部，并行执行的任务

**1.3     创建线程（两种方式）**

  继承 Thread

  实现 Runnable

**1.4     继承 Thread**

编写 Thread 的子类，并重写 run() 方法。启动之后，自动运行 run() 方法中的代码

**练习4  线程**

项目：day1504\_线程

类：day1504.Test1

**package** day1504;

**public** **class** Test1 {

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

    T1 t1 = **new** T1();

    T1 t2 = **new** T1();

    //启动线程

    //线程启动后，自动执行 run() 方法

    t1.start();

    t2.start();

    System.***out***.println("main");

}

**static** **class** T1 **extends** Thread {

    @Override

**public** **void** run() {

        //获取线程名称

        String n = getName();

        //打印1到1000

**for** (**int** i = 1; i <= 1000; i++) {

           System.***out***.println(n+" - "+i);

        }

    }

}

}

**1.5     实现 Runnable**

实现 Runnable 接口，实现它的 run() 方法，Runnable 封装在线程中执行的代码，新建线程对象时，把Runnable对象放在线程内，启动

**练习5  Runnable**

Test2

**package** day1504;

**public** **class** Test2 {

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

    R1 r1 = **new** R1();

    Thread t1 = **new** Thread(r1);

    Thread t2 = **new** Thread(r1);

    //线程启动后，自动执行 r1.run()

    t1.start();

    t2.start();

}

**static** **class** R1 **implements** Runnable {

    @Override

**public** **void** run() {

        //获得正在执行的线程实例

        Thread t = Thread.*currentThread*();

        //获得线程名

        String n = t.getName();

        //打印1到1000

**for** (**int** i = 1; i <= 1000; i++) {

           System.***out***.println(n+" - "+i);

        }

    }

}

}

**1.6     线程的状态**

**1.7     线程的方法**

**Thread.currentThread()**

获得正在执行的线程实例

**Thread.sleep(毫秒值)**

让正在执行的线程，暂停指定的毫秒值时长

**getName()**

**setName()**

线程名

**start()**

启动线程

**interrupt()**

打断线程的暂停状态

**join()**

当前线程暂停，等待被调用的线程结束

**练习6  循环打印时间**

Test3

**package** day1504;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** java.util.Scanner;

**public** **class** Test3 {

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

    T1 t1 = **new** T1();

    t1.start();

    Thread t2 = **new** Thread() {

        @Override

**public** **void** run() {

           System.***out***.println("按回车捅醒 t1");

**new** Scanner(System.***in***).nextLine();

           t1.interrupt();

        }

    };

    t2.start();

}

**static** **class** T1 **extends** Thread {

    @Override

**public** **void** run() {

        SimpleDateFormat f =

**new** SimpleDateFormat("HH:mm:ss");

**while**(**true**) {

           String s = f.format(**new** Date());

           System.***out***.println(s);

**try** {

              Thread.*sleep*(1000);

           } **catch** (InterruptedException e) {

              System.***out***.println("谁 TMD 捅醒了老子");

**break**;

           }

        }

    }

}

}

**练习7  join**

Test4

**package** day1504;

**public** **class** Test4 {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

    //1000万内，有多少个质数

    //2,3,5,7,11,13,17,19,23....

    System.***out***.println("\n--单线程-----------------");

*f1*();

    System.***out***.println("\n--5个线程-----------------");

*f2*();

}

**private** **static** **void** f1() **throws** InterruptedException {

**long** t = System.*currentTimeMillis*();

    T1 t1 = **new** T1(0, 10000000);

    t1.start();

    //main线程暂停等待 t1 的执行结果

    t1.join();

**int** c = t1.count;

    t = System.*currentTimeMillis*()-t;

    System.***out***.println("数量："+c);

    System.***out***.println("时间："+t);

}

**private** **static** **void** f2() **throws** InterruptedException {

**long** t = System.*currentTimeMillis*();

**int** n = 5;

**int** m = 10000000/n;

    T1[] a = **new** T1[n];

**for** (**int** i = 0; i < a.length; i++) {

        a[i] = **new** T1(m\*i, m\*(i+1));

        a[i].start();

    }

**int** sum = 0;

**for** (T1 t1 : a) {

        t1.join();

        sum += t1.count;

    }

    t = System.*currentTimeMillis*()-t;

    System.***out***.println("数量："+sum);

    System.***out***.println("时间："+t);

}

**static** **class** T1 **extends** Thread {

**int** from;

**int** to;

**int** count;//计数变量

**public** T1(**int** from, **int** to) {

**if**(from<3) {

           from = 3;

           count = 1;

        }

**this**.from = from;

**this**.to = to;

    }

    @Override

**public** **void** run() {

**for** (**int** i = from; i < to; i++) {

**if** (isPrime(i)) {//判断i是否是质数

              count++;

           }

        }

    }

**private** **boolean** isPrime(**int** i) {

        /\*

         \* 从2到 (i开方+1)

         \* 找有能把i整除的，i不是质数

         \* 都不能吧i整除，i是质数

         \*/

**double** d = Math.*sqrt*(i) + 1;

**for** (**int** j = 2; j < d; j++) {

**if** (i%j == 0) {

**return** **false**;

           }

        }

**return** **true**;

    }

}

}

**2   作业**

  重写

  day1502\_集合排序

  集合

  Collections工具类

  匿名内部类

  比较器

  重写时,子类异常管道并不能比父类多

  模仿day1501\_查找图片文件

  用栈运算方式，来计算文件夹的总大小