**1、判断电话号码和邮箱**

package dh14;

import java.util.Scanner;

//判断电话号码

public class MatchesDemo1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入要进行判断的qq号码");

String string = sc.nextLine();

boolean b = isNuCheck(string);

System.out.println(b);

}

public static boolean isNuCheck(String string) {

boolean flag = true;

if(string.length()>=5&&string.length()<=15) {

if(string.charAt(0)!='0') {

char[] ch = string.toCharArray();

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

if(!Character.isDigit(ch[i])) {

flag = false;

return flag;

}

}

}else {

flag = false;

return flag;

}

}else {

flag = false;

return flag;

}

return flag;

}

}

**2、按照不同的规则分割数据**

package dh14;

//按照不同的规则分割数据

public class SpiltDemo1 {

public static void main(String[] args) {

String s1 ="aa,bb,cc";

String[] strings = s1.split(",");

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

System.out.print(strings[i]+" ");

}

System.out.println();

String s2 = "aa.bb.cc";

String[] strings2 = s2.split("\\.");

for(int i=0;i<strings2.length;i++) {

System.out.print(strings2[i]+" ");

}

System.out.println();

//路径必须是双\\

String s3 = "D:\\JSP实验\\第二次实验\\登录验证";

String[] strings3 = s3.split("\\\\");

for(int i=0;i<strings3.length;i++) {

System.out.print(strings3[i]+" ");

}

}

}

**3、把字符串中的数字替换成\***

package dh14;

//将字符串中的数字替换成\*

public class ReplaceAllDemo1 {

public static void main(String[] args) {

String string = "java121212andweb";

String string2 = string.replaceAll("\\d+", "\*");//表示用\*替换满足正则表达式的字符串

System.out.println(string2);

//代表一个数字就用一个\*替换

String string3 = "java121212andweb";

String string4 = string3.replaceAll("\\d", "\*");

System.out.println(string4);

}

}

**4、获取字符串中由三个字符组成的字符串**

package dh14;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

//获取字符串中由三个字符组成的类

public class HuoQuDemo1 {

public static void main(String[] args) {

String string = "ni sha diao ba shi jiu shi";

//创建模式对象

Pattern p = Pattern.compile("\\b\\w{3}\\b");

//将模式对象放在匹配器中

Matcher m = p.matcher(string);

//调用 find 找到 group 输出找到的字符串

while(m.find()) {

System.out.println(m.group());

}

}

}

**6、获取任意范围内的随机数**

package dh14;

import java.util.Scanner;

//获取任意范围内的随机数

public class RandomDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入起始范围：");

int start = sc.nextInt();

System.out.println("请输入终止范围：");

int end = sc.nextInt();

for(int i=0;i<100;i++) {

int m =(int)(Math.random()\*(end-start))+start;

System.out.println(m);

}

}

}

**7、来到世界多少天**

package dh14;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

//来到这个世界多少天

public class DateCountDemo {

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

Scanner sc = new Scanner(System.in);

System.out.println("请输入出生日期：");

String string = sc.nextLine();

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

Date date = dateFormat.parse(string);

long l =(new Date().getTime()-date.getTime())/1000/60/60/24;

System.out.println(l);

}

}

**8、输出任意年的2月份有多少天**

package dh14;

import java.util.Calendar;

import java.util.Scanner;

//获取任意年份的2月份有多少天

public class CalendarDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入要进行判断的年份");

int year = sc.nextInt();

Calendar c = Calendar .getInstance();

c.set(year, 2, 1);

c.add(Calendar.DATE, -1);

System.out.println(c.get(Calendar.DATE));

}

}

**9、将字符串“91 27 46 38 50”**

**\* 用代码实现输出结果为：“27 38 46 50 91”**

package dh14;

import java.util.Arrays;

/\*

\* 将字符串“91 27 46 38 50”

\* 用代码实现输出结果为：“27 38 46 50 91”(字符串)

\*/

public class SplitDemo {

public static void main(String[] args) {

String string = "91 27 46 38 50";

String[] strings = string.split(" ");

int[] arr = new int[strings.length];

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

arr[i] = Integer.parseInt(strings[i]);

}

Arrays.sort(arr);

StringBuffer buffer = new StringBuffer();

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

buffer.append(arr[j]+" ");

}

String string2 = buffer.toString().trim();

System.out.println(string2);

}

}

**10、Date---String 之间的转换**

package dh14;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

//Date----String之间的转换

public class Date\_String {

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

Date d = new Date();

String string = dateToString(d,"yyyy-MM-dd");

System.out.println(string);

String s = "2020-01-02";

Date date = stringToDate(s,"yyyy-MM-dd");

System.out.println(date);

}

public static String dateToString(Date d,String pattern ) {

return new SimpleDateFormat(pattern).format(d);

}

public static Date stringToDate(String s,String pattern) throws ParseException {

return new SimpleDateFormat(pattern).parse(s);

}

}

StringBuffer---------------Arrays-----------Integer---------Character

StringBuffer的

构造方法：

public StringBuffer(int i)

public StringBuffer(String s)

添加(拼接)方法：append() //任意类型均可

插入 insert()

删除方法：delete(int start, int end)

deleteCharAt(int index) //删除指定索引位置的字符

替换方法：replace(int start ,int end ,String s) //表明从哪个位置开始到那个位置结束，用字符串s替换

反转方法：reverse（）

截取方法：substring(int start,int end)

substring(int start) //默认到最后结束

转变成字符串类型： toString()

题目：

**1、StringBuffer与String的转换**

//String -----StringBuffer (构造方法)

Stringbuffer buffer = new StringBuffer(String s);

//StringBuffer----String (构造方法)

String s = new String(new StringBuffer().append(str)); //str代表字符串的内容

/\*\*\*\*\*/ public String toString( ) 将一个StringBuffer 类型用String类输出

StringBuffer buffer2 = new StringBuffer().append("java");

String ss = buffer2.toString();

**2、把数组拼接(append)成指定格式的字符串 (用StringBuffer类写)**

public class Int\_StringDemo2 {

public static void main(String[] args) {

//定义一个数组

int[] arr = {60,20,16,20,83};

//方法调用

String string = new String(intToChange(arr));

System.out.println("按格式输出的字符串为 ="+string);

}

public static StringBuffer intToChange(int[] arr) {

StringBuffer buffer = new StringBuffer();

buffer.append("[");

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

if(i!=arr.length-1) {

buffer.append(arr[i]).append(",");

}else {

buffer.append(arr[i]).append("]");

}

}

return buffer;

}

}

**3、判断一个字符串是否对称**

分析思路：

\* 方法一；将字符串转换成字符数组，（分别前后遍历开始）

\* 遍历判断前面的字符与后面字符是否相等

\* 方法二、将string类型变成stringBbuffer

\* 用StringBuffer中的reverse方法进行翻转

\* 在转换回string类型。在equals比较

\*

\*/

public class DuiChenDemo3 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入要进行判断的字符串：");

String string = sc.nextLine();

boolean b = isSame(string);

System.out.println("字符串是否对称："+b);

//方法二

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

boolean b2 = isTheSame(string);

System.out.println("字符串是否对称："+b2);

}

//方法实现

public static boolean isSame(String string) {

char[] chs = string.toCharArray();

boolean flag = true;

for(int start=0,end=chs.length-1;start<=end;start++,end--) {

if(chs[start]!=chs[end]) {

return flag=false;

}

}

return flag;

}

//方法实现二

public static boolean isTheSame(String s) {

//把S变成StringBuffer类型

boolean flag = new StringBuffer(s).reverse().toString().equals(s);

return flag;

}

}

**4、冒泡排序**

5**、选择排序**

/\*冒泡排序

\* （相邻数之间进行比较）

\*

\*选择排序

\* 一个索引值跟其他所有索引值进行比较

\*

\*/

public class MaoPaoDemo4 {

public static void main(String[] args) {

//定义一个字符串

String string = "bsagxod";

//定义一个新的字符串用于接收字符数组的内容 也可以用valueOf进行转换数据

String string2 = new String(bubbleSort(string));

System.out.println("排序后的字符串为："+string2);

//方法二

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

String ss = String.valueOf(choiceSort(string));

System.out.println("选择排序后的数组："+ss);

}

//方法实现 冒泡排序 （相邻数之间进行比较）

public static char[] bubbleSort(String string) {

//将字符串变成字符数组

char[] ch = string.toCharArray();

//外层控制比较次数

for(int x=0;x<ch.length-1;x++) {

//内层控制比较的数据，，，，-1是为了防止溢出

for(int y=0;y<ch.length-1-x;y++) {

if(ch[y]>ch[y+1]) {

char c = ch[y];

ch[y] = ch[y+1];

ch[y+1] = c;

}

}

}

return ch;

}

//选择排序

public static char[] choiceSort(String string) {

char[] ch = string.toCharArray();

for(int x=0;x<ch.length-1;x++) {

for(int y=x+1;y<ch.length;y++) {

if(ch[x]>ch[y]) {

char c = ch[x];

ch[x] = ch[y];

ch[y] = c;

}

}

}

return ch;

}

}

**6、高级二分查找（条件是必须有序）**

\* binaySearch二分查找

\* 思路：

\* 1、定义一个最小索引min=0，最大索引=arr.length-1;

\* 2、对中间索引进行求解mid=(min+max)/2

\* 3、判断要查找的数值与中间索引对应的数比较 (肯定要进行循环操作，并且次数未知，用while循环)

\* A；如果相等，返回中间索引值

\* B：如果不相等

\* 大了（中间索引值比要查找的索引值大），往左找

\* min不变

\* max=mid-1;

\* 重新计算mid的值

\* 小了（中间索引值比要查找的索引值小）,往右找

\* min=mid+1;

\* max不变

\* 重新计算mid的值

\* C：如果在给定数据中找不到药查找的数

\* 如何判断呢？

\* 因为min始终做加一操作，max始终做减一操作

\* 所以当min的值大于max的时候表示数据不存在，则返回-1

\*

\*/

public class binaySearchDemo5 {

public static void main(String[] args) {

int[] arr = {12,23,36,48,57,69,80};

int i =binaySearch(arr,69);

System.out.println("数据在数组中的索引位置为："+i);

}

//方法实现

public static int binaySearch(int[] arr,int value) {

//定义最大，最小索引变量

int min = 0;

int max = arr.length-1;

int mid = (min+max)/2;

while(arr[mid]!=value) {

if(arr[mid]>value) {

max = mid-1;

mid = (min+max)/2;

}else {

min = mid+1;

mid = (max+min)/2;

}

if(min>max) {

return -1;

}

}

return mid;

}

}

Arrays工具类

类型转换方法：

public static String toString( ) ;//将任意类型数据转换为String类型

数组排序：

public static void sort(); // 默认升序方式对数组进行排序

二分查找

public static int binaySearch(int[] arr ,int value); //查找value在数组中的位置

Integer包装类

构造方法：

//将int类型转换为Integer包装类

Integer ins = new Integer(int i);

//将String类型转换为Integer包装类

Integer ins = new Integer(String s);

（条件是原始的string类型的数据必须只能是数字字符组成，不能是英文字符组成）

注意：如果要将int类型或者是string类型的数据转换成integer的话

必须用构造方法，创建对象进行转换

**Integer进制的转换**

//十进制到其他进制的转换 radix只能是 0 到 26

public static String toString(int value,int raidx) ;radix表示几进制

//其他进制转换为十进制

public static int parseInt(String s ,int radix) ; 将一个其他进制换成是十进制必须要是可以转变的

**int 与String类型的转换**

//int -------String

最好用 String.valueOf();

//String -----int (Integer包装类的功能)

最好用public static int parseInt(String s)

public class Int\_StringDemo6 {

public static void main(String[] args) {

//1、将int变成包装类，包装类变成string

int i = 100;

Integer integer = new Integer(i);

System.out.println(integer.toString());

//拼接

String s = "";

s +=i;

System.out.println(s);

//String。valueOf()

String string = String.valueOf(i);

System.out.println(string);

//Integer的静态toString方法

String string2 = Integer.toString(i);

System.out.println(string2);

//2、string转换成int

String string3 = "100";

//将string转换成Integer，在到int

Integer integer2 = new Integer(string3);

int j = integer2.intValue();

System.out.println(j);

//直接转换

int k = Integer.parseInt(string3);

System.out.println(k);

}

}

character

构造方法：public character(char value)；

成员方法：

public static boolean isUpperCase (char ch);//判断字符是不是大写

public static boolean isLowerCase (char ch );//判断字符是不是小写

public static boolean isDigit(char ch) ; //判断指定字符是不是数字

public static char toUpperCase(char ch);//把指定的字符转成大写

public static char toLowerCase(char ch);//把指定的字符转成小写

**题目：**

**1、判断字符串中大写，小写，数字出现的次数**

\* 判断字符串中大写，小写，数字出现的次数

\*/

public class CountDemo7 {

public static void main(String[] args) {

//定义字符串

String string ="jacd32WC4WD0";

countNumber(string);

}

//方法实现

public static void countNumber(String string) {

//取出字符串中对应索引的字符比较

//定义统计大写，小写，数字的变量

int upper = 0;

int lower = 0;

int digit = 0;

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

char c = string.charAt(i);

if(Character.isUpperCase(c)) {

upper++;

}else if(Character.isLowerCase(c)) {

lower++;

}else if(Character.isDigit(c)) {

digit++;

}

}

System.out.println("大写字母有："+upper+" "+"小写字母有："+lower+" "+"数字有："+digit);

}

}

一、

**int----integer**

1、构造方法

public Integer(int value)

**Integer-----int**

1、public int intValue( ) //以一个int类型返回integer的值

二、

String-----Integer

1、构造方法

public Integer(String s)

2、public static Integer valueOf(string s) ;//将一个字符串对象用integer输出

I**nteger---String**

1、public String toString( );//Integer包装类的方法

对象数组：

**1、数组可以存储基本数据类型，也可以存储引用类型，它存储引用类型的时候的数组就是一个对象数组**

**2、用数存储5个学生对象，并遍历数组组**

class Student {

private String name;

private int age;

//无参构造

Student() {}

//有参构造

Student(String name,int age) {

this.name = name;

this.age = age;

}

//省略getXXX和setxxx()方法

//重写toString() 方法

public void toString() {

//override

}

}

public static void main(String[] args) {

//创建学生类

//创建集合对象 Collection集合为接口，由具体的子类的实现类去调用方法实现

Collection c = new ArrayList();

//创建学生对象

Student s1 = new Student("lili",19);

Student s2 = new Student("mmmm",21);

//将学生对象存储在集合中

c.add(s1);

c.add(s2);

//遍历集合输出

//将集合转换成数组，遍历数组输出 (此时的集合默认是Object类的对象)

Object[] obj = c.toArray();

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

Student s = (Student) obj[i]; //将Object类的数组强制转换为Student类的对象用于输出，其方法实质是多态

System.out.println(s);

}

}

**二、collection功能概述;**

功能:

1、添加功能

boolean add(Object obj) //在集合中添加一个元素

boolean addAll(Collection c) //在一个集合中添加另一个集合的所有元素

2、移除功能

void clear() //清除集合中的所有元素

boolean. remove(Object obj) //清除集合中的一个元素

Boolean. removeAll(Collection c )// 清除集合中的所有元素 (只有当所以元素被清除时才返回true)

3、长度功能

int size() //用于显示集合的长度

5、判断功能

boolean contains(Object obj)//判断集合中是否包含了目标元素，包含一个就返回true

boolean containsALL(Collection c) 集合只有包含了另一个集合的所有元素才能返回true

boolean isEmpty() 判断集合是否为空

6、把集合转换为数组

Object[] toArray()

4、迭代功能(遍历)

Iterator iterator()

用boolean hasNext()方法 判断是否有元素可以进行迭代

用Object next()方法获取每一个元素，并且自动移到下一个位置

注意:

用集合对象去创建迭代对象

迭代器对象是依赖于集合存在而存在的

（了解）

交集

boolean retain(Collection c) //调用此方法的集合是否含有形参集合的元素，，只要含有一个就会返回true

集合转换为数组

Object[ ] toArray( ) ;;将一个集合转换为数组

三、collection 集合的遍历

a:把集合转换为数组

Object[] toArray()

b:迭代器

Iterator iterator();//collection集合下的迭代器方式

四、遍历案例

**1、存储字符串并遍历**

public static void main(String[] args) {

//创建集合对象

Collection c = new ArrayList();

//创建字符串对象

String s1 = "hello";

String s2 = "java";

//将字符串对象添加到集合中

c.add(s1);

c.add(s2);

//用迭代器遍历集合

//创建迭代器对象（迭代器是一个接口，依赖于集合而存在，所以通过集合对象来创建迭代器对象，调用方法进行实现集合遍历）

Iterator it = c.iterator();

//迭代器输出时，容易抛越界异常 ，因此要进行判断集合中是否还有元素可以取出（因为元素为多个，因此用循环判断输出）

while(it.hasNext()) {

String s = (String) it.next(); // 强转为字符串类型输出

System.out.println(s)

}

/\*

//for循环遍历输出元素 (初始化对象，判断语句，第三个语句可以不写)

for(Iterator it = c.iterator();it.hasNext();) {

String s = (String) it.next(); // 强转为字符串类型输出

System.out.println(s)

}

\*/

}

**2、自定义对象遍历**

//创建学生类

class Student {

private String name;

private int age;

//无参构造

Student() {}

//有参构造

Student(String name,int age) {

this.name = name;

this.age = age;

}

//省略getXXX和setxxx()方法

//重写toString() 方法

public void toString() {

//override

}

}

public static void main(String[] args) {

//创建集合对象 Collection集合为接口，由具体的子类的实现类去调用方法实现

Collection c = new ArrayList();

//创建学生对象

Student s1 = new Student("lili",19);

Student s2 = new Student("mmmm",21);

//将学生对象存储在集合中

c.add(s1);

c.add(s2);

//通过集合对象创建迭代器对象（多态）

Iterator it = c.iterator();

//判断循环条件输出

while(it.hasNext()) {

Student s = (Student) it.next();

//Object next(); 返回的是一个Object类的对象，进行向下转型输出学生类对象，可以调用学生类特有的输出方法

System.out.println(s.getName() +" "+s.getAge());

//如果学生类重写了Object类的toString()方法则可以直接输出对象 ，否则输出包名.类名.地址值

}

}

五、迭代器Iterator方法

1、判断下一个是否有元素

boolean hasNext( ) //用于遍历时判断元素是否越界

2、获取元素并且移动到下一个位置

Object next( ); //自动返回的是一个Object类的对象

六、List的功能（索引 有序 对象内容可以重复）

1、添加（根据特定索引添加元素）

void add(int index,Object element) ; // 在特定的索引位置进行添加，可以紧跟集合最后一个元素添加，但是不能越界

2、删除

boolean remove(int index) 把指定位置的元素删除

3、修改

boolean set(int index, Object element) ;//在指定的位置，用element元素替代原有元素

4、迭代器方法

ListIterator listIterator( );

5、获取

public Object get(int index); //根据具体的索引位置获取元素

六 ---------------list集合的特有遍历方法：(List集合是有序的，且可以根据具体的索引值进行元素的操作，且对象内容可以重复)

1、size() 和 get() 方法结合

//创建List集合对象

List list = new ArrayList();

//创建字符串对象

String s1 = "hello";

String s2 = "java";

//将字符串对象添加到集合中

list.add(s1);

list.add(s2);

//遍历List集合

for(int i=0;i<list.size();i++) {

//用public Object get(int index);方法实现获取元素 （返回具体索引位置的元素）

String s = (String) list.get(i);

System.out.println(s);

}

2、迭代器方法

/创建List集合对象

List list = new ArrayList();

//创建字符串对象

String s1 = "hello";

String s2 = "java";

//将字符串对象添加到集合中

list.add(s1);

list.add(s2);

//通过List集合对象创建迭代器对象

ListIterator it = list.listIterator();

while(it.hasNext()) {

String s = (String) it.next();

System.out.println(s);

}

七、ListIterator 迭代器的使用： （遍历）

1、查询下一个元素是否存在

boolean hasNext( ); //放在while循环中用于判断条件

2、获取元素，并移动到下一个位置

Object next( );

八、数组与集合的区别

A：长度区别：

数组的长度是固定的

集合的长度是可变的

B：内容区别：

数组元素的类型可以是基本类型也可以是引用类型

集合元素的类型只能是引用类型

C：形式区别

数组中元素类型必须相同

集合元素类型可以不同

九、出现不合法的并发修改时解决方法：

1、出现现象

集合长度出现越界异常

2、原因

迭代器是依赖于集合而存在的，先有集合再有迭代器，所以当集合发生改变但是迭代器无法预知改变时，出现异常现象

3、解决方案

**A：用集合进行元素的增减或其他操作 ，并且用集合进行遍历操作**

**/创建List集合对象**

List list = new ArrayList();

//创建字符串对象

String s1 = "hello";

String s2 = "java";

String s3 = "caddo";

//将字符串对象添加到集合中

list.add(s1);

list.add(s2);

list.add(s3);

//集合修改元素

for(int i=0;i<list.size( );i++) {

String s = (String) list.get(i);

if("java".equals(s)) {

list.add("basic");

}

}

System.out.println(list); //此时添加的元素默认在集合的末尾

**B：用迭代器进行元素的增减或其他操作，且用迭代器进行遍历**

**（注意：用迭代器进行元素的改变时，只有ListIterator可以对元素进行操作，因此用ListIterator迭代器）**

public static void main(String[] args) {

//创建List集合对象

List list = new ArrayList();

//创建字符串对象

String s1 = "hello";

String s2 = "java";

String s3 = "caddo"

//将字符串对象添加到集合中

list.add(s1);

list.add(s2);

list.add(s3);

//通过集合创建迭代器对象

ListIterator it = list.listIterator();

//迭代器进行添加判断操作、并且遍历

while(it.hasNext()) {

String s = (String) list.next();

if("java".equals(s)) { //(list.contains("java")) //判断集合中是否包含某个元素

it.add("basic");

}

}

//输出

System.out.println(list); //此时通过迭代器进行添加的元素默认添加到判断元素的后面

}

十、list子类的特点

ArrayList:

底层数据结构为数组，查询快，增删慢

线程不安全，效率高

Vector:

底层数据结构为数组，查询快，增删慢

线程安全，效率低

LinkedList:

底层数据结构是链表，查询慢，增删快

线程不安全，效率高

十一、操作步骤

1、创建类

2、创建集合对象

3、创建类对象

4、迭代器遍历

A:通过集合对象创建迭代器对象（多态 向上转型）

B：迭代器对象调用方法判断是否有元素可以输出，防止越界

C：迭代器获取元素输出

向下转型输出（多态的方式，调用类的特有方法进行格式输出）

否则类重写Object类的toString() 方法以特定格式输出

十、练习题

**1、产生10个1-20之间的随机数，要求随机数不可以重复**

package dh16;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.Random;

//产生10个1-20之间的随机数，要求随机数不可以重复，并且输出这是个随机数

public class RandomText2 {

public static void main(String[] args) {

//创建集合对象用于存储随机数

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

//定义一个统计变量用于统计是否存储了十个数

int count = 0;

//创建随机数对象

Random random = new Random();

//循环判断是否进行了十次添加操作

while(count<10) {

//生成随机数nextInt(n) 表示0-n范围内int类型的随机数

int number = random.nextInt(20)+1;

//判断生成的随机数在集合中有无重复的数

if(!list.contains(number)) {

list.add(number);

count++;

}

}

//最终遍历集合输出随机数

//迭代器迭代

Iterator<Integer> it =list.iterator();

while(it.hasNext()) {

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

}

System.out.println("===============");

//List集合特有方法遍历

for(int i=0;i<list.size();i++) {

System.out.println(list.get(i));

}

System.out.println("================");

//增强for遍历

for(Integer i : list) {

System.out.println(i);

}

}

}

**2、集合的循环嵌套，用泛型实现 ,并且用增强for进行遍历**

package dh16;

import java.util.ArrayList;

//集合的循环嵌套，用泛型实现 ,并且用增强for进行遍历

public class QianTaoListText3 {

public static void main(String[] args) {

// 创建大集合

ArrayList<ArrayList<Student>> arrayList = new ArrayList<ArrayList<Student>>();

// 创建单个小集合并且存储相应的对象类型数据

ArrayList<Student> list1 = new ArrayList<Student>();

// 创建第一个集合中要进行添加的学生类对象

Student s1 = new Student("caddo", 21);

Student s2 = new Student("dada", 20);

// 添加对象到集合中

list1.add(s1);

list1.add(s2);

// 将第一个小集合添加到大集合中

arrayList.add(list1);

// 第二个小集合

ArrayList<Student> list2 = new ArrayList<Student>();

// 创建第二个集合中要进行添加的学生类对象

Student s3 = new Student("mike", 21);

Student s4 = new Student("lili", 20);

// 添加对象到集合中

list1.add(s3);

list1.add(s4);

// 将第二个小集合添加到大集合中

arrayList.add(list2);

// 第三个小集合

ArrayList<Student> list3 = new ArrayList<Student>();

// 创建第二个集合中要进行添加的学生类对象

Student s5 = new Student("jack", 21);

Student s6 = new Student("liu", 20);

// 添加对象到集合中

list1.add(s5);

list1.add(s6);

// 将第二个小集合添加到大集合中

arrayList.add(list3);

//遍历

//外层循环对大集合尽心遍历，

for(ArrayList<Student> x : arrayList) {

for(Student y : x) {

System.out.println(y.getName()+" "+ y.getAge());

}

}

}

}

**//创建自定义类实现默认自定义栈**

package dh16;

import java.util.LinkedList;

public class MyLinkedListText6 {

//定义集合对象为私有，在其他类中不能通过创建对象来使用

private LinkedList link;

public MyLinkedListText6() {

//通过创建自定义对象实际上创建的是LinkedList的对象

link = new LinkedList();

}

public void add(Object obj) {

link.addFirst(obj);

}

public Object get() {

return link.removeFirst();

//每次移除第一个，并且返回第一个对象

}

public boolean isEmpty() {

return link.isEmpty();

}

}

**4、删除ArrayList集合中重复的元素**

package dh16;

//删除List集合中重复出现的元素

//方法一、定义新的集合，进行元素添加判断

//注意，判断是否重复使用contains时，必须注意，其实使用的是equals()比较是否相同

//自定义类中没有equals（）方法，则默认使用object类的，若要使用本身的，应该对方法进行重写

import java.util.ArrayList;

public class RemoveSameText5 {

public static void main(String[] args) {

// 创建集合对象

ArrayList<Student> list = new ArrayList<Student>();

// 创建学生对象

Student s1 = new Student("林", 21);

Student s2 = new Student("刘", 20);

Student s3 = new Student("游", 21);

Student s4 = new Student("林", 21);

Student s5 = new Student("吴", 22);

Student s6 = new Student("刘", 20);

// 将学生对象添加到集合中

list.add(s1);

list.add(s2);

list.add(s3);

list.add(s4);

list.add(s5);

list.add(s6);

//方法一、定义新集合

//创建新集合

ArrayList<Student> l = new ArrayList<Student>();

//遍历原集合进行判断是否在新集合中存在相同的元素

for(int i=0;i<list.size();i++) {

if(!l.contains(list.get(i))) {

l.add(list.get(i));

}

}

//遍历新集合

for(Student s : l) {

System.out.println(s.getName()+"\t"+s.getAge());

}

//方法二、选择排序

for(int i=0;i<list.size()-1;i++) {

for(int j=i+1;j<list.size();j++) {

if(list.get(i).equals(list.get(j))) {

list.remove(j);

}

}

}

System.out.println("===================");

for(Student ss : list) {

System.out.println(ss.getName()+"\t"+ss.getAge());

}

}

**//键盘录入数据，当录入0的时候结束，要求在控制台输出这些数据的最大值，并且将所有数据(如果是数组最终已字符串形式输出)进行输出**

package dh16;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

public class ScannerText4 {

public static void main(String[] args) {

//创建集合对象

ArrayList<Integer> array = new ArrayList<Integer>();

//键盘录入对象

Scanner sc = new Scanner(System.in);

//循环判断键盘录入数据是否为0

while(true) {

System.out.println("请输入数据:");

int number = sc.nextInt();

if(number==0) {

break;

}else {

array.add(number);

}

}

//将集合排序

//将集合进行转换成数组，并且排序

Object[] obj = array.toArray();

Arrays.sort(obj);

//用特定形式输出数组

String s = ScannerText4.arrayToString(obj);

System.out.println("以特定形式返回的数据为："+s);

System.out.println("录入数据中最大值为："+obj[obj.length-1]);

/\* //遍历数组

for(Object ob : obj) {

System.out.println(ob);

}\*/

}

/\*\*

\* 一某种特定形式输出数组

\* @param obj

\* 目标数组

\* @return

\* 特定形式的字符串

\*/

public static String arrayToString(Object[] obj) {

//将数组转换成特定形式，采用StringBuffer或者StringBuilder进行字符串添加

StringBuilder builder = new StringBuilder();

builder.append("[");

//遍历数组

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

if(i==obj.length-1) {

builder.append(obj[i]).append("]");

}else {

builder.append(obj[i]).append(",");

}

}

return builder.toString();

}

}

**七、练习**

**1、（编写一个程序，获取10个1-20之间的数，且不能有重复）**

package dh17;

import java.util.HashSet;

import java.util.Random;

//获取10个1-20内的无重复的随机数

public class RandomText1 {

public static void main(String[] args) {

//创建集合对象

HashSet<Integer> hashSet = new HashSet<Integer>();

//创建随机数对象

Random random = new Random();

//循环判断次数，并且进行元素添加

//因为set集合元素唯一性，所以无法通过定义统计变量实现操作

//只能通过判断集合最后的长度来判断是否有10个数

while(hashSet.size()<10) {

int number = random.nextInt(20)+1;

hashSet.add(number);

}

for(Integer i : hashSet) {

System.out.println(i);

}

}

}

1. **键盘录入3次学生的信息（姓名，数学成绩，英语成绩，语文成绩），根据总分进行从高到低排序**

//键盘录入3个学生的信息（姓名，数学成绩，英语成绩，语文成绩），根据总分进行从高到低排序

/\*分析：

\* 1、键盘录入学生信息，所以创建学生类（姓名，数学成绩，英语成绩，语文成绩）

\* 2、创建对象录入成绩，姓名，存储到学生对象中

\* 3、因为要输出，所以使用集合，又因为要排序，所以使用TreeSet

\* 4、将每个对象存储到集合中

\* 5、因为依靠总分进行排序，因此实现总分的计算

\* 6、又因为此集合不能有重复的元素出现，所以要实现接口，并重写compare()方法

\*

\*/

package dh17;

public class Student {

private String name;

private int mathScore;

private int englishScore;

private int chineseScore;

public Student() {

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getMathScore() {

return mathScore;

}

public void setMathScore(int mathScore) {

this.mathScore = mathScore;

}

public int getEnglishScore() {

return englishScore;

}

public void setEnglishScore(int englishScore) {

this.englishScore = englishScore;

}

public int getChineseScore() {

return chineseScore;

}

public void setChineseScore(int chineseScore) {

this.chineseScore = chineseScore;

}

public int sum() {

return this.mathScore+this.chineseScore+this.chineseScore;

}

}

package dh17;

import java.util.Comparator;

import java.util.Scanner;

import java.util.TreeSet;

public class SumScoreSort {

public static void main(String[] args) {

//创建集合有参构造的对象

TreeSet<Student> set = new TreeSet<Student>(new Comparator<Student>() {

@Override

public int compare(Student s1, Student s2) {

//比较总分是否相同(从高到低，后面减前面)

int num1 = s2.sum()-s1.sum();

//判断是否名字相同

int num2 = num1==0 ? s2.getName().compareTo(s1.getName()) : num1;

//判断数学成绩是否相同

int num3 = num2==0 ? s2.getMathScore()-s1.getMathScore() : num2;

//判断英语成绩是否相同

int num4 = num3==0 ? s2.getChineseScore()-s1.getChineseScore() : num3;

return num4;

}

});

//键盘录入

while(set.size()<3) {

Scanner sc = new Scanner(System.in);

System.out.print("请录入学生姓名：");

String name = sc.nextLine();

System.out.print("请录入数学成绩：");

int mathScore = sc.nextInt();

System.out.print("请录入英语成绩：");

int englishScore = sc.nextInt();

System.out.print("请录入语文成绩：");

int chineseScore = sc.nextInt();

//创建学生对象，将键盘录入的信息存储到学生对象中

Student stu = new Student();

stu.setName(name);

stu.setMathScore(mathScore);

stu.setEnglishScore(englishScore);

stu.setChineseScore(chineseScore);

//将学生对象存储在集合中

set.add(stu);

}

System.out.println("姓名\t总分\t数学成绩\t英语成绩\t语文成绩");

//遍历输出

for(Student s : set) {

System.out.println(s.getName()+"\t"+s.sum()+"\t"+s.getMathScore()+"\t"+s.getEnglishScore()+"\t"+s.getChineseScore());

}

}

}

**3、自定义对象存储并且遍历，要求元素唯一，并且按某种规则进行排序，按照姓名长度排序**

package dh17;

public class Student implements Comparable<Student>{

private String name;

private int age;

public Student() {

}

public Student(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

@Override

public int compareTo(Student s) {

//根据姓名长度判断

int num1 = this.getName().length()-s.getName().length();

//根据名字是否一样

int num2 = num1==0 ? this.getName().compareTo(s.getName()) : num1;

//根据年龄是否一样

int num3 = num2==0 ? this.getAge()-s.getAge() : num2;

return num3;

}

}

package dh17;

import java.util.TreeSet;

//根据学生姓名长度进行排序(采用自然排序)

public class TreeSetSortText2 {

public static void main(String[] args) {

// 创建集合对象

TreeSet<Student> set = new TreeSet<Student>();

// 创建学生对象

Student s1 = new Student("mike", 25);

Student s2 = new Student("jack", 21);

Student s3 = new Student("caddo", 21);

Student s4 = new Student("kkkkk", 25);

Student s5 = new Student("sss", 20);

Student s6 = new Student("mik", 25);

Student s7 = new Student("kkkkk", 25);

// 将学生对象添加到集合中

set.add(s1);

set.add(s2);

set.add(s3);

set.add(s4);

set.add(s5);

set.add(s6);

for(Student s : set) {

System.out.println(s.getName()+"\t"+s.getAge());

}

}

}

**//统计字符串中字符出现的次数**

**//需求”aababcabcdabcde“，获取字符串中每一个字符出现的次数，要求结果a(5)b(4)c(3)d(2)e(1)**

**/\*分析：**

package dh18;

import java.util.Scanner;

import java.util.Set;

import java.util.TreeMap;

\* 1、定义一个字符串

\* 2、将字符串转换为字符数组

\* 3、遍历字符数组的到每一个字符

\* 4、定义一个Map集合

\* 5、拿这个字符到Map集合里找，看返回值

\* 是null；就把字符作为键，1作为初始元素值存储

\* 不是null ， 值就++；然后重新将值进行存储

\* 6、定义一个字符串缓冲区

\* 7、遍历集合 ，获取每一个键值和元素值进行拼接（输出对应键值所对应的元素值）

\* 8、把字符串缓冲区转换为字符串输出

\*

\*/

public class TongJiText1 {

public static void main(String[] args) {

//定义一个字符串

Scanner sc = new Scanner(System.in);

System.out.println("请输入要进行判断的字符串:");

String s = sc.nextLine();

//创建集合对象

TreeMap<Character, Integer> map = new TreeMap<Character,Integer>();

//将字符串转换为字符数组

char[] chs = s.toCharArray();

//遍历字符数组,判断键是否存在于集合中

for(char ch : chs) {

//把每一个字符获取出来

Integer i = map.get(ch);

//是null；就把字符作为键，1作为初始元素值存储

//null表示的是键值所对应的元素值是否为空，元素值为空则添加对应键值/

//如果不存在就直接替换之前的元素值

if(i==null) {

map.put(ch, 1);

}else {

//不是null ， 值就++；然后重新将值进行存储

i++;

map.put(ch,i);

}

}

//定义字符串缓冲区

StringBuilder bd = new StringBuilder();

//遍历集合

//将所有键值存存到集合中

Set<Character> set = map.keySet();

for(Character key : set) {

bd.append(key).append("(").append(map.get(key)).append(")");

}

//将结果以字符串形式输出

String result = bd.toString();

System.out.println("result:"+result);

}

}

//ArrayList集合存储自定义对象并且进行排序

**package** dh18;

**public** **class** **Student** **implements** Comparable<Student> {

**private** **String** name;

**private** **int** age;

**private** **String** number;//学号

**public** **Student**() {

}

**public** **Student**(**String** name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** **Student**(**String** number) {

**this**.number = number;

}

**public** **String** **getNumber**() {

**return** number;

}

**public** **void** **setNumber**(**String** number) {

**this**.number = number;

}

**public** **String** **getName**() {

**return** name;

}

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

**this**.name = name;

}

**public** **int** **getAge**() {

**return** age;

}

**public** **void** **setAge**(**int** age) {

**this**.age = age;

}

***@Override***

**public** **int** **hashCode**() {

**final** **int** **prime** = 31;

**int** **result** = 1;

result = prime \* result + ((number == **null**) ? 0 : number.hashCode());

**return** result;

}

***@Override***

**public** **boolean** **equals**(**Object** obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

**Student** **other** = (**Student**) obj;

**if** (number == **null**) {

**if** (other.number != **null**)

**return** **false**;

} **else** **if** (!number.equals(other.number))

**return** **false**;

**return** **true**;

}

***@Override***

**public** **int** **compareTo**(**Student** s) {

//根据年龄进行排序

**int** **num1** = **this**.age-s.age;

**int** **num2** = (num1==0) ? **this**.name.compareTo(s.number) : num1;

**return** num2;

}

/\*

\*

public int compareTo(Student s) {

//根据键值的大小进行排序

//因为键值是一个字符串类型，而返回的需要比较的是int类型，所以将String类型转换为int类型

int num = Integer.parseInt(this.number) - Integer.parseInt(s.number);

return num;

}

\*/

package dh18;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

//ArrayList集合存储自定义对象并且进行排序

public class ArrayListSortText1 {

public static void main(String[] args) {

// 创建集合对象

List<Student> array = new ArrayList<Student>();

// 创建学生对象

Student s1 = new Student("刘大大", 21);

Student s2 = new Student("游大大", 20);

Student s3 = new Student("可乐", 22);

//将学生对象添加到集合

array.add(s1);

array.add(s2);

array.add(s3);

//排序

Collections.sort(array);

//遍历集合

for(Student stu : array) {

System.out.println(stu.getName()+"\t"+stu.getAge());

}

}

}

需求：HashMap 嵌套 HashMap 再嵌套ArrayList集合

package dh18;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Set;

/\*需求：HashMap 嵌套 HashMap 再嵌套ArrayList集合

\* gz 广州校区

\* jc 基础班

\* 王 21

\* 李 25

\* xa 西安校区

\* jy 就业班

\* 刘 21

\* 程 22

\* bj 北京校区

\* cz 传播客

\* 游 20

\* 赵 22

\*/

public class QianTaoText1 {

public static void main(String[] args) {

//创建大的集合对象

HashMap<String, HashMap<String,ArrayList<Student>>> map = new HashMap<String, HashMap<String,ArrayList<Student>>>();

//创建学生对象

Student s1 = new Student("王",21);

Student s2 = new Student("李",25);

Student s3 = new Student("刘",21);

Student s4 = new Student("程",22);

Student s5 = new Student("游",20);

Student s6 = new Student("赵",22);

//创建第一个ArrayList集合对象

ArrayList<Student> array1 = new ArrayList<Student>();

array1.add(s1);

array1.add(s2);

//创建第二个ArrayList集合对象

ArrayList<Student> array2 = new ArrayList<Student>();

array2.add(s3);

array2.add(s4);

//创建第三个ArrayList集合对象

ArrayList<Student> array3 = new ArrayList<Student>();

array3.add(s5);

array3.add(s6);

//创建第一个小的HashMap对象

HashMap<String,ArrayList<Student> > m1 = new HashMap<String,ArrayList<Student> >();

m1.put("jc", array1);

//创建第二个小的HashMap对象

HashMap<String,ArrayList<Student> > m2 = new HashMap<String,ArrayList<Student> >();

m2.put("jy", array2);

//创建第三个小的HashMap对象

HashMap<String,ArrayList<Student> > m3 = new HashMap<String,ArrayList<Student> >();

m3.put("cz", array3);

//在大集合中存储键值对

map.put("gz", m1);

map.put("xa", m2);

map.put("bj", m3);

//遍历

//将大集合的键值存储在集合中

Set<String> maxkeyset = map.keySet();

for(String maxkey : maxkeyset) {

//因为大集合的元素值是一个HashMap的集合，所有用集合对象接收

HashMap<String, ArrayList<Student>> minmap = map.get(maxkey);

//将小集合的键值存储在set集合中

Set<String> minkeyset = minmap.keySet();

for(String minkey : minkeyset) {

//又因为小集合的元素值是一个ArrayList的集合，所有用ArrayList集合对象接收

ArrayList<Student> list = minmap.get(minkey);

for(Student stu : list) {

System.out.println(maxkey+"\t"+minkey+"\t"+stu.getName()+"\t"+stu.getAge());

}

}

}

}

}

//HashMap嵌套ArrayList集合

/\*HashMap的集合元素是一个ArrayList集合

\*需求;有三个键值对,每个键值对应的ArrayList集合中有三个元素

\*已知：：

\* 三国演义

\* 武松 , 周瑜

\* 笑傲江湖

\* 令狐冲 , 林平之

\* 神雕侠侣

\* 杨过 , 小龙女

\*/

package dh18;

import java.util.ArrayList;

import java.util.Set;

import java.util.TreeMap;

public class HashMapToArrayListDemo1 {

public static void main(String[] args) {

// 创建大的集合对象

TreeMap<String, ArrayList<String>> map = new TreeMap<String, ArrayList<String>>();

// 创建第一个ArrayList集合

ArrayList<String> array1 = new ArrayList<String>();

array1.add("武松");

array1.add("周瑜");

// 创建第二个ArrayList集合

ArrayList<String> array2 = new ArrayList<String>();

array2.add("令狐冲");

array2.add("林平之");

// 创建第三个ArrayList集合

ArrayList<String> array3 = new ArrayList<String>();

array3.add("杨过 ");

array3.add("小龙女 ");

//在大集合中存储键值对

map.put("三国演义", array1);

map.put("笑傲江湖", array2);

map.put("神雕侠侣", array3);

//遍历集合

Set<String> maxkeyset = map.keySet();

for(String maxkey : maxkeyset) {

ArrayList<String> minlist = map.get(maxkey);

for(String min : minlist) {

System.out.println(maxkey+"\t"+min);

}

}

}

}

//1、HashMap嵌套HashMap

package dh18;

import java.util.HashMap;

import java.util.Set;

//1、HashMap嵌套HashMap

public class HashMapToHashMapDemo1 {

public static void main(String[] args) {

//创建集合嵌套对象大

HashMap<String, HashMap<String,Integer>> cz = new HashMap<String, HashMap<String,Integer>>();

//创建小集合对象

HashMap<String, Integer> jc = new HashMap<String, Integer>();

//创建第二个小集合对象

HashMap<String, Integer> jy = new HashMap<String, Integer>();

//在集合中添加键值对

jc.put("水星记", 21);

jc.put("Caddo", 22);

//在第二个小集合中添加键值对

jy.put("可乐", 23);

jy.put("cheng", 21);

//在大集合中添加键值对

cz.put("jc", jc);

cz.put("jy", jy);

//遍历集合

//将大集合的键值放在一个集合中

Set<String> czkeyset = cz.keySet();

for(String czkey : czkeyset) {

//大集合的元素值是一个集合对象，定义集合对象进行接收

HashMap<String,Integer> minmap = cz.get(czkey);//此处取出的是一个集合对象

//将小集合的键值放在集合中

Set<String> minkeymap = minmap.keySet();

for(String minkey : minkeymap) {

Integer minvalue = minmap.get(minkey);

System.out.println(minkey+"\t"+minvalue);

}

}

}

}

登录注册功能

**package** DengLu.cn.itcast.pojo;

/\*\*

\* 用户基本描述类

\* **@author** 老腰

\*

\*/

**public** **class** **User** {

**private** **String** UserName;//姓名

**private** **String** password;//年龄

**public** **User**() {

}

**public** **String** **getUserName**() {

**return** UserName;

}

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

**this**.UserName = UserName;

}

**public** **String** **getPassword**() {

**return** password;

}

**public** **void** **setPassword**(**String** password) {

**this**.password = password;

}

package DaoFunction;

import java.util.ArrayList;

import java.util.Scanner;

import DengLu.cn.itcast.pojo.User;

/\*\*

\* 用户功能类

\* @author 老腰

\*

\*/

public class UserFunction {

/\*//创建集合存储数据

\* 为了避免数据被外部使用，因此使用private修饰

\* 并且为了是注册和登录为同一个集合上的操作，因此定义集合变量为静态的

\*/

private static ArrayList<User> array = new ArrayList<User>();

/\*\*

\* 用户注册

\* @param user

\* 用户注册对象

\*/

public void regist(User user) {

//将用户注册对象信息存储在集合中

array.add(user);

}

/\*\*

\* 用户登录功能

\* @param userName

\* 键入的用户账号

\* @param userPassword

\* 键入的用户密码

\* @return 登录是否成功的结果

\*/

public boolean isLogin(String userName,String userPassword) {

boolean flag = false;

for(User u : array) {

if(u.getUserName().equals(userName)&&u.getPassword().equals(userPassword)) {

flag = true;

//只要有一个用户信息匹配就表明登录成功

break;

}

}

return flag;

}

public static void startGame() {

//产生一个随机数

int number = (int) (Math.random()\*100)+1;

//定义变量用于存储猜的次数

int count = 0;

while(true) {

//键入数据

Scanner sc = new Scanner(System.in);

System.out.println("请输入要进行猜测的数据:");

int guessNumber = sc.nextInt();

//只要输一次就统计一次

count++;

if(guessNumber>number) {

System.out.println("所猜的数"+guessNumber+"大了");

}else if (guessNumber<number) {

System.out.println("所猜的数"+guessNumber+"小了");

}else {

System.out.println("恭喜你在第"+count+"次猜对了");

break;

}

}

}

/\*

//简单测试类，测试猜数字游戏是否正确

public static void main(String[] args) {

UserFunction function = new UserFunction();

function.startGame();

}\*/

}

package DaoImpl;

import java.util.Scanner;

import DaoFunction.UserFunction;

import DengLu.cn.itcast.pojo.User;

/\*\*

\* 用户操作类，实现主界面，注册，登录跳转

\* @author 老腰

\*/

public class UserDao {

//主界面

public void startMenu() {

System.out.println("--- ------欢迎使用-----------");

System.out.println("1: 注册");

System.out.println("2: 登录");

System.out.println("3: 退出");

System.out.print("请输入选择:");

//键盘录入主界面选项

Scanner sc = new Scanner(System.in);

String choice = sc.nextLine();

//switch循环

switch(choice) {

case "1":

System.out.println("--------欢迎进入用户注册界面--------");

System.out.print("请输入用户注册账号名:");

String newUserName = sc.nextLine();

System.out.print("请输入用户注册密码:");

String newUserPassword = sc.nextLine();

//调用功能

//首先必须创建用户对象

User user = new User();

user.setName(newUserName);

user.setPassword(newUserPassword);

UserFunction ud = new UserFunction();

ud.regist(user);

//当注册完成后重新跳转到主界面

System.out.println("注册成功!");

startMenu();

break;

case "2" :

//定义变量设置登录的次数，超过次数则强制退出

int count = 0;

while(count<3) {

System.out.println("--------欢迎进入用户登录界面--------");

System.out.print("请输入用户登录账号名:");

String userName = sc.nextLine();

System.out.print("请输入用户登录密码:");

String userPassword = sc.nextLine();

//输入一次统计变量就自加

count++;

//创建对象调用登录方法

UserFunction us = new UserFunction();

//方法调用后其实是一个布尔值，根据布尔值结果自主判断是否登录成功

boolean flag = us.isLogin(userName, userPassword);

if(flag) {

System.out.println("登录成功!");

System.out.println("--------欢迎进入猜数字游戏界面--------");

System.out.println("你玩吗？y/n");

//多次输入进行游戏

//String string = sc.nextLine();

while(true) {

String string = sc.nextLine();

if(string.equalsIgnoreCase("y")) {

UserFunction.startGame();

System.out.println("你还玩吗？y/n");

}else if(string.equals("n")){

System.out.println("用户退出游戏!");

}else {

System.out.println("输入的选项有误，请重新输入。");

continue;

}

}

}else {

if(count==3) {

System.out.println("机会已经用尽，请与客服进行联系。");

System.exit(0);//强制退出系统

}

else {

System.out.println();

System.out.println("用户名或密码错误，请重新输入");

System.out.println("你还有"+(3-count)+"次机会");

System.out.println();

//回到主菜单重新输入

//startMenu();

continue;

}

}

}

break;

case "3":

System.out.println("用户退出系统，欢迎下次使用！");

System.exit(0);//终止虚拟机

break;

default :

System.out.println("你输入的选项有误，请重新输入!");

startMenu();//回到主界面重新输入选项判断

break;

}

}

}

**package** DaoText;

**import** DaoImpl.UserDao;

/\*\*

\* 用户功能测试类

\* **@author** 老腰

\*/

**public** **class** **UserText** {

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

**UserDao** **dao** = **new** UserDao();

dao.startMenu();

}

}

}

效果











