1. **编写一个程序，实现从命令行参数输入两个字符串类型的数值，并计算输出两个数值的和。**

**2、编写一个程序，实现从命令行参数输入一字符串，统计该字符串中字符“e”出现的次数。**

**package** zuoye;

/\*\*

\* 输出e的出现次数

\*/

**import** java.util.Scanner;

**public** **class** Test2 {

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

Scanner input = **new** Scanner(System.***in***);

String str=input.next();

**int** count=0;

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

{

**if**(str.charAt(i)=='e')

{

count++;

}

}

System.***out***.println("e的出现次数是："+count);

}

}

**3、生成十个0~100之间的随机数，放到数组中，然后排序输出。**

**package** zuoye;

**import** java.util.Arrays;

/\*\*

\* 生成十个0~100之间的随机数，放到数组中，然后排序输出

\* **@author** Administrator

\*

\*/

**public** **class** Test3 {

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

**int** arr[]=**new** **int**[10];//定义一个长度为10的int数组

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

arr[i]=(**int**) (Math.*random*()\*100);//取0~1的随机数\*100，即为0~100的随机数

}

Arrays.*sort*(arr);//排序方法

**for**(**int** a:arr)//增强for循环

{

System.***out***.print(a+" ");

}

}

}

**4、巴黎时间比北京时间晚7个小时，纽约时间比北京时间晚12个小时，试编写一程序，根据输入的北京时间输出相应的巴黎和纽约时间。**

package zuoye;

import java.util.Calendar;

import java.util.Date;

public class Test4 {

public void getTime(int y,int m,int d,int h,int mi,int s){

//获取Calendar实例

Calendar time = Calendar.getInstance();

//设置巴黎时间

time.set(y,m,d,h-7,mi,s);

//输出巴黎时间

System.out.println("巴黎时间是"+time.get(Calendar.YEAR)+"年"+

((time.get(Calendar.MONTH)))+"月"+

time.get(Calendar.DAY\_OF\_MONTH)+"日"+

time.get(Calendar.HOUR\_OF\_DAY)+"时"+

time.get(Calendar.MINUTE)+"分"+

time.get(Calendar.SECOND)+"秒");

//设置纽约时间

time.set(y,m,d,h-12,mi);

//输出纽约时间

System.out.println("纽约时间是"+time.get(Calendar.YEAR)+"年"+

((time.get(Calendar.MONTH)))+"月"+

time.get(Calendar.DAY\_OF\_MONTH)+"日"+

time.get(Calendar.HOUR\_OF\_DAY)+"时"+

time.get(Calendar.MINUTE)+"分"+

time.get(Calendar.SECOND)+"秒");

}

public static void main(String[] args) {

Test4 time = new Test4();

//设置北京时间

time.getTime(2018,3,15,16,53,9);

}

}

**5、**解析一个邮箱地址是否合法，如果合法则打印出用户名部分和该邮箱所属的网站域名

如果邮箱地址不合法则显示不合法的原因

提示：邮箱地址不合法的因素：

1. 邮箱地址中不包含@或。
2. 邮箱地址中含有多了@或。
3. 邮箱地址中。出现在@的前面
4. 用户名里有其他字符

实现步骤：

(1) 创建一个类，类名：mailtest

类图如下：



(类名和方法名必须与要求一样。区分大小写)

**package** youxiang;

**import** java.util.Scanner;

**public** **class** MailTest {

**public** **static** **boolean** testMail(){

Scanner input = **new** Scanner(System.***in***);

String s = input.next();

**if**(s.indexOf("@")==-1||s.indexOf(".")==-1)

{

System.***out***.println("邮箱地址中不包含@或.");

**return** **false**;

}

**if**(s.indexOf("@")!=s.lastIndexOf("@")||s.indexOf(".")!=s.lastIndexOf("."))

{

System.***out***.println("邮箱地址中含有多余的@或.");

**return** **false**;

}

**if**(s.indexOf("@")>s.lastIndexOf("."))

{

System.***out***.println("邮箱地址中.出现在@的前面");

**return** **false**;

}

**for**(**int** i=0;i<s.indexOf("@");i++)

{

**if**( (s.charAt(i)>='a'&&s.charAt(i)<='z')||(s.charAt(i)>='A'&&s.charAt(i)<='Z'))

{

}**else**

{

System.***out***.println("用户名里有其他字符");

**return** **false**;

}

}

**return** **true**;

}

}

**package** youxiang;

**public** **class** Test {

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

// **TODO** Auto-generated method stub

**if**(MailTest.*testMail*()){

System.***out***.println("邮箱格式合法");

}**else**{

System.***out***.println("邮箱格式不合法");

}

}

}

**6、分别在控制台输入字符串和子字符串，并计算字符串中子字符串出现的次数。**

**package** zuoye;

**import** java.util.Scanner;

/\*\*

\* 控制台输入一段字符串，再查询一段子字符串出现的次数

\*

\*/

**public** **class** Test6 {

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

Scanner input = **new** Scanner(System.***in***);

String a = input.next();

String b = input.next();

**int** count = 0;//计数器

**for**(**int** i = 0;i<a.length()-b.length();i=a.indexOf(b, i)+1)

{

**if**(a.indexOf(b, i)!=-1)

{

count++;

}

}

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

}

}

# 集合类

1、请使用LinkedList来模拟一个队列(先进先出的特性):

1) 拥有放入对象的方法void put(Object o)

1. 取出对象的方法Object get()
2. 判断队列当中是否为空的方法boolean isEmpty()；并且，编写测试代码，验证你的队列是否正确。

**public** **class** Linkeds {

List l;

Linkeds(){

l=**new** LinkedList();

}

**public** **void** put(Object o){

l.add(o);

}

**public** Object get(){

Object o= l.get(0);

l.remove(0);

**return** o;

}

**public** **boolean** isEmpty(){

**if**(l.isEmpty()){

**return** **true**;

}

**return** **false**;

}

public static void main(String[] args){

//定义一个空队列

Linkeds l = new Linkeds();

//往队列中放入对象

l.put("Tom1");

l.put("John2");

l.put("Mary3");

//如果队列不为空，依次输出队列中的元素

while(!l.isEmpty()){

System.out.println(l.get());

}

}

}

**2、假设顺序列表ArrayList中存储的元素是整型数字1~5，遍历每个元素，将每个元素顺序输出。**

package zuoye;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class Test1 {

public static void main(String[] args) {

List<Number> list=new ArrayList<>();

Number n1=new Number(1);

Number n2=new Number(2);

Number n3=new Number(3);

Number n4=new Number(4);

Number n5=new Number(5);

list.add(n1);

list.add(n2);

list.add(n3);

list.add(n4);

list.add(n5);

Iterator it=list.iterator();

while(it.hasNext())

{

System.out.print(it.next());

}

}

}

3、在一个列表中存储以下元素：apple,grape,banana,pear

1）返回集合中的最大的和最小的元素

2）将集合进行排序，并将排序后的结果打印在控制台上

**package** zuoye;

**public** **class** Fruit {

**private** String fruit;

**public** Fruit()

{

}

**public** Fruit(String fruit)

{

**this**.fruit=fruit;

}

**public** String getFruit() {

**return** fruit;

}

**public** **void** setFruit(String fruit) {

**this**.fruit = fruit;

}

}

package zuoye;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class FruitTest {

public static void main(String[] args) {

List fruit = Arrays.asList("apple grade banana pear".split(" "));

System.out.println("最大值是：" + Collections.max(fruit));

System.out.println("最小值是：" + Collections.min(fruit));

}

}

**4、编写一个程序，创建一个 HashMap对象，用于存储银行储户的信息(其中储户的主要信息有储户的ID，姓名和余额)。另外，计算并显示其中某个储户的当前余额。**

**package** zuoye;

**public** **class** Bank {

**private** String id;

**private** String name;

**private** **double** money;

**public** Bank()

{

}

**public** Bank(String id,String name,**double** money)

{

**this**.id=id;

**this**.name=name;

**this**.money=money;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **double** getMoney() {

**return** money;

}

**public** **void** setMoney(**double** money) {

**this**.money = money;

}

}

package zuoye;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

public class BankTest {

public static void main(String[] args) {

Map<String,Bank> map=new HashMap<String,Bank>();

map.put("用户1", new Bank("1","张三",Math.random()\*100000));

map.put("用户2", new Bank("2","李四",Math.random()\*100000));

map.put("用户3", new Bank("3","王五",Math.random()\*100000));

map.put("用户4", new Bank("4","赵六",Math.random()\*100000));

Set keySet=map.keySet();

Iterator it = keySet.iterator();

double sum = 0;

while(it.hasNext()){

Object key = it.next();

Bank a = map.get(key);

System.out.println("账户"+a.getId()+"的余额："+a.getMoney());

//计算所有帐户的余额之和

sum += a.getMoney();

}

System.out.println("所有账户余额："+sum);

}

}

5、从控制台输入若干个单词（输入回车结束）放入集合中，将这些单词排序后（忽略大小写）打印出来。

package zuoye;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Test5 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

String str = input.nextLine();

String[] arr = str.split(" ");

String tmp;

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

for (int j = i + 1; j < arr.length - 1; j++) {

if (arr[i].compareToIgnoreCase(arr[j]) > 0) {

str = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = str;

}

}

}

for (String a : arr) {

list.add(a);

System.out.println(a);

}

}

}

# IO流

**1、在本机的磁盘系统中，找一个文件夹，利用File类的提供方法，列出该文件夹中的所有文件的文件名和文件的路径，执行效果如下：**

-

路径名：c:\temp\def.txt

**package** 作业;

**import** java.io.File;

**public** **class** Test1 {

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

File file = **new** File("E:\\新建文件夹");

String[] files = file.list();

**for** (String o : files) {

System.***out***.println("文件名:" + o);

System.***out***.println("路径:" + file.getAbsolutePath() + "\\" + o);

System.***out***.println("--------------------------------------");

}

}

}

**2、编写一个java程序实现文件复制功能，要求将d:/io/copysrc.doc中的内容复制到d:/io/copydes.doc中。**

package 作业;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

/\*\*

\* 文件的拷贝

\*/

public class Test2 {

public static void main(String[] args) {

File file = new File("file\\abc.txt");

File file\_back = new File("file\\abc\_back.txt");

try (FileInputStream fi = new FileInputStream(file); FileOutputStream fo = new FileOutputStream(file\_back);) {

int b = -1;

while ((b = fi.read()) != -1) {

fo.write(b);

}

} catch (Exception e) {

e.printStackTrace();

}

System.out.println("拷貝成功！");

}

}

**3、创建c:/test.txt文件并在其中输入"hello world"**

**创建一个输入流读取该文件中的文本**

**并且把小写的l变成大写L再利用输出流写入到d:\test.txt中**

**实现步骤：1.在本地硬盘C盘下创建一个文件test.txt**

**2.创建一个包含main()方法的类，并在main中编写代码**

**3．运行代码并且测试结果**

**实现过滤器的功能**

**效果显示：**



package 作业;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

\* 替换文件中的小写字母l为大写字母L

\*/

public class Test3 {

public static void main(String[] args) {

FileReader f;

String s = "";

BufferedReader br = null;

try {

f = new FileReader("F:\\test.txt");

br = new BufferedReader(f);

s = br.readLine();

System.out.println("源文件" + s);

s = s.replace("l", "L");

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

br.close();

} catch (IOException e) {

e.printStackTrace();

}

}

BufferedWriter bw = null;

System.out.println(s);

try {

FileWriter fw = new FileWriter("F:\\test.txt");

bw = new BufferedWriter(fw);

bw.write(s);

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

bw.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**4、在程序中创建一个Student类型的对象，并把对象信息保存到d:/io/student.txt文件中，然后再从文件中把Student对象的信息读出显示在控制台上，Student类的描述如下：**

**package** 作业;

**import** java.util.Date;

**public** **class** Student {

**private** **int** id;

**private** String name;

**private** Date birth;

**public** Student() {

**super**();

}

**public** Student(**int** id, String name, Date birth) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.birth = birth;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Date getBirth() {

**return** birth;

}

**public** **void** setBirth(Date birth) {

**this**.birth = birth;

}

@Override

**public** String toString() {

**return** "Student [id=" + id + ", name=" + name + ", birth=" + birth + "]";

}

}

public class SerializationExercise {

public static void main(String args[]){

Student s1 = new Student("1","zhangsan","1999-11-11");

try {

FileOutputStream fos = new FileOutputStream("d:\\io\\student.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

System.out.println("对象序列化...");

oos.writeObject(s1);

oos.flush();

oos.close();

} catch (FileNotFoundException e1) {

System.out.println(e1.getMessage());

} catch (IOException e2) {

System.out.println(e2.getMessage());

}

try {

FileInputStream fis = new FileInputStream("d:\\io\\student.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

System.out.println("反序列化...");

Student ss1 = (Student)ois.readObject();

System.out.println(ss1);

ois.close();

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} catch (IOException e) {

System.out.println(e.getMessage());

} catch (ClassNotFoundException e) {

System.out.println(e.getMessage());

}

}

}

# 多线程

**1、利用Thread实现，要求多线程求解某范围素数每个线程负责1000范围：线程1找1-1000；线程 2 找 1001-2000；线程 3 找2001-3000。编程程序将每个线程找到的素数及时打印。**

**package** 找素数;

**public** **class** Number {

**private** **int** num;

**public** Number() {

**super**();

}

**public** Number(**int** num) {

**super**();

**this**.num = num;

}

**public** **int** getNum() {

**return** num;

}

**public** **void** setNum(**int** num) {

**this**.num = num;

}

**public** **synchronized** **void** checkNum(**int** num) {

**if** (num == 1000) {

**for** (**int** n = 2; n <= num; n++) {

**boolean** b = **true**;

**if** (n != 1) {

**for** (**int** i = 2; i < n; i++) {

**if** (n % i == 0) {

b = **false**;

**break**;

}

}

}

**if** (b) {

System.***out***.println(n + "是质数");

}

}

}

**if** (num == 2000) {

**for** (**int** n = 1001; n <= num; n++) {

**boolean** b = **true**;

**if** (n != 1) {

**for** (**int** i = 2; i < n; i++) {

**if** (n % i == 0) {

b = **false**;

**break**;

}

}

}

**if** (b) {

System.***out***.println(n + "是质数");

}

}

}

**if** (num == 3000) {

**for** (**int** n = 2001; n <= num; n++) {

**boolean** b = **true**;

**if** (n != 1) {

**for** (**int** i = 2; i < n; i++) {

**if** (n % i == 0) {

b = **false**;

**break**;

}

}

}

**if** (b) {

System.***out***.println(n + "是质数");

}

}

}

}

}

**package** 找素数;

//public class Numbers1 implements Runnable

**public** **class** Numbers1 **extends** Thread {

**private** Number num;

**public** Numbers1(Number num) {

**this**.num=num;

}

**public** **void** run() {

num.checkNum(1000);

}

}

**package** 找素数;

//public class Numbers1 implements Runnable

**public** **class** Numbers2 **extends** Thread {

**private** Number num;

**public** Numbers2(Number num) {

**this**.num=num;

}

**public** **void** run() {

num.checkNum(2000);

}

}

**package** 找素数;

//public class Numbers1 implements Runnable

**public** **class** Numbers3 **extends** Thread {

**private** Number num;

**public** Numbers3(Number num) {

**this**.num=num;

}

**public** **void** run() {

num.checkNum(3000);

}

}

**package** 找素数;

**public** **class** Test {

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

Number num=**new** Number();

Numbers1 n1=**new** Numbers1(num);

Thread t1=**new** Thread(n1);

t1.setName("1~1000:");

t1.start();

Numbers2 n2=**new** Numbers2(num);

Thread t2=**new** Thread(n2);

t2.setName("1001~2000:");

t2.start();

Numbers3 n3=**new** Numbers3(num);

Thread t3=**new** Thread(n3);

t3.setName("2001~3000");

t3.start();

}

}

**2、利用Runnable实现，要求多线程求解某范围素数每个线程负责1000范围：线程1找1-1000；线程 2 找 1001-2000；线程 3 找2001-3000。编程程序将每个线程找到的素数及时打印。**

与第一题相同，不再贴出代码，第一题注释即为修改处。

**3、编写一个Java程序(包括一个主程序类，一个线程类。在主程序类中创建2个线程，将其中一个线程的优先级设为10，另一个线程的优先级设为6。让优先级为10的线程打印200次“线程1正在运行”，优先级为6的线程打印200次“线程2正在运行”。**

**package** 优先级;

**public** **class** Main {

**private** **int** num;

**public** **int** getNum() {

**return** num;

}

**public** **void** setNum(**int** num) {

**this**.num = num;

}

**public** Main() {

**super**();

}

**public** Main(**int** num) {

**super**();

**this**.num = num;

}

**public** **void** dayin(**int** num){

**for**(**int** i=0;i<num;i++){System.***out***.println(Thread.*currentThread*().getName()+"正在运行");}

}

}

**package** 优先级;

**public** **class** Xiancheng1 **implements** Runnable {

**private** Main num;

**public** Xiancheng1() {

**super**();

}

**public** Xiancheng1(Main num) {

**this**.num = num;

}

@Override

**public** **void** run() {

num.dayin(200);

}

}

**package** 优先级;

**public** **class** Xiancheng2 **implements** Runnable {

**private** Main num;

**public** Xiancheng2() {

**super**();

}

**public** Xiancheng2(Main num) {

**this**.num = num;

}

@Override

**public** **void** run() {

num.dayin(200);

}

}

**package** 优先级;

**public** **class** Test {

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

Main num=**new** Main();

Xiancheng1 x1=**new** Xiancheng1(num);

Thread t1=**new** Thread(x1);

t1.setPriority(10);

t1.setName("线程1");

t1.start();

Xiancheng2 x2=**new** Xiancheng2(num);

Thread t2=**new** Thread(x2);

t2.setPriority(6);

t2.setName("线程2");

t2.start();

}

}

**4、编写一个计时器，每隔一秒钟，在控制台打印出最新时间。**

package 计时器;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Main {

private Date date;

public Main() {

super();

// TODO Auto-generated constructor stub

}

public Main(Date date) {

super();

this.date = date;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public void getDay() {

while (true) {

Date date=new Date();

SimpleDateFormat num = new SimpleDateFormat("hh:mm:ss");

String dateString = num.format(date);

System.out.println(Thread.currentThread().getName() +dateString );

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**package** 计时器;

**public** **class** Clock **implements** Runnable {

**private** Main num;

**public** Clock() {

**super**();

}

**public** Clock(Main num) {

**super**();

**this**.num = num;

}

@Override

**public** **void** run() {

num.getDay();

}

}

**package** 计时器;

**public** **class** Test {

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

Main num = **new** Main();

Clock c = **new** Clock(num);

Thread t = **new** Thread(c);

t.setName("当前时间为：");

t.start();

}

}