黑马入学考试题

黑马入学考试题

1、取出一个字符串中字母出现的次数

取出一个字符串中字母出现的次数。如：字符串："abcdekka27qoq" ，输出格式为：a(2)b(1)k(2)...

```
package com.itheima;

import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

/**
 * 第一题：取出一个字符串中字母出现的次数。如：字符串："abcdekka27qoq" ，输出格式为：a(2)b(1)k(2)...
 * @author
 *
 */
public class Test1 {
	public static  void main(String[] args){
	 String str = "abcdekka27qoq";
	 String char_count = getCharCount(str);
	 System.out.println(char_count);
	}
	public static String getCharCount(String str) {
		char[] chs = str.toCharArray();		//将字符串转成字符数组
		Map<Character,Integer> map = new TreeMap<Character,Integer>();  //定义 map 集合
		for (int i = 0; i < chs.length; i++) {	//遍历字符数组，获取每一个字母
				if(!(chs[i] >= 'a' && chs[i] <= 'z')){	//过滤掉非字母字符
				if(!(chs[i] >= 'A' && chs[i] <= 'Z')){
					continue;
				 }
			}
			Integer value = map.get(chs[i]);	//将遍历到的字母作为键去查表，获取值
			int count = 0;		//用于记录次数
			if(value != null){	//判断次数是否存在
				count = value;		//存在，就用 count 记录次数
```

```
                }
                count++;                    //次数不存在，就不记录，只对 count 自增变成 1
                map.put(chs[i], count);     //将字符和次数进行存储
            }
            return toString(map);
        }
        public static String toString(Map<Character, Integer> map) {
            StringBuilder sb = new StringBuilder();     //使用缓冲区
            Set<Character> keySet = map.keySet();       //将 map 集合转变成 set 集合
            for (Iterator<Character> it = keySet.iterator(); it.hasNext();) {     //通过迭代器，取出 map 集合中的元素
                Character key = it.next();
                Integer value = map.get(key);
                sb.append(key+"("+value+")");
            }
            return sb.toString();
        }
}
```

2、假如我们在开发一个系统时需要对员工进行建模

假如我们在开发一个系统时需要对员工进行建模，【员工】包含 3 个属性：姓名、工号以及工资。 【经理】也是员工，除了含有员工的属性外，另为还有一个奖金属性。请使用继承的思想设计出员工类和经理类。要求类中提供必要的方法进行属性访问。

```
package demo;
/**
 * 2、 假如我们在开发一个系统时需要对员工进行建模，员工包含 3 个属性：姓名、工号以及工资。
 * 经理也是员工，除了含有员工的属性外，另为还有一个奖金属性。
 * 请使用继承的思想设计出员工类和经理类。要求类中提供必要的方法进行属性访问。
 *
 * 涉及知识点：
 * 继承相关内容
 *
 * 思路：
 * 1、首先明确什么是子类什么是父类，明确之间的继承关系
 * 2、父类已经拥有的，子类通过 super 就可以获取到。
 * 3、对于父类来说，提供最基本的属性以及方法。对于子类来说，既然是继承，必须要有自己独特的方法和属性才存在意义。
 *
```

```
 * 面向对象的一个特征继承，很大程度上减少了代码量，增加了程序的功能扩展，并提高代码的复用性。
 * 让类与类之间产生了关系，有了这个关系才有了多态的特性。
 *
 * 注意：不要为了获取其他类的功能简化代码而继承；必须是类与类之间有所属关系才可以继承，父类中的内容子类都应具备才可继承。
 *
 * 继承机制：
 *  a、java 中只支持单继承，不支持多继承，因为多继承容易带来安全隐患，当多个父类中定义了相同功能时，子类不确定要使用哪一个。
 *  b、java 中的多继承机制是通过实现接口实现的，即 java 支持多实现。
 *  c、java 支持多层继承。
 * */

public class demo02 {
    public static void main(String[] args) {
        Employee em1 = new Employee("zhangsan", "20150001", 8800);
        Employee em2 = new Employee("lisi", "20150002", 8600);
        Employee em3 = new Employee("wangwu", "20150003", 8900);
        Manager ma = new Manager("tianqi", "20151001", 8800,12000);
        System.out.println(em1);
        System.out.println(em2);
        System.out.println(em3);
        System.out.println(ma);
    }
}
class Employee{       //经理也是员工的一种，所以员工是父类
    private String name;
    private String id;
    private double salary;
    public Employee(String name,String id,double salary) {
        this.name = name;
        this.id = id;
        this.salary = salary;
    }
    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    @Override
    public String toString() {
        return "姓名："+name+" 工号："+id+" 工资："+salary;
    }
}

class Manager extends Employee{    //经理类是子类，继承员工类，比员工类多一个奖金属性，相应的方法也要做对应改变
    private double bonus;
    public Manager(String name, String id, double salary,double bonus) {
        super(name, id, salary);
        this.bonus = bonus;
    }
    public double getBonus() {
        return bonus;
    }
    public void setBonus(double bonus) {
        this.bonus = bonus;
    }
    @Override
    public String toString() {
        return super.toString()+" 奖金："+bonus;
    }
```

}

3、 ArrayList<Integer> list = new ArrayList<Integer>();

ArrayList<Integer> list = new ArrayList<Integer>();在这个泛型为 Integer 的 ArrayList 中存放一个 String 类型的对象。

```
package demo;

import java.lang.reflect.Method;
import java.util.ArrayList;

/**
 * 3、 ArrayList<Integer> list = new ArrayList<Integer>();
 * 在这个泛型为 Integer 的 ArrayList 中存放一个 String 类型的对象。
 *
 * 涉及知识点：
 * 1、泛型相关知识
 * 2、通过反射获取并调用对象的方法
 *
 * 思路:
 * 1、无法向泛型为 Integer 的 ArrayList 中添加一个 String 类型对象，因为 ArrayList 只接受 Integer 类型。
 * 2、想要接受不同泛型的数据，那就要靠反射跳过泛型，调用 ArrayList 类中的 add 方法的接收 String 类型对象。
 * 3、具体先通过 list 对象获取类，再反射获取 add 方法，通过反射添加数据
 *
 * 总结:
 * 1、泛型，就是在方法上对于接受的类型进行了一个限定，JDK 1.5 版本以后出现的新特性，用于解决安全问题，是一个安全机制。
 *  a、将运行时期出现的问题 ClassCastException，转移到了编译时期，方便于程序员解决问题，让运行时期问题减少，提高安全性。
 *  b、避免了强制转化的麻烦。
 * 2、要注意反射的对象被修饰的修饰符是什么：
 *  a、如果是 public 修饰的，可以直接 getMethod()；
 *  b、如果是默认的，就需要通过 getDeclaredMethod()；
 *  c、若是 private，必须通过暴力反射，就是要通过 getDeclaredMethod()，并设置 setAccessible(true)来进行设置权限。
 *
 * 思考:
 * 1、泛型规范了元素的存储，提高了安全性。可是通过反射，就可以轻易地打破这个规范。
```

```
 * 2、反射尽管让程序变得便捷，但也让程序变得不安全，因此要慎用反射。
 * */

public class demo03 {
    //用反射跳过泛型，调用 ArrayList 类中的 add 方法添加 String
    public static void main(String[] args) throws Exception{
        ArrayList<Integer> list = new ArrayList<Integer>();
        String str="I'm a String";
        Integer i = 7;

        Method m = list.getClass().getMethod("add", Object.class);//得到 add 方法对象

        //利用正常方式添加 Integer 类型的对象
        list.add(i);
        //利用反射的方式添加
        m.invoke(list, i);

        //正常方法没法添加泛型不符合的对象
        //list.add(str);
        //利用反射的方法则可以添加
        m.invoke(list, str);//通过反射来执行 list 的第一个方法，第一个是 list 对象，代表该对象的方法，第二个是方法参数

        for (Object obj : list) {
            System.out.println(obj);
        }
    }
}
```

4、有五个学生，每个学生有 3 门课（语文、数学、英语）的成绩

有五个学生，每个学生有 3 门课（语文、数学、英语）的成绩，写一个程序接收从键盘输入学生的信息，输入格式为：name,30,30,30（姓名，三门课成绩），然后把输入的学生信息按总分从高到低的顺序写入到一个名称"stu.txt"文件中。要求：stu.txt 文件的格式要比较直观，打开这个文件，就可以很清楚的看到学生的信息。

```
package demo;

import java.io.BufferedWriter;
import java.io.FileWriter;
```

```
import java.util.Scanner;
import java.util.TreeSet;


/**
 * 4、 有五个学生，每个学生有 3 门课（语文、数学、英语）的成绩，写一个程序接收从键盘输入学生的信息。
 * 输入格式为：name,30,30,30（姓名，三门课成绩），
 * 然后把输入的学生信息按总分从高到低的顺序写入到一个名称"stu.txt"文件中。
 * 要求 :stu.txt 文件的格式要比较直观 ,打开这个文件 ,就可以很清楚的看到学生的信息。
 *
 * 涉及知识点：
 * 1、IO 流的相关操作
 * 2、集合的运用
 *
 * 思路：
 * 1、首先要定义学生类，定义基本的属性与方法，考虑到要使用集合存储，复写好 hashCode，equals，compareTo 等方法。
 * 2、要求输入的信息按总分从高到低写入，所以选择 TreeSet 集合比较方便。
 * 3、输入信息，读取字符串，按照","分割，封装到对象，存储到集合，再遍历集合输出到文件即可。
*/

public class demo04 {
    public static void main(String[] args) throws Exception {
        TreeSet<Student> ts = getSet();
        writeToFile(ts);
    }
    //将集合数据写入到文件，接收一个集合
    public static void writeToFile(TreeSet<Student> ts) throws Exception {
        BufferedWriter bufw = new BufferedWriter(new FileWriter("stu.txt"));
        for (Student student : ts) {
            //System.out.println(student.toString());
            bufw.write(student.toString());
            bufw.newLine();
            bufw.flush();
        }
        bufw.close();
```

```
    }
    //从键盘输入信息，将信息包装成对象，将对象存储到集合中，返回一个集合
    public static TreeSet<Student> getSet(){
        TreeSet<Student> ts = new TreeSet<Student>();
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            String str = sc.nextLine();
            if(str.equals("over")){
                sc.close();
                break;
            }
            String[] s = str.split(",");
            ts.add(new
Student(s[0],Integer.parseInt(s[1]),Integer.parseInt(s[2]),Integer.parseInt(s[3])));
        }
        return ts;
    }
}
class Student implements Comparable<Student>{
    private String name;
    private int ch,math,en;
    private int sum;
    public Student(String name,int ch,int math,int en) {
        this.name = name;
        this.ch = ch;
        this.math = math;
        this.en = en;
        this.sum = ch+math+en;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getCh() {
        return ch;
    }
```

```
    public void setCh(int ch) {
        this.ch = ch;
    }
    public int getMath() {
        return math;
    }
    public void setMath(int math) {
        this.math = math;
    }
    public int getEn() {
        return en;
    }
    public void setEn(int en) {
        this.en = en;
    }
    public int getSum() {
        return sum;
    }
    @Override
    public int compareTo(Student s) {
        if(!(s instanceof Student))
            throw new RuntimeException("类型错误");
        int num = new Integer(this.sum).compareTo(new Integer(s.sum));
        if(num==0){
            return this.name.compareTo(s.name);
        }
        return num;
    }
    @Override
    public int hashCode() {
        return name.hashCode()+sum*66;
    }
    @Override
    public boolean equals(Object obj) {
        Student s = (Student)obj;
        return this.name.equals(s.name) && this.ch==s.ch && this.math==s.math
&& this.en==s.en;
    }
```

```
    @Override
    public String toString() {
        return "姓名："+name+" 语文："+ch+" 数学："+math+" 英语："+en+" 总分"+sum;
    }
}
```

5、定义一个文件输入流，调用 read(byte[] b)方法

定义一个文件输入流，调用 read(byte[] b)方法将 exercise.txt 文件中的所有内容打印出来(byte 数组的大小限制为 5)。

```
package com.itheima;

import java.io.FileInputStream;
import java.io.IOException;

/**
 * 第 5 题： 定义一个文件输入流，调用 read(byte[] b)方法将 exercise.txt 文件中的所有内容打印出来(byte 数组的大小限制为 5)。
 *
 * @author
 *
 *
 */
public class Test5 {
    public static void main(String[] args) {
        FileInputStream fr = null;
                ByteArrayOutputStream bos =null;
        try {
            fr = new FileInputStream("d:\\exercise.txt");
            byte[] buf = new byte[5];
            int len = 0;
            while ((len = fr.read(buf)) != -1) {
                bos.write(buf, 0, len);//ByteArrayOutputStream 会把读到的内容全部存储到一个字节数组缓冲区中
            }
                System.out.println(bos.toString());      //最后用 toString 方法转成字符串
        } catch (IOException e) {
            e.printStackTrace();
```

```
        } finally {
            if (fr != null) {
                try {
                    fr.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

6、自定义字符输入流的包装类

自定义字符输入流的包装类，通过这个包装类对底层字符输入流进行包装，让程序通过这个包装类读取某个文本文件（例如，一个 java 源文件）时，能够在读取的每行前面都加上有行号和冒号。

```
package demo;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

/**
 * 6、 自定义字符输入流的包装类，通过这个包装类对底层字符输入流进行包装，
 * 让程序通过这个包装类读取某个文本文件（例如，一个 java 源文件）时，能够在读取的每行前面都加上有行号和冒号。
 *
 * 涉及知识点：
 * 1、装饰设计模式的应用
 * 2、IO 流的使用
 *
 * 思路：
 * 1、题目要求定义字符输入流的包装类，达到增强读取方法的目的，在读取的每行前加上行号和冒号。
 * 2、参考装饰设计模式的相关方法，定义一个类，接收一个要包装的底层类的对象，对对象的方法进行增强。
 *
 *  装饰设计模式：
 *      当想要对已有的对象进行功能增强时，可以定义类，
```

```
 *      将已有的对象传入，基于已有的功能，并提供加强功能。
 *      那么自定义的类成为装饰类。
 *
 *      装饰类通常会通过构造方法接受被装饰的对象。
 *      并基于被装饰的对象的功能，提供更强的功能。
 *
 *      装饰模式比继承要灵活，避免了继承体系的臃肿，而且降低了类与类之间的关系。
 *      装饰类增强已有对象，具备的功能和已有的是相同的，只不过提供了更强的功能。

 * */

public class demo06 {
    public static void main(String[] args) throws Exception {
        BufferedReader bufr = new BufferedReader(new FileReader("exercise.txt"));
        MyBufferedReader mbufr = new MyBufferedReader(bufr);
        String line = null;
        while((line=mbufr.readLine())!=null){
            System.out.println(line);
        }
    }
}
//装饰类
class MyBufferedReader {
    private BufferedReader bufr = null;
    private int row = 0;//行号标记
    public MyBufferedReader(BufferedReader bufr) {//装饰类构造函数,接收一个需要
增强方法的对应的对象
        this.bufr = bufr;
    }
    /*
     要增强方法的具体实现

    首先明确这个方法的返回值类型以及接收参数等，一定要注意方法有没有结束标志
    */
    public String readLine() {
        row++;
        String str = "";
        try {
```

```
                str = bufr.readLine();
            } catch (IOException e) {
                e.printStackTrace();
            }
            if(str==null){
                return null;
            } else {
                return row+":"+str;
            }
        }
}
```

7、分析以下程序运行结果，说明原理。

分析以下程序运行结果，说明原理。(没有分析结果不得分)

```
package demo;

/**
 * 7、 分析以下程序运行结果，说明原理。(没有分析结果不得分)
 *
public class ThreadTest {
    public static void main(String args[]) {
        MyThread t = new MyThread();
        t.run();
        t.start();
        System.out.println("A");
    }
}
class MyThread extends Thread {
    public void run() {
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("B");
    }
}
 * 运行结果是
 * B
```

```
 * A
 * B
 *
 * 分析：
 * 线程是进程中的一个独立的控制单元，一个进程可以有多个线程，线程与线程之间的运行是互相独立的。
 * 线程一旦开启，就会按照自己的执行顺序依次执行自己的代码语句。
 * 逐步分析一下本例的程序做了什么事：
 *  1、创建了一个 MyThread 对象
 *  2、主线程调用了 MyThread 类的 Run 方法，Run 方法延时 3 秒后打印"B"，语句执行结束，开始向下执行
 *  3、  a.主线程开启了 t 线程，t 线程开始独立执行，语句执行结束，开始向下执行
 *      b.t 线程开始执行其 Run 方法中的语句
 *  4、主线程打印"A"，到此主线程执行结束
 *  5、t 线程延时 3 秒后打印"B"，t 线程执行完毕
 *  6、程序运行结束
 *
 * 由于线程之间是独立的，所以主线程开启 t 线程之后，并不会等 t 线程结束，而直接向下运行。
 * t.run();是主线程调用的方法，执行结束才能向下执行
 * t.start();方法功能就是开启线程，线程一开启就继续向下执行
 * 可以做个实验，如果执行完 t.start()让主线程休眠 5 秒，那么结果就是 BBA 了
 */

public class demo07 {
    public static void main(String args[]) {
        MyThread t = new MyThread();//创建了一个 MyThread 对象
        t.run();//主线程调用 MyThread 类的 Run 方法，延时 3 秒后打印"B，语句执行结束，开始向下执行
        t.start();//主线程开启 t 线程，语句执行结束，开始向下执行
        System.out.println("A");//主线程打印"A"
    }
}
class MyThread extends Thread {
    public void run() {
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
```

```
            e.printStackTrace();
        }
        System.out.println("B");
    }
}
```

8、将字符串中进行反转。abcde --> edcba

将字符串中进行反转。abcde --> edcba

```
package demo;

/**
 * 8、 将字符串中进行反转。abcde --> edcba
 *
 * 涉及知识点：
 * 1、String 的相关操作
 * 2、StringBuffer 类的相关应用
 * */

public class demo08 {
    public static void main(String[] args) {
        String str = "abcde";
        System.out.println(str+" : "+reverse_01(str));
        System.out.println(str+" : "+reverse_02(str));
    }
    //方法一：将字符串转化成字符数组，然后首尾交换，完成反转
    public static String reverse_01(String str){
        char[] arr = str.toCharArray();//字符串转为字符数组
        for(int start=0,end=arr.length-1;start<end;start++,end--){//java 中取从头到
尾一般含头不含尾
            char temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
        }
        return new String(arr);
    }
    //方法二：使用 StringBuffer 中的 reverse 方法
    public static String reverse_02(String str){
        return new StringBuffer(str).reverse().toString();
    }
```

}

9、写一方法，打印等长的二维数组

写一方法，打印等长的二维数组，要求从 1 开始的自然数由方阵的最外圈向内螺旋方式地顺序排列。 如： n = 4 则打印：

```
package demo;

import java.util.Arrays;

/**
 * 9、 写一方法，打印等长的二维数组，要求从 1 开始的自然数由方阵的最外圈向内螺旋方式地顺序排列。 如： n = 4 则打印：
 *    1     2     3     4
 *    12    13    14    5
 *    11    16    15    6
 *    10    9     8     7
 *
 * 思路：
 * 1、观察方阵的规律，每一圈左上角的数字是最小的设为 min，每一圈有(n-1)*4 个数字，圈数设为 count
 * 2、可以做一个循环，给最外的一圈赋值，然后递归给里圈的赋值，直到 n=1 或 n=0 为止
 * */

public class demo09 {
    public static void main(String[] args) {
        int n = 4;
        int[][] arr = new int[n][n];
        int min = 1;
        int count = 0;
        setNum(n,arr,min,count);
        for(int i=0;i<n;i++){
            System.out.println(Arrays.toString(arr[i]));
        }
    }
    public static void setNum(int n,int[][] arr,int min,int count) {
        for(int i=0;i<=(n-1)*4;i++){
            if(n==0){//循环结束条件，n 为偶数情况
                break;
```

```
            }
            if(n==1){//循环结束条件，n 为奇数情况
                arr[count][count] = min;
                break;
            }

            //n-1 个为一组，一共赋值 4 组
            if(i<n-1){//第一组，向右赋值
                arr[count][count+i%(n-1)] = min++;
            } else if (i<(n-1)*2){//第二组，向下赋值
                arr[count+i%(n-1)][count+n-1] = min++;
            } else if (i<(n-1)*3) {//第三组，向左赋值
                arr[count+n-1][count+n-1-i%(n-1)] = min++;
            } else if (i<(n-1)*4){//第四组，向上赋值
                arr[count+n-1-i%(n-1)][count] = min++;
            } else {
                //当一圈赋值完毕，圈数加一，边长减二，递归调用
                setNum(n-2,arr,min,count++);
            }

        }
    }
}
```

10、28 人买可乐喝，3 个可乐瓶盖可以换一瓶可乐

28 人买可乐喝，3 个可乐瓶盖可以换一瓶可乐，那么要买多少瓶可乐，够 28 人喝？假如是 50 人，又需要买多少瓶可乐？

```
package demo;

/**
 * 10、 28 人买可乐喝，3 个可乐瓶盖可以换一瓶可乐，那么要买多少瓶可乐，够 28 人喝？
 *   假如是 50 人，又需要买多少瓶可乐？（需写出分析思路）
 * */
/*
    分析思路：
        要买够多少人喝，先一瓶一瓶的买，边买边换，直到可乐总数满足所有人喝为止
        可以设可乐总数为 sum，购买的瓶数用 count 来计数
        每购买一瓶 count 自加一，可乐总数 sum 也加一
        可乐总数 sum 达到三的倍数时，就可以用瓶盖再换一瓶，即总数 sum 加一
```

按以上规律做一个循环，当可乐总数 sum 满足所有人需要时跳出循环，此时的 count 值即为需要购买的瓶数

28 人共需要买：19 瓶
50 人共需要买：34 瓶
*/

```
public class demo10 {
    public static void main(String[] args) {
        int n=28;//人数
        System.out.println(n+"人共需要买："+getCount(n)+"瓶");
    }
    public static int getCount(int n){
        int sum = 0;//用于记录当前可乐总数
        int count = 0;//用于记录购买的可乐数
        while(sum<n){
            count++;
            sum++;
            if(sum%3==0){
                sum++;
            }
        }
        return count;
    }
}
```

11、编写一个类，在 main 方法中定义一个 Map 对象

编写一个类，在 main 方法中定义一个 Map 对象（采用泛型），加入若干个对象，然后遍历并打印出各元素的 key 和 value。

```
package com.itheima;

import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

/**
 * 第 11 题：编写一个类，在 main 方法中定义一个 Map 对象（采用泛型），加入若干个对象，然后遍历并打印出各元素的 key 和 value。
```

```
 * @author
 *
 *
 */
public class Test11 {
   public static void main(String[] args){
        Map<Integer,String> map = new HashMap<Integer,String>();
        map.put(27, "LinWuzhe");
        map.put(39, "TuoYonggang");
        map.put(41, "ChengCai");
        map.put(42, "XuSanduo");
        map.put(31, "TanWei");

        Set<Integer> keySet = map.keySet();
        for (Iterator<Integer> it = keySet.iterator(); it.hasNext();) {
           Integer key = it.next();
           String value = map.get(key);
           System.out.println(key+":"+value);

     }

   }
}
```

12、方法中的内部类能不能访问方法中的局部变量，为什么？

```
package demo;

import java.lang.reflect.Method;

/**
 * 12、 题目：方法中的内部类能不能访问方法中的局部变量，为什么？
 *
 * 涉及知识点：
 * 1、final 关键字的特点
 * 2、内部类的相关知识
 * 3、反射的相关知识
 *
 * 答：方法中的内部类不能直接访问方法中的局部变量；
 * 原因：内部类的生命周期和方法中的局部变量是不一样的，内部类是也是一个类，是存
```

储在堆中，也只有当对该类的引用消失时，内部类才会消亡。
 *　　而方法的局部变量是存储在栈内存中的，当调用结束时就会弹栈，就是在内存中这个属性就消失了。
 *　　内部类的生命周期超过了方法中局部变量的生命周期，内部类可能会调用到已经消失的局部变量，因此内部类不能直接访问方法中的局部变量。
 * 解决方法:局部变量前加修饰符 final，此时局部变量就会存在堆中，生命周期和内部类的生命周期是一样的，此时内部类就可以访问方法中的局部变量。
 *　　看下面的代码，方法中的内部类 Inner.class 调用方法中的局部变量 x，正常调用的时候内部类能够调用到方法
 * 中的局部变量，并将内部类对象 inner 返回，正常调用结束后，如果方法中的局部变量 x 没有被 final 关键字修饰，
 * x 则会被销毁，我们再通过反射的方式去调用 x 的时候则会发现找不到变量 x，如果局部变量 x 被 final 关键字修饰后，
 * 则 x 在方法运行结束后不会被销毁，而是常驻在内存中直到 JVM 退出，这样再通过反射调用的时候依然可以找到 x。
 *　　final 关键字特点：
 * a、可以修饰类、函数及变量；
 * b、被 final 修饰的类不可以被继承，为了避免被子类复写功能；
 * c、被 final 修饰的方法不可以被复写；
 * d、被 final 修饰的变量是一个常量只能赋值一次，既可以修饰成员变量也可修饰局部变量。
 * 反射中 getMethod 方法后面括号里第一项是方法的名称，后面是返回值的字节码类型
 * 调用后获得一个方法对象，该对象封装的方法是类上的一个方法，invoke 调用时需要传入调用该方法的对象，以及参数列表
 * */
```
public class demo12 {
    public static void main(String[] args) throws Exception {
        Outer outer = new Outer();//正常调用
        Object object = outer.outerfun();//获取内部类对象

        Method method = object.getClass().getMethod("innerfun");//反射调用内部类方法
        method.invoke(object);
    }
}

class Outer{
    public Object outerfun(){
```

```
        final int x = 5;
        class Inner{
            public void innerfun(){
                System.out.println(x);
            }
        }
        Inner inner = new Inner();
        inner.innerfun();
        return inner;
    }
}
```

13、定义一个交通灯枚举

定义一个交通灯枚举，包含红灯、绿灯、黄灯，需要有获得下一个灯的方法，例如：红灯获取下一个灯是绿灯，绿灯获取下一个灯是黄灯。

```
package demo;

/**
 * 13、 题目：定义一个交通灯枚举，包含红灯、绿灯、黄灯，需要有获得下一个灯的方法，
 *   例如：红灯获取下一个灯是绿灯，绿灯获取下一个灯是黄灯。
 *
 * 涉及知识点：
 * 枚举 enum 的相关应用
 *
 * 分析：
 * 1.定义一个交通灯类 Lamp，用 abstract 修饰
 * 2.构造方法私有化，目的是不允许程序员自己创建该类的对象。
 * 3.使用 new Lamp();枚举出红灯、绿灯、黄灯。
 * 4.创建获取下一个灯的方法 nextLight()，用 abstract 修饰。
 * 5.重载 toString()方法。
 * 6.在主函数中调用获取下一个交通灯的方法
 *
 * 枚举类型更规范，在函数形参定义为枚举类型的时候，只接受枚举类型的常量。比起普通的更安全
 * 枚举类型成员默认被 public static final 修饰
 * */

public class demo13 {
    public static void main(String[] args) {
```

```
        Lamp lamp = Lamp.YELLOW.nextLamp();
        System.out.println(lamp);
    }
    public enum Lamp {

        RED(30){public Lamp nextLamp(){return GREEN;}},    //构造函数接收下一个灯的字符串
        GREEN(45){public Lamp nextLamp(){return YELLOW;}},
        YELLOW(5){public Lamp nextLamp(){return RED;}};

        private int time;       //成员变量，每一个灯时间

        private Lamp(int time){          //构造函数必须私有化

            this.time = time;
        }

        public abstract Lamp nextLamp();    //下一个灯方法

    }
}
```

14、编写一个类，增加一个实例方法用于打印一条字符串

编写一个类,增加一个实例方法用于打印一条字符串。并使用反射手段创建该类的对象， 并调用该对象中的方法

```
package demo;

import java.lang.reflect.Constructor;
import java.lang.reflect.Method;

/**
 * 14、题目：编写一个类，增加一个实例方法用于打印一条字符串。并使用反射手段创建该类的对象，并调用该对象中的方法。
 *
 * 涉及知识点
 * 反射的相关内容
 *
 * 分析：
 * 1、首先要加载类，获取该类的字节码文件
```

```
 * 2、获取接收一个参数的构造函数
 * 3、创建该类的对象，用 newInstance 方法，传入参数
 * 4、获取类中名为 printString 的方法
 * 5、调用这个方法，传入之前创建的对象
 *
 * 创建字节码对象的三种方式：
 * 1、类名.class;
 * 2、对象.getClass();
 * 3、Class.forName(类全名字符串);
 *
 * 注意:
 * 对于不同权限，有不同的获取方式。
 * 有三种权限：
 * a、默认权限，默认需要加入 Declared
 * b、public，public 不需要加入 Declared
 * c、private，private 需要加入 Declared，还要设置 setAccessible(true)暴力反射
 * */

public class demo14 {
    public static void main(String[] args) throws Exception {
        Class<?> c = Class.forName("demo.ReflectDemo");//1、加载类
        Constructor<?> constructor = c.getConstructor(String.class);//2、获取构造
函数
        ReflectDemo rd = (ReflectDemo)constructor.newInstance("print
String");//3、创建对象
        Method method = c.getMethod("printString");//4、获取方法
        method.invoke(rd);//5、调用方法

        //直接用字节码文件创建对象
        Haha h = (Haha) Class.forName("demo.Haha").newInstance();
        h.ha();
    }
}
class Haha{
    public void ha(){
        System.out.println("haha");
    }
}
```

```
class ReflectDemo{
    private String str;//成员变量
    public ReflectDemo(String str) {//构造方法
        this.str = str;
    }
    public void printString() {//打印方法
        System.out.println(str);
    }
}
```

15、把当前文件中的所有文本拷贝，存入一个 txt 文件

把当前文件中的所有文本拷贝 ,存入一个 txt 文件 ,统计每个字符出现的次数并输出 ,例如：a： 21 次 b： 15 次 c: ： 15 次 把： 7 次...

```
package demo;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;
import java.util.TreeMap;

/**
 * 15、 题目：把当前文件中的所有文本拷贝，存入一个 txt 文件，统计每个字符出现的次数并输出，
 *  例如：
 *      a： 21 次
 *      b： 15 次
 *      c： 15 次
 *      把： 7 次
 *      当： 9 次
 *      前： 3 次
 *        ,: 30 次
 *
 * 涉及知识点：
 * 1、IO 流的应用，File，FileWriter，FileReader 等的使用
```

```
 * 2、集合的应用，Map 集合
 *
 * 思路：
 * 1、首先明确源文件和目标文件，这文本文件的拷贝，所以选择字符流来进行操作
 * 2、将当前目录下的文本文件封装成 File 对象，依次读取，并存储到同一个 txt 文件中
 * 3、统计各个字符出现的次数，这之间存在这映射关系，选择 Map 集合
 * */

public class demo15 {
    //声明一个静态的 TreeMap 集合用来存储字符与出现的次数
    public static TreeMap<Character,Integer> tm = new TreeMap<Character,
Integer>();

    public static void main(String[] args) throws Exception {
        String path = "E:\\Download\\eclipse\\workspace\\exam";
        getFiles(path);
        printMap(tm);
    }
    //接收一个文件夹路径，过滤不符合条件的文件，找出以".txt"结尾的文件
    public static void getFiles(String path) throws Exception{
        File dir = new File(path);
        File[] files = dir.listFiles();
        for (File file : files) {
            if(file.getName().endsWith(".txt")
&& !file.getName().equals("demo15.txt")){
                System.out.println(file.getName());
                txtIO(file);//符合条件，读入文本文件
            }
        }
    }
    //读入文本文件，将文本文件写入到指定的文本文件中
    public static void txtIO(File file) throws Exception{
        BufferedReader bufr = new BufferedReader(new FileReader(file));
        BufferedWriter bufw = new BufferedWriter(new
FileWriter("demo15.txt",true));
        String line = "";
        while((line=bufr.readLine())!=null){
            countChar(line);//逐句统计字符与字符出现的次数
```

```
                bufw.write(line);
                bufw.newLine();
                bufw.flush();
            }
            bufr.close();
            bufw.close();
        }
        //接收一个字符串，转化成字符数组，遍历数组统计字符与字符出现的次数，并存储到
TreeMap 集合中去
        public static void countChar(String line){
            char[] chs = line.toCharArray();
            for (char c : chs) {
                if(!tm.containsKey(c)){
                    tm.put(c, 1);
                } else {
                    tm.put(c, tm.get(c)+1);
                }
            }
        }
        //打印 Map 集合
        public static void printMap(Map<Character, Integer> m){
            Set<Entry<Character,Integer>> en = m.entrySet();
            for (Entry<Character, Integer> entry : en) {
                System.out.println(entry.getKey()+": "+entry.getValue()+"次");
            }
        }
}
```

16、编写程序，循环接收用户从键盘输入多个字符串

编写程序，循环接收用户从键盘输入多个字符串，直到输入“end”时循环结束，并将所有已输入的字符串按字典顺序倒序打印。

```
 package demo;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
```

```
/**
 * 16、 题目：编写程序，循环接收用户从键盘输入多个字符串，直到输入“end”时循环结束，
 *  并将所有已输入的字符串按字典顺序倒序打印。
 *
 * 涉及知识点：
 * 1、IO 流读取键盘录入
 * 2、集合相关应用
 *
 * 思路:
 * 1、在没有接收到"end"之前，输入的字符串需要存储，可以使用 ArrayList
 * 2、字符串要按照字典顺序倒序打印，可以利用集合工具类 Collections 进行相关的操作
 * */

public class demo16 {
    public static void main(String[] args) throws Exception {
        ArrayList<String> al = new ArrayList<String>();
        getArrayList(al);
        Comparator<String> c = Collections.reverseOrder();//获取反向的字符串比较器
        Collections.sort(al, c);//利用反向的比较器排序
        for (String string : al) {
            System.out.println(string);
        }
    }
    //获取键盘录入信息，存储到 ArrayList 集合中
    public static void getArrayList(ArrayList<String> al) throws Exception {
        BufferedReader bufr = new BufferedReader(new InputStreamReader(System.in));
        String line = null;
        while((line=bufr.readLine())!=null){
            if("over".equals(line))
                break;
            al.add(line);
        }
        bufr.close();
    }
}
```

17、一个 ArrayList 对象 aList 中存有若干个字符串元素

一个 ArrayList 对象 aList 中存有若干个字符串元素，现欲遍历该 ArrayList 对象，删除其中所有值为"abc"的字符串元素，请用代码实现。

```
package com.itheima;

import java.util.ArrayList;
/**
 * 第 17 题：一个 ArrayList 对象 aList 中存有若干个字符串元素，现欲遍历该 ArrayList 对象，删除其中所有值为"abc"的字符串元素，请用代码实现。
 *
 * @author   作者--谭威
 *
 */
public class Test17 {
   public static void main(String[] args){
        ArrayList<String> aList = new ArrayList<String>();
        aList.add("abc");
        aList.add("tw");
        aList.add("kjh");
        aList.add("abc");
        aList.add("sanduo");
        aList.add("abc");
        aList.add("wuzhe");
        aList.add("yuanlang");
        System.out.println(aList);

        String delstr = "abc";

        //普通 for 循环，通过角标删除
        for(int i = 0;i < aList.size();i++){
                if(aList.get(i).equals(delstr)){
                    aList.remove(i);
                }
        }
       System.out.println(aList);

   }
}
```

18、写一个 Properties 格式的配置文件，配置类的完整名称

1.写一个 Properties 格式的配置文件，配置类的完整名称。写一个程序，2.读取这个 Properties 配置文件，获得类的完整名称并加载这个类，用反射 的方式运行 run 方法。

```
package demo;

import java.io.FileInputStream;
import java.lang.reflect.Method;
import java.util.Properties;

/**
 * 18、题目：
 *   (1) 写一个 Properties 格式的配置文件，配置类的完整名称。
 *   (2) 写一个程序，读取这个 Properties 配置文件，获得类的完整名称并加载这个类,用反射 的方式运行 run 方法。
 *
 * 涉及知识点：
 *   1、配置文件 properties 的使用方法
 *   2、反射相关应用
 *
 * 思路:
 *   1、首先通过 FileInputStream 关联配置文件
 *   2、创建 Properties 对象，通过 load()方法将流数据读取到集合中
 *   3、Properties 是以键值对的形式存储的，通过 getProperty(key)获取对应的值
 *   4、获取配置文件中的类名，通过 forName 方法获取该类的字节码文件
 *   5、然后创建对象或直接获取方法，调用即可
 *
 * 总结:
 *   1、读取配置文件，其底层需要通过 IO 流
 *   2、Properties 集合通过 load 将集合关联的数据写入 properties 中
 *   3、通过 getProperty(key)获取对应的 value
 * */

public class demo18 {
    public static void main(String[] args) throws Exception {
        Properties prop = new Properties();
        prop.load(new FileInputStream("test.properties"));
        Class<?> c = Class.forName(prop.getProperty("className"));
        Method m = c.getMethod("run");
```

```
            m.invoke(c);
        }
}
class Runner{
        public static void run(){
            System.out.println("Running Man");
        }
}
```

19、单例模式获取设置配置信息

```
package demo;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Properties;

/**
 * 19、 单例模式获取设置配置信息
 *
 * 涉及知识点：
 * 1、单例设计模式的应用
 * 2、Properties 配置文件的相关操作
 * */

public class demo19 {
        public static void main(String[] args) throws Exception {
            PropTool pt = PropTool.getInstance();
            System.out.println(pt.getValue("className"));
            pt.setValue("name", "seven");
        }
}
class PropTool{
        private static PropTool pt = new PropTool();
        private PropTool() {
        }
        public static PropTool getInstance() {
            return pt;
        }
        public String getValue(String key) throws Exception{
```

```
            FileInputStream fis = new FileInputStream("test.properties");
            Properties prop = new Properties();
            prop.load(fis);
            return prop.getProperty(key);
        }
        public void setValue(String key,String value) throws Exception {
            FileInputStream fis = new FileInputStream("test.properties");
            Properties prop = new Properties();
            prop.load(fis);
            prop.setProperty(key, value);
            prop.store(new FileOutputStream("test.properties"), prop.toString());
        }
}
```

20、编写一个程序，获取 10 个 1 至 20 的随机数，要求随机数不能重复。

```
package demo;

import java.util.HashSet;
import java.util.Random;

/**
 * 20、编写一个程序，获取 10 个 1 至 20 的随机数，要求随机数不能重复。
 *
 * 思路:
 *   1、获取随机数，就用 Random
 *   2、随机数不能重复，就用 set 集合
 *   3、长度为 10，让 set 集合长度<=10
 *
 * 总结:
 *   1、不能重复，要装元素，元素无映射关系，就用 Set 集合；不需要排序用 HashSet，需要排序用 TreeSet
 *   2、满足以上条件，如果元素存在映射关系，就用 Map 集合
 * */

public class demo20 {
    public static void main(String[] args) {
        HashSet<Integer> hs = getRandom();
        System.out.println(hs.toString());
```

```
    }
    public static HashSet<Integer> getRandom() {
        Random r = new Random();
        HashSet<Integer> hs = new HashSet<Integer>();
        while(hs.size()<10){
            hs.add(r.nextInt(20)+1);
        }
        return hs;
    }
}
```

21、声明一个共享数组，起两个线程

声明一个共享数组，起两个线程，两个线程分别隔一段时间(可以写一个随机数)，给数组中添加数据，每一个线程为数组添加 3 个数据即可。

```
 package demo;

import java.util.Arrays;
import java.util.Random;

/**
 * 21、声明一个共享数组，起两个线程，两个线程分别隔一段时间(可以写一个随机数)，给数组中添加数据，
 *   每一个线程为数组添加 3 个数据即可。
 *
 * 涉及知识点：
 * 多线程之间的通讯
 *
 * 思路:
 *   1、数组是共享的，拥有数组的对象只能有一个，创建一个资源类封装一个数组，提供添加和查询的方法
 *   2、线程要有两个，Thread 需要创建两个，创建一个线程类继承或实现方法均可，run 方法调用资源类提供的方法添加元素
 *
 * 总结:
 *   1、共享的数据只能一份，那么在创建这个共享数据对象时候，就只能有一个对象
 *   2、实现了 Runnable 的类，只是一个协商提供线程运行的执行地
 *   3、共享数据是实实在在的，而线程是虚拟的
 *   4、wait 和 sleep 的用法：
 *        wait 是 Object 中的方法，使当前线程放弃 cup 执行权，同时可以等待时间，如
```

果我们想要让该线程放弃 cup 执行权(放弃锁)，那么就可以使用 wait 来控制；
 *　　而 sleep 是 Thread 中的方法，使线程等待一段时间，但是线程并没有释放 cup 使用权，如果只是让线程等待一段时间，那么就使用 sleep()来操作
 *　5、线程之所以要实现 Runnable 而不继承 Thread，是因为继承只支持单继承，限制了类的扩展性
 * */

```
public class demo21 {
	public static void main(String[] args) {
		ArrayResource r = new ArrayResource();
		AddToArray a = new AddToArray(r);
		new Thread(a).start();
		new Thread(a).start();
	}
}
//实现 Runnable 接口的类，复写了 run 方法，添加数据打印数组
class AddToArray implements Runnable{
	private ArrayResource r;
	public AddToArray(ArrayResource r) {
		this.r = r;
	}
	public void run() {
		for(int i=0;i<3;i++){//每个线程添加三个数据
			r.setInt();
		}
		r.printArray();//添加完毕打印数组
	}
}
//创建资源类，封装了数组，提供了添加数组元素和打印数组的方法
class ArrayResource{
	private int[] a = new int[6];
	private int count = 0;
	public synchronized void setInt(){
		Random r = new Random();
		a[count] = r.nextInt(10);
		System.out.println(Thread.currentThread().getName()+"　添　加　了　："
"+a[count]);
		count++;
```

```
            try {
                wait(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        public void printArray() {
            System.out.println(Arrays.toString(a));
        }
}
```

22、代码实现 c 盘某个文件夹复制到 D 盘中

代码实现 c 盘某个文件夹复制到 D 盘中，加上代码，加上注释，加上思路。

```
package demo;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;

/**
 * 22、 复制一个文件夹到另一个目录下，并将源文件夹下的所有以.jpg 结尾的文件改名为以.png 结尾
 *
 * 涉及知识点：
 * 1、File 的使用
 * 2、IO 流的使用
 * 3、递归的思想
 *
 * 思路：
 * 1、因为文件夹是一个层次目录，要复制多层文件夹，使用递归比较方便
 * 2、使用 File 的 listFiles 方法可以获得该文件目录下的文件列表，遍历这个列表，遇到目录就获取该目录的 listFiles，直到全部遍历结束
 * 3、文件的复制，遍历的同时，在目标目录下创建同名目录并复制普通文件
 *
 * 思考:
 * 该题的难点就在于新旧文件和文件夹的路径层次同步问题
 * 每当 listFiles 之后，新旧文件或文件夹就有了一个层次的差距，要明确旧文件的路径
```

```
 * */

public class demo22 {
    public static void main(String[] args) throws Exception {
        String oldpath = "E:\\demo";//将 demo 目录连同子目录子文件复制到目标路径
中
        File dir = new File(oldpath);

        String newpath = "D:\\de";//目标路径
        File newdir = new File(newpath);

        listDirs(dir,newdir);
        //renameToFile(dir);
    }
    public static void listDirs(File dir,File newdir) throws Exception {
        if (!dir.exists()) {
            System.out.println("原文件不存在");
            return ;
        }
        if (!newdir.exists() || !newdir.isDirectory()) {
            newdir.mkdirs();//创建多级目录要用 mkdirs()
        }
        File f = new File(newdir.getPath()+"\\"+dir.getName());
        f.mkdir();//在目标目录下创建一个与待复制文件夹同名的目录。

        //遍历目录下文件，碰到目录进入下层递归
        File[] files = dir.listFiles();
        for (File file : files) {
            if(file.isDirectory()){
                listDirs(file,f);
            } else {
                copyTo(file,f);
            }
        }
    }
    //文件复制方法
    public static void copyTo(File oldpath,File newpath) throws Exception {
        BufferedInputStream bfis = new BufferedInputStream(new
```

```
FileInputStream(oldpath));
        BufferedOutputStream bfos = new BufferedOutputStream(new
FileOutputStream(newpath.getPath()+"\\"+oldpath.getName()));
        int by = 0;
        while((by=bfis.read())!=-1){
            bfos.write(by);
        }
        bfis.close();
        bfos.close();
    }
    //改文件名方法
    public static void renameToFile(File dir) {
        File[] files = dir.listFiles();
        for (File file : files) {
            if(!dir.isDirectory()){
                renameToFile(file);
            } else {
                if(file.getName().endsWith(".png")){
                    System.out.println(file.getPath().replaceAll("\\.png", ".jpg"));
                    File newfile = new File(file.getPath().replaceAll("\\.png",
".jpg"));
                    file.renameTo(newfile);
                }
            }
        }
    }
}
```

23、写一个 ArrayList 类的代理

写一个 ArrayList 类的代理，实现和 ArrayList 类中完全相同的功能，并可以计算每个方法运行的时间。

```
package demo;

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
import java.lang.reflect.Proxy;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * 23、 写一个 ArrayList 类的代理，实现和 ArrayList 类中完全相同的功能，并可以计算每个方法运行的时间。
 * 思路:
 *        1.目标类是谁?ArrayList().给这个目标类定义一个代理.那么首先要确定,它的父类接口是谁?List(); 因为创建代理,需要和需要代理的类实现共同的接口.
 *        2.确定了父类接口之后,如何去创建目标的代理呢?通过 Proxy.newProxyInstance()来创建,参数有三个:1,父类(list)的类加载器;2,通过该对象字节码获取的接口.3.调用程序实现的接口子类.(通过内部类形式)
 *        3.对于调用程序实现的接口子类需要重写接口中的 invoke 方法(对象,方法,方法参数).然后对该方法进行一些设置,method.invoke(代理的对象,该对象的参数类型);这就是在代理内部对方法进行了设置.加入我们需求的一些信息.比如时间统计.
 *
 * 总结:
 *        1.代理类的出现,使程序可以自由地实现一些额外的系统功能.
 *        2.创建代理的方式.1,与目标类拥有共同的实现接口.2,目标类的子类也可以用作代理.(需要 CGLIB 库)
 *        (更多不太明白,需要进一步学习)
 *
 * */

public class demo23 {
    @SuppressWarnings({ "rawtypes", "unchecked" })
    public static void main(String[] args) {
        final ArrayList al = new ArrayList(); //必须 final 修饰.
        List proxy = (List)Proxy.newProxyInstance(
            List.class.getClassLoader(), //父类类加载器.
            ArrayList.class.getInterfaces(), //类的接口
            new InvocationHandler() {   //内部类.是一个实现了调用程序实现的接口子类
                @Override
                //实现了这个接口复写 invoke 方法. (对象,方法,方法参数)
                public Object invoke(Object proxy, Method method, Object[] args)throws Throwable {
                    long beginTime = System.currentTimeMillis();
                    Thread.sleep(10);
                    Object reVal = method.invoke(al,args);
                    long endTime = System.currentTimeMillis();
```

```
                        System.out.println(method.getName()+"  运  行了:"+(endTime-beginTime));
                        return reVal;
                    }
                }
            );
            proxy.add("黑马程序员");
            proxy.add("你好");
            proxy.add("程序员");
            System.out.println(proxy.toString());
        }
}
```

24、存在一个 javaBean,它包含以下几个属性

存在一个 javaBean,它包含以下几个属性，1Boolean/boolean

```
package demo;

import java.beans.BeanInfo;
import java.beans.Introspector;
import java.beans.PropertyDescriptor;
import java.lang.reflect.Method;

/**
 * 24、 存在一个 javabean,设置 string，Boolean，double，integer
 *  的默认初值为 www.itheima.com，true，0.01D，100
 *
 * 思路:
 * 1.创建一个类,该类是标准 javabean,并拥有一些自定义属性.
 * 2.获取到该类的字节码文件,并通过字节码创建该类对象.
 * 3.获取 BeanInfo 对象.通过 Introspector.getBeanInfo(类字节码)这样就将该类的信息封装到了 BeanInfo 对象中.
 * 4.通过 beanInfo.getPropertyDescriptors()返回一个数组,将信息存入数组中.
 * 5.遍历该 PropertyDescriptors 类型数组,并将数组元素通过各种方法获取到信息.
 * 6.用属性类型与提供的类型想匹配,然后赋值.
 *
 * 总结:
 * 1.javaBean 是一个协议类,也就是该类拥有一些功能,主要用于对目标类进行字段的操作,但是,必须要按照 javaBean 自己提供的一个标准,它才可以识别;
 * 2.javaBean 和反射有什么区别呢?javaBean 是把数据先封装到自己中,然后再通过自己的
```

方法,将这些信息提供出来.针对字段,并具有很好的普遍性.而反射可以获取到任何信息,也更具有针对性.

```
 *
 * 思考:
 * JDK 中提供了对 JavaBean 进行操作的一些 API，这套 API 就称为内省。
 * */

public class demo24 {
    @SuppressWarnings("rawtypes")
    public static void main(String[] args) throws Exception {
        Class clazz = Class.forName("cn.itheima.demo01.testBean");
        Object bean = clazz.newInstance();
        BeanInfo beanInfo = Introspector.getBeanInfo(clazz);
        PropertyDescriptor[] ppd = beanInfo.getPropertyDescriptors();
        for(PropertyDescriptor pro: ppd)
        {
            Object name = pro.getName();   //属性名称 str,in 等等
            Object type = pro.getPropertyType(); //属性类型 String,int 等等
            Method set = pro.getWriteMethod(); //set 方法.
            Method get = pro.getReadMethod(); //get 方法.
            if(!name.equals(".class"))
            {
                if(set != null)
                {
                    if(type == Boolean.class || type == Boolean.class)
                    {
                        set.invoke(bean, true);
                    }
                    if(type == String.class )
                    {
                        set.invoke(bean, "com.ithema.com");
                    }
                    if(type == Integer.class || type == int.class)
                    {
                        set.invoke(bean, 100);
                    }
                    if(type == Double.class || type == double.class)
                    {
```

```
                        set.invoke(bean, 0.01D);
                    }
                }
                if(get != null)
                {
                    System.out.println(type + " "
                            + name + "=" + get.invoke(bean,null) );
                }
            }

        }
    }
}

class testBean{
    private boolean b;
    private String str;
    private Integer i;
    private double d;
    public boolean isB() {
        return b;
    }
    public void setB(boolean b) {
        this.b = b;
    }
    public String getStr() {
        return str;
    }
    public void setStr(String str) {
        this.str = str;
    }
    public Integer getI() {
        return i;
    }
    public void setI(Integer i) {
        this.i = i;
    }
    public double getD() {
```

```
        return d;
    }
    public void setD(double d) {
        this.d = d;
    }
}
```

25、编写一个程序，它先将键盘上输入的一个字符串转换成十进制整数

编写一个程序，它先将键盘上输入的一个字符串转换成十进制整数，然后打印出这个十进制整数对应的二进制形式。这个程序要考虑输入的字符串不能转换成一个十进制整数的情况，并对转换失败的原因要区分出是数字太大，还是其中包含有非数字字符的情况。提示：十进制数转二进制数的方式是用这个数除以 2，余数就是二进制数的最低位，接着再用得到的商作为被除数去除以 2，这次得到的余数就是次低位，如此循环，直到被除数为 0 为止。其实，只要明白了打印出一个十进制数的每一位的方式(不断除以 10，得到的余数就分别是个位，十位，百位)，就很容易理解十进制数转二进制数的这种方式。

```
package com.itheima;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * 第 25 题：编写一个程序，它先将键盘上输入的一个字符串转换成十进制整数，然后打印出这个十进制整数对应的二进制形式。
 * 这个程序要考虑输入的字符串不能转换成一个十进制整数的情况，并对转换失败的原因要区分出是数字太大，还是其中包含有非数字字符的情况。
 * 提示：
 * 十进制数转二进制数的方式是用这个数除以 2，余数就是二进制数的最低位，接着再用得到的商作为被除数去除以 2，这次得到的余数就是次低位，如此循环，直到被除数为 0 为止。
 * 其实，只要明白了打印出一个十进制数的每一位的方式（不断除以 10，得到的余数就分别是个位，十位，百位），就很容易理解十进制数转二进制数的这种方式。
 *
 * @author
 *
 *
 */
public class Test25 {
```

```
    public static void main(String[] args) throws IOException{
        System.out.println("请输入一个整数字符串(最大值为<" +
Integer.MAX_VALUE + ">):");
        BufferedReader bufr = new BufferedReader(new
InputStreamReader(System.in));
        String line = bufr.readLine();
        bufr.close();

        Test9.ChangeToBinary(line);
    }

    public static void ChangeToBinary(String str) {
        int shang = 0;
        int yushu;
        StringBuilder sb = new StringBuilder();

        double d = 0;
        try {
            d = Double.parseDouble(str);
        } catch (Exception e) {
            throw new RuntimeException("包含有非数字字符");
        }
        if (d > Integer.MAX_VALUE) {
            System.out.println("数字太大");
        } else {
            shang = Integer.parseInt(str);
            while (shang > 0) {          //转二进制数
                yushu = shang % 2;
                sb.append(yushu);
                shang = shang / 2;
            }
        }
        System.out.println(sb.reverse());
    }
}
```

26、已知一个 int 数组，用冒泡排序法对数组中元素进行升序排列。

已知一个 int 数组，用冒泡排序法对数组中元素进行升序排列。（简单题也不放过）

```
package demo;
```

```
import java.util.Arrays;

/**
 * 26、 冒泡排序，选择排序
 * */

public class demo26 {
    public static void main(String[] args) {
        int[] arr = {5,0,2,9,6,7,3,4,1,8};
        bubbleSort(arr);
        selectionSort(arr);
    }
    //冒泡排序
    public static void bubbleSort(int[] arr) {
        int temp;
        for(int i=0;i<arr.length-1;i++){
            for(int j=0;j<arr.length-i-1;j++){
                if(arr[j]>arr[j+1]){
                    temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }
        System.out.println(Arrays.toString(arr));
    }
    //选择排序
    public static void selectionSort(int[] arr) {
        int temp,index;
        for(int i=0;i<arr.length-1;i++){
            index = i;
            for(int j=i+1;j<arr.length-1;j++){
                if(arr[index]>arr[j]){
                    index = j;
                }
            }
            temp = arr[i];
```

```
            arr[i] = arr[index];
            arr[index] = temp;
        }
        System.out.println(Arrays.toString(arr));
    }
}
```

27、自定义枚举 week 表示星期几

自定义枚举 week 表示星期几，要求每个枚举值都有 tolocaleString 方法。返回中文格式的星期几。

```
package demo;
/**
 * 27、 自定义枚举 week 表示星期几，要求每个枚举值都有 tolocaleString 方法。返回中文格式的星期几。
 * */
public class demo27 {
    public static void main(String[] args)throws Exception {
        System.out.println(WeekDay.MON.toLocaleString());
    }
    public enum WeekDay{
        MON{ public String toLocaleString(){ return "星期一"; }},
        TUE{ public String toLocaleString(){ return "星期二"; }},
        WEN{ public String toLocaleString(){ return "星期三"; }},
        THU{ public String toLocaleString(){ return "星期四"; }},
        FRI{
            public String toLocaleString() {
                return "星期五";
            }
        },
        SAT{
            public String toLocaleString() {
                return "星期六";
            }
        },
        SUN{
            public String toLocaleString() {
                return "星期日";
            }
        };
```

```
        public abstract String toLocaleString();
    }
}
```

28、编写 三个类 ticket，sealWindow,ticketSealCenter

编写 三个类 ticket，sealWindow,ticketSealCenter,代表票信息，售票窗口，售票中心，售票中心分配一定数量的票（多个窗口实现多线程售票）

```
package demo;

import java.util.Random;

/**
 * 28、 模拟一个火车站卖票程序,需求 5 个窗口,需要多线程完成.
 *
 * 涉及知识点：
 * 多线程的相关操作
 *
 * 思路：
 * 1、火车票是共享资源，多个窗口或者说是线程同时操作，共享火车票数这一数据
 * 2、假设火车站持有火车票资源，窗口来操作资源
 * */

public class demo28 {
    public static void main(String[] args) {
        TicketResource tr = new TicketResource();
        new Thread(new TicketWindow(tr)).start();
        new Thread(new TicketWindow(tr)).start();
        new Thread(new TicketWindow(tr)).start();
        new Thread(new TicketWindow(tr)).start();
        new Thread(new TicketWindow(tr)).start();
    }
}
class TicketWindow implements Runnable{      //售票窗口类，随机出售 1 到 5 张车票
    private TicketResource tr;
    public TicketWindow(TicketResource tr) {
        this.tr = tr;
    }
    Random r = new Random();
    @Override
```

```
    public void run() {
        while(tr.getTicket()>0){
            tr.sellTicket(r.nextInt(5)+1);
        }
    }
}
class TicketResource{  //车票资源类，封装了车票数，定义了查询余票张数的方法和票数减少的方法
    private int tickets = 100;
    public int getTicket() {
        return tickets;
    }
    public synchronized void sellTicket(int num) {
        if(this.tickets>=num){
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();}
            this.tickets = this.tickets - num;
            System.out.println(Thread.currentThread().getName()+"卖出去了"+num+"张票，剩余："+tickets);
        } else {
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println(Thread.currentThread().getName()+"余票不足");
        }
    }
}
```

29、编写程序，生成 5 个 1 至 10 之间的随机整数

编写程序，生成 5 个 1 至 10 之间的随机整数，存入一个 List 集合，编写方法对 List 集合进行排序（自定义排序算法，禁用 Collections.sort 方法和 TreeSet），然后遍历集合输出

```
package demo;

import java.util.ArrayList;
import java.util.Arrays;
```

```
import java.util.List;
import java.util.Random;

/**
 * 29、 编写程序，生成 5 个 1 至 10 之间的随机整数，存入一个 List 集合，编写方法对 List 集合进行排序
 * （自定义排序算法，禁用 Collections.sort 方法和 TreeSet），然后遍历集合输出。
 *
 * 涉及知识点：
 * 1、集合的基本应用
 * 2、排序算法
 *
 * 思路：
 * 1、用 Random 生成随机数，存入一个 List 集合
 * 2、自定义排序，冒泡选择都可
 * */

public class demo29 {
    public static void main(String[] args) {
        List<Integer> al = sortArray(getRandom());
        for (Integer integer : al) {
            System.out.println(integer);}
    }
    public static List<Integer> getRandom() {    //获取随机数存入集合
        ArrayList<Integer> al = new ArrayList<Integer>();
        Random r = new Random();
        for(int i=0;i<5;i++){
            al.add(r.nextInt(10)+1);
        }
        return al;
    }
    public static List<Integer> sortByBubble(List<Integer> list){   //冒泡排序
        Integer[] arr= list.toArray(new Integer[list.size()]);
        int temp;
        for(int i=0;i<arr.length-1;i++){
            for(int j=0;j<arr.length-i-1;j++){
                if(arr[j]>arr[j+1]){
                    temp = arr[j];
```

```
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;}
            }
        }
        return Arrays.asList(arr);
    }
}
```

30、编写一个程序，这个程序把一个整数数组中的每个元素用逗号连接成一个字符串

编写一个程序，这个程序把一个整数数组中的每个元素用逗号连接成一个字符串，例如，根据内容为[1][2][3]的数组形成内容为"1,2,3"的字符串。

```
package demo;

/**
 * 30、 编写一个程序，这个程序把一个整数数组中的每个元素用逗号连接成一个字符串，
 *  例如，根据内容为[1][2][3]的数组形成内容为"1,2,3"的字符串。
 *
 * 思路：
 * 1、接收一个数组，遍历，将每一个元素转换成字符串并加上逗号即可，判断为最后一个时不加逗号
 * 2、操作字符串，使用 StringBuilder 或 StringBuffer
 * */

public class demo30 {
    public static void main(String[] args) {
        int[] arr = {1,2,6,4,5,7,8,9,10};
        System.out.println(arrToString(arr));
    }
    private static String arrToString(int[] arr) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < arr.length; i++) {
            if(i!=arr.length-1)
                sb.append(arr[i]+",");
            else
                sb.append(arr[i]+"");
        }
        return sb.toString();
    }
```

}

31、金额转换，阿拉伯数字转换成中国传统形式。

金额转换，阿拉伯数字转换成中国传统形式。例如：101000001010 转换为壹仟零壹拾亿零壹仟零壹拾圆整

```
package demo;

import java.util.Scanner;

/**
 * 31、 金额转换，阿拉伯数字转换成中国传统形式。例如：101000001010 转换为壹仟零壹拾亿 零 壹仟零壹拾 圆整
 *
 * 涉及知识点：
 * 1、String 的相关操作
 * 2、正则表达式的应用
 *
 * 思路：
 * 1、观察规律 ,阿拉伯数字转化成汉字 ,每一个都有自己的单位 ,声明一个 StringBuilder，从第一位开始添加单位以及数字
 * 2、1001 壹仟零壹，1100 壹仟壹佰，1010 壹仟零壹拾，1101 壹仟壹佰零壹，可以看出只要两个非零数字中间有零就要写个零
 * 3、最后，做一些替换
 * */

public class demo31 {
    private static final char[] number = {'零','壹','贰','叁','肆','伍','陆','柒','捌','玖'};
    private static final char[] units = {'圆','拾','佰','仟','万','拾','佰','仟','亿','拾','佰','仟'};

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            long num = sc.nextLong();
            if(num==0){
                break;
            }
            String str = convert(num);
            System.out.println(str);
```

```
        }
        sc.close();
    }
    private static String convert(long money) throws Exception{
        StringBuilder sb = new StringBuilder();
        int index = 0;
        while(money!=0){
            sb.insert(0, units[index++]);
            sb.insert(0, number[(int)(money%10)]);
            money = money/10;
        }
        return sb.toString().replaceAll("零[仟佰拾]", "零").replaceAll("零+万", "万")
                .replaceAll("零+亿", "亿").replaceAll("亿万", "亿零")
                .replaceAll("零+", "零").replaceAll("零圆", "圆");
    }
}
```

32、编写一个程序，当用户输入一个目录时

编写一个程序，当用户输入一个目录时， 该程序能列出该目录下的所有子目录和文件

```
package demo;

import java.io.File;

/**
 * 32、编写一个程序，当用户输入一个目录时，该程序能列出该目录下的所有子目录和文件
 *
 * 涉及知识点：
 * IO 流的相关操作，File 类的使用
 *
 * 思路：
 * 传入一个路径，封装成 File 对象，调用 listFiles 方法，递归列出所有子目录和文件
 * */
public class demo32 {
    public static void main(String[] args) {
        //Scanner sc = new Scanner(System.in);
        //String path = sc.next();
        //sc.close();
        int level = 0;
```

```
        printFiles(new File("E:\\demo"),level);
    }
    public static void printFiles(File dir,int level) {
        if(!dir.exists()){
            System.out.println("文件不存在");
            return;
        }
        level++;
        File[] files = dir.listFiles();
        for (File file : files) {
            if(file.isDirectory()){
                for(int i=0;i<level;i++){
                    System.out.print("    ");
                }
                System.out.println("|---"+file.getName());
                printFiles(file,level);
            } else {
                for(int i=0;i<level;i++){
                    System.out.print("    ");
                }
                System.out.println("|---"+file.getName());
            }
        }
    }
}
```

33--我们要给每个字母配一个 1-26 之间的整数,……求最大完美度

```
package demo;

import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;
import java.util.Set;

/**
 * 33、 从键盘接受一个字符串,每个字母可自由匹配 1-26 值,求这个字符串所能达到的最大数(最完美度)
 *
 * 原理：
```

```
 * 要给每个字母配一个 1-26 之间的整数，出现次数越多分配的数字越大
 * 一个字符串的完美度等于它里面所有字母的完美度之和，且与字母大小写无关
 * 现在给定一个字符串，输出它的最大可能的完美度，例如：dad，将 26 分配给 d，25 分配给 a，这样整个字符串最大可能的完美度为 77

 * 思路:
 * 1、从键盘接受一个字符串，将字符串转化成字符数组，遍历，并将字符和字符出现的次数放入 map 集合中
 * 2、遍历 map 集合，获取 value 值存入一个数组中,并对数组进行排序
 * 3、倒着遍历，开始从 26 逐个递减匹配，计算完美度
 * */

public class demo33 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            String str = sc.next();
            if(str.equals("over")){
                break;
            }
            System.out.println(getMax(str));
        }
        sc.close();
    }
    public static int getMax(String str) {
        //将字符串转化成字符数组，遍历，并将字符和字符出现的次数放入 map 集合中
        HashMap<Character, Integer> hm = new HashMap<Character, Integer>();
        char[] chs = str.toCharArray();
        for (char c : chs) {
            if(hm.containsKey(c)){
                hm.put(c, hm.get(c)+1);
            } else {
                hm.put(c, 1);
            }
        }
        //遍历 map 集合，获取 value 值存入一个数组中
        int[] arr = new int[hm.size()];
        Set<Character> s = hm.keySet();
```

```
            int i = 0;
            for (Character character : s) {
                arr[i] = hm.get(character);
                i++;
            }
            //对数组进行排序
            Arrays.sort(arr);
            //倒着遍历，开始从 26 逐个递减匹配，计算完美度
            int max = 0,num = 26;
            for (int j=arr.length-1;j>=0;j--) {
                max = max+arr[j]*num;
                num--;
            }
            return max;
        }
}
```

34、用 TCP 协议写一个客户端和一个服务端，实现上传文件

```
package demo;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * 34、用 TCP 协议写一个客户端和一个服务端，实现上传文件
 * */

public class demo34 {

}
/*
```

服务端
步骤：
1、建立服务端的 Socket 服务，通过 ServerSocket()，并监听一个端口
2、获取连接过来的客户端对象，通过 accept()方法。此方法为阻塞方法，没有连接就会等待
3、客户端如果发来数据，那么服务端要接收文件数据并将文件存储到本地
4、关闭服务端
*/
```
class Server{
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket(12345);
        Socket s = ss.accept();
        String ip = s.getInetAddress().getHostAddress();
        System.out.println(ip+" connected");

        BufferedInputStream bis = new BufferedInputStream(s.getInputStream());
        BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream("pic.jpg"));
        int by = 0;
        while ((by=bis.read())!=-1) {
            bos.write(by);
        }

        PrintWriter pw = new PrintWriter(s.getOutputStream(),true);
        pw.println("上传成功");

        bos.close();
        s.close();
        ss.close();
    }
}
```
/*
客户端
步骤：
1、连接服务端
2、读取客户端的文件
3、通过 Socket 输出流发给服务端
4、读取服务端的反馈信息

```
	5、关闭服务
*/
class Client{
	public static void main(String[] args) throws Exception {
		Socket s = new Socket("192.168.0.77", 12345);
		File file = new File("Simba.jpg");
		FileInputStream fis = new FileInputStream(file);
		OutputStream os = s.getOutputStream();
		byte[] buf = new byte[1024];
		int len = 0;
		while((len=fis.read(buf))!=-1){
			os.write(buf,0,len);
		}
		s.shutdownOutput();

		InputStream in = s.getInputStream();
		while((len=in.read(buf))!=-1){
			System.out.println(new String(buf,0,len));
		}

		fis.close();
		s.close();
	}
}
```

35、输入一个路径，将该路径下（包括子文件夹）所有以.txt 结尾

输入一个路径，将该路径下（包括子文件夹）所有以.txt 结尾的文件复制到别的路径

```
package demo;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Scanner;

/**
 * 35、输入一个路径，将该路径下（包括子文件夹）所有以.txt 结尾的文件复制到别的路径
```

```
 *
 * 思路：
 * 1、使用 Scanner 在控制台输入一个路径字符串
 * 2、listFiles 获得该路径文件夹下的文件列表 File[] ,遍历 ,遇到文件夹就递归 ,遇到以".txt" 结尾的就复制。
 * */
public class demo35 {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入一个路径：");
        String path = sc.next();
        sc.close();

        File olddir = new File(path);
        File newdir = new File("d:\\txt");

        listTxt(olddir,newdir);
        renameTxt(newdir);
    }
    public static void listTxt(File olddir, File newdir) throws Exception {
        if(!olddir.exists()){
            System.out.println("文件不存在");
            return;
        }
        if(!newdir.exists() || !newdir.isDirectory()){
            newdir.mkdirs();
        }
        File f = new File(newdir.getPath()+"\\"+olddir.getName());
        f.mkdir();

        File[] files = olddir.listFiles();
        for (File file : files) {
            if(file.isDirectory()){
                listTxt(file,f);
            } else if(file.getName().endsWith(".txt")) {
                copyTxt(file,f);
            }
        }
```

```
    }
    public static void copyTxt(File olddir, File newdir) throws Exception {
        BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(olddir));
        BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(newdir.getPath()+"\\"+olddir.getName()));
        int by = 0;
        while ((by=bis.read())!=-1) {
            bos.write(by);
        }
        bis.close();
        bos.close();
    }
    public static void renameTxt(File dir) {
        if(!dir.exists()){
            System.out.println("文件不存在");
        }
        File[] files = dir.listFiles();
        for (File file : files) {
            if (file.isDirectory()) {
                renameTxt(file);
            } else if(file.getName().endsWith(".txt")){
                File newfile = new File(file.getPath().replaceAll("\\.txt", ".java"));
                file.renameTo(newfile);
            }
        }
    }
}
```

36、有 100 个人围成一个圈，从 1 开始报数，报到 14 的这个人就要退出。

有 100 个人围成一个圈 ,从 1 开始报数 ,报到 14 的这个人就要退出。然后其他人重新开始，从 1 报数，到 14 退出。问：最后剩下的是 100 人中的第几个人？

```
package com.itheima;

import java.util.ArrayList;
import java.util.Iterator;

/**
```

```
 * 第 36 题：有 100 个人围成一个圈，从 1 开始报数，报到 14 的这个人就要退出。然后其他人重新开始，从 1 报数，到 14 退出。问：最后剩下的是 100 人中的第几个人？
 *
 * @author
 *
 */

public class Test36 {
    public static void main(String[] args) {

        ArrayList<Integer> al = new ArrayList<Integer>();      //创建 List 集合
        for (int i = 1; i <= 100; i++) {       //循环往集合中添加 1~100 的数据
            al.add(i);
        }

        int count = 0;       //s 初始化遍历集合中元素的次数

        while (al.size() > 1) {    //当集合的长度大于 1 时继续遍历集合中的元素
            Iterator<Integer> it = al.iterator();    //获取最新集合的迭代器
            while (it.hasNext()) {
                int n = it.next();    //定义 n 表示具体元素
                count++;

                if (count % 14 == 0) {       //当遍历集合中元素的次数是 14 的整数倍时将该元素移除出集合
                    System.out.println("当前删除的元素是：" + n);
                    it.remove();
                }
            }
        }
        System.out.println("最后剩下的是 100 人中的第" + al.get(0) + "个人");
    }
}
```

37、有一个类为 ClassA,有一个类为 ClassB，在 ClassB 中有一个方法 b

有一个类为 ClassA,有一个类为 ClassB，在 ClassB 中有一个方法 b，此方法抛出异常，在 ClassA 类中有一个方法 a，请在这个方法中调用 b,然后抛出异常。在客户端有一个类为 TestC,有一个方法为 c ,请在这个方法中捕捉异常的信息。完成这个例子，请说出 java 中针对异常的处理机制。

```
package com.itheima;

/**
 * 第 37 题：有一个类为 ClassA,有一个类为 ClassB，在 ClassB 中有一个方法 b，此方法抛出异常，在 ClassA 类中有一个方法 a，请在这个方法中调用 b,然后抛出异常。
 *          在客户端有一个类为 TestC,有一个方法为 c ,请在这个方法中捕捉异常的信息。完成这个例子，请说出 java 中针对异常的处理机制。
 *
 * @author
 * 答：（1）用 try...catch 语句捕获并且处理异常；
 *      （2）使用 throw 抛出异常，异常必须是 java.lang.Throwable 类的对象或者该类的子类的对象；
 *      （3）使用 throws 声明方法抛出异常；
 *      （4）使用 finally 语句声明在任何情况下都会执行的代码。
 *
 */
public class Test37 {
    public static void main(String[] args) {
        ClassC.methodC();

    }
}

class ClassB {
    public static void methodB() throws Exception {
        throw new Exception("this is exception");
    }
}

class ClassA {
    public static void methodA() throws Exception {
        ClassB.methodB();
    }
}

class ClassC {
    public static void methodC() {
        try {
```

```
            ClassA.methodA();
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

38、已知文件 a.txt 文件中的内容为“bcdeadferwplkou”

已知文件 a.txt 文件中的内容为“bcdeadferwplkou”，请编写程序读取该文件内容，并按照自然顺序排序后输出到 b.txt 文件中。即 b.txt 中的文件内容应为“abcd……”这样的顺序。

```
package com.itheima;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Arrays;

/**
 * 第 38 题：已知文件 a.txt 文件中的内容为“bcdeadferwplkou”，请编写程序读取该文件内容，并按照自然顺序排序后输出到 b.txt 文件中。
 * 即 b.txt 中的文件内容应为“abcd……”这样的顺序。
 *
 * @author
 *
 */
public class Test38 {
    public static void main(String[] args) {

        FileInputStream fis = null;
        FileOutputStream fos = null;
        try {
            fis = new FileInputStream("d:\\a.txt");
            fos = new FileOutputStream("d:\\b.txt");
            byte[] buf = new byte[fis.available()];    //创建了一个关联文件大小一样的缓冲区
                    fis.read(buf);
                Arrays.sort(buf);
                fos.write(buf);
```

```
                fos.flush();
            } catch (IOException e) {
                System.out.println(e.toString());
            } finally {
                try {
                    fis.close();
                } catch (IOException e) {
                    System.out.println(e.toString());
                }
                try {
                    fos.close();
                } catch (IOException e) {
                    System.out.println(e.toString());
                }
            }
        }
}
```

39、一共有四个售票口同时卖 100 张票

一共有四个售票口同时卖100张票 ,请模拟售票过程输出时要显示是哪个窗口卖第几张票。

```
package examTwo;

/**
 *
 * 第 39 题：一共有四个售票口同时卖 100 张票，请模拟售票过程输出时要显示是哪个窗口卖第几张票。
 *
 * @author 作者--谭威
 *
 */

public class Demo39 {
   public static void main(String[] args){
        Tickets t = new Tickets();
        Thread t1 = new Thread(t);
        Thread t2 = new Thread(t);
        Thread t3 = new Thread(t);
        Thread t4 = new Thread(t);
        t1.start();
```

```
        t2.start();
        t3.start();
        t4.start();

    }
}
class Tickets implements Runnable{
    private int tickets = 100;
    Object obj = new Object();
    @Override
    public void run() {
        while(true){
            synchronized(obj){
                if(tickets > 0){
                    try{Thread.sleep(3);}catch(InterruptedException e){}

System.out.println(Thread.currentThread().getName()+"..."+tickets--);
                }
            }
        }
    }

}
```

40、声明类 Student，包含 3 个成员变量

声明类 Student，包含 3 个成员变量：name、age、score，创建 5 个对象装入 TreeSet，按照成绩排序输出结果（考虑成绩相同的问题）。

```
package com.itheima;

import java.util.TreeSet;

/**
 * 第 40 题： 声明类 Student，包含 3 个成员变量：name、age、score，创建 5 个对象装入 TreeSet，按照成绩排序输出结果（考虑成绩相同的问题）。
 * @author
 *
 */
class Student implements Comparable{
    private String name;
```

```
    private int age;
    private int score;

    Student(String name,int age,int score){
        this.name = name;
        this.age = age;
        this.score = score;
    }

    public int compareTo(Object o) {

        Student stu = (Student) o;

        if (this.score > stu.score)
                return 1;
        if (this.score == stu.score) {
            // 如果成绩相同的时候，要判断姓名是否相同
                return this.name.compareTo(stu.name);
        }

            return -1;
    }

    public String toString(){

        return "Student:name=" + name + ",age=" + age + ",score=" + score+" ";
    }


}
public class Test40 {
  public static void main(String[] args){
      TreeSet<Student> treeSet = new TreeSet<>();  //创建集合对象
              //添加元素
        treeSet.add(new Student("Xu Sanduo", 18, 96));
        treeSet.add(new Student("Lin Wuzhe", 20, 95));
        treeSet.add(new Student("Cheng Cai", 22, 95));
        treeSet.add(new Student("Yuan Lang", 23, 111));
```

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

    }
}
```

41、编写程序，将指定目录下所有.java 文件拷贝到另一个目的中

编写程序，将指定目录下所有.java 文件拷贝到另一个目的中，并将扩展名改为.txt。

```
package com.itheima;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.ArrayList;
import java.util.List;

/**
 * 第 41 题：编写程序，将指定目录下所有.java 文件拷贝到另一个目的中，并将扩展名改为.txt。
 * @author
 */
public class Test41 {
    public static void main(String[] args) throws Exception{
        List<String> list = new ArrayList<String>();  //定义用于存储.java 文件的名字

        File file = new File("d:\\MyProject\\exam\\src\\com\\itheima");  //定义指定目录
        File destFile = new File("d:\\txt");    //定义目的文件路径
        if(! destFile.exists()){   //判断目的文件是否存在
            destFile.mkdirs();      //如果不存在，进行创建
        }

        fileList(file,list);          //将指定文件中的.java 文件的名字存储到集合中
        for (String fileName : list) {       //高级 for 循环遍历
            write(file,destFile,fileName);    //将指定文件中的.java 文件存储到目的文件中
        }
```

```
  }

    public static void write(File file, File destFile, String fileName) throws Exception {
        BufferedReader bufr = new BufferedReader(new FileReader(new File(file,fileName)));  //通过缓冲区读取指定文件夹中的.java 文件名称
        BufferedWriter bufw = new BufferedWriter(new FileWriter(new File(destFile,fileName.replace(".java", ".txt"))));  //通过缓冲区进行对扩展名的替换，并存储到目的文件中

          //频繁读写操作
        String line = null;
        while((line = bufr.readLine()) != null){
            bufw.write(line);
            bufw.newLine();
            bufw.flush();
        }

         //资源的关闭
        bufr.close();
        bufw.close();

  }

public static void fileList(File file, List<String> list) {
            File[] files = file.listFiles();      //获取指定文件夹中的所有.java 文件
            for (File f : files) {      //高级 for 循环遍历
              list.add(f.getName());    //将所有的.java 文件添加到集合中
         }

    }


}
```

42、用代码证明，在 try 中写了 return，后面又写了 finally

用代码证明 ,在 try 中写了 return ,后面又写了 finally ,是先执行 return 还是先执行 fianlly ?

```
package com.itheima;

/**
```

```
 * 第 42 题： 用代码证明，在 try 中写了 return，后面又写了 finally，是先执行 return 还是先执行 fianlly ?
 * @author
 * 答：return 语句已经执行了再去执行 finally 语句，不过并没有直接返回，而是等 finally 语句执行完了再返回结果。
 * 证明代码如下：
 */
public class Test42 {
    public static void main(String[] args) {

        System.out.println(Test6.test());
    }

    public static int test(){
        int x = 3;
        try{
            System.out.println("try run");
            return x = 3 + 4;
        }
        finally{
            System.out.println("finally run");
            if(x > 2){
                System.out.println("x="+x);
            }
        }

    }
}
```

43、编写一个可以获取文件扩展名的函数

编写一个可以获取文件扩展名的函数 ,形参接收一个文件名字符串 ,返回一个扩展名字符串。

```
package com.itheima;

import java.io.File;

/**
 * 第 43 题：编写一个可以获取文件扩展名的函数，形参接收一个文件名字符串，返回一个扩展名字符串。
 * @author
```

```
 *
 */
public class Test43 {
   public static void main(String[] args){
         File file = new File("d:\\me.txt");
         getName(file);
      }
         public static void getName(File file){
         String name = file.getName();
         String s = name.substring(name.lastIndexOf(".")+1);
         System.out.println("文件名为"+name+"拓展名为："+s);
    }

}
```

44、判断一个字符串是否是对称字符串

判断一个字符串是否是对称字符串，例如"abc"不是对称字符串，"aba"、"abba"、"aaa"、"mnanm"是对称字符串

```
package com.itheima;

/**
 * 第 44 题 判断一个字符串是否是对称字符串 ,例如"abc"不是对称字符串 ,"aba"、"abba"、"aaa"、"mnanm"是对称字符串
 * @author
 *
 */
public class Demo44 {
   public static void main(String[] args){
         String str = "abc";
         cmp(str);

   }

public static void cmp(String str) {
      boolean result = true;   //定义默认结果是正确的
      int count = (str.length()-1)/2; //定义个数 ,如果是奇数除以 2,int 类型结果还是整数，用于 for 循环比较的次数;

      for (int x = 0;x <= count ;x++ ){   //循环遍历
```

```
            if(str.charAt(x) != str.charAt(str.length()-1-x)){   //指定索引位置返回的字符值进行比较，如果不相等
                    result = false; //结果为不对称
            }
        }
        if(! result){
            System.out.println("该字符串是不对称的");

        }
        else{
            System.out.println("该字符串是对称的");

        }
    }

}
```