上周作业：

Lyric

**import** java.io.\*;

**import** java.util.\*;

**import** acm.program.ConsoleProgram;

**import** acm.util.\*;

**public** **class** Lyric **extends** ConsoleProgram{

**public** BufferedReader openFile(String prompt){

BufferedReader rd=**null**;

**while**(rd==**null**){

**try**{

String name=readLine(prompt);

rd=**new** BufferedReader(**new** FileReader(name));

}

**catch**(IOException ex){

println("文件不存在！");

}

}

**return** rd;

}

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

BufferedReader rd=openFile("请输入文件名：");

ArrayList lyrics=**new** ArrayList();

**int** countline=0;

**int** countword=0;

**try**{

PrintWriter pw=**new** PrintWriter("E:\\gc.txt");

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

String line=rd.readLine();

countline++;

**if**(line==**null**)

**break**;

**else** **if**(!lyrics.contains(line)){

lyrics.add(line);

countword++;

println(line);

pw.println(line);

}

}

pw.println("共包含行数："+countline);

pw.println("共包含不同词："+countword);

pw.close();

rd.close();

}

**catch**(IOException ex){

**throw** **new** ErrorException(ex);

}

}

}

CountCNum

**import** java.io.\*;

**import** acm.program.\*;

**import** acm.util.\*;

**public** **class** CountCNum **extends** ConsoleProgram{

**public** BufferedReader openFile(String prompt){

setFont("宋体-50");

BufferedReader rd=**null**;

**while**(rd==**null**){

**try**{

String name=readLine(prompt);

rd=**new** BufferedReader(**new** FileReader(name));

}

**catch**(IOException ex){

println("文件不存在！");

}

}

**return** rd;

}

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

BufferedReader rd=openFile("请输入一个文件名：");

String s=readLine("请输入一个要查询的字符串：");

**try**{

**int** count=0;

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

String line=rd.readLine();

**if**(line==**null**)

**break**;

**int** start =0;

**while**((start=line.indexOf(s,start))>=0){

start += s.length();

count++;

}

}

println(count);

println("完成");

rd.close();

}**catch**(IOException ex){

**throw** **new** ErrorException(ex);

}

}

}

LyricSort

/\*

\* 本程序由CodingCat小朋友编写（修改版，适合同学们理解）

\* 因课程时间关系，没有办法给同学们讲解每一个方法和类

\* 课程省略了部分内容，感兴趣的同学可自己学习

\* 包括Comparator接口，方法的重写，泛型等内容

\* 需要哪个方面的内容，就去搜哪方面的知识，多看相关博客

\*/

**import** java.io.\*;

**import** java.util.\*;

**import** acm.program.\*;

//歌词类，用于存放词以及词的个数

**class** LyricWord{

**public** LyricWord(){

word="";

num=0;

}

**public** String getWord(){

**return** word;

}

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

**return** num;

}

**public** **void** addnum(){

num++;

}

**public** **void** setWord(String w){

word=w;

num=1;

}

**public** String toString()

{

**return** word+" "+num;

}

**private** String word="";

**private** **int** num=0;

}

**public** **class** LyricSort **extends** ConsoleProgram{

//主方法

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

//在java中,如果要对集合对象或数组对象进行排序,需要实现Comparator接口以达到我们想要的目标。

Comparator<LyricWord> comparator = **new** Comparator<LyricWord>(){

**public** **int** compare(LyricWord A1, LyricWord A2) {

**if**(A1.getNum()!=A2.getNum()){**return** A1.getNum()-A2.getNum();}

**else**{

**return** A1.getWord().compareTo(A2.getWord());

}

}

};

//声明一个ArrayList来存放歌词

ArrayList<LyricWord> lw=**new** ArrayList<LyricWord>();

BufferedReader rd=openFile("请输入一个文件名：");

**try** {

PrintWriter pw;

pw=**new** PrintWriter("f:\\gc.txt");

**boolean** Flag = **true**;//声明一个标记值

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

String line = rd.readLine();

LyricWord w=**new** LyricWord();

Flag = **true**;

**if**(line==**null**)

**break**;

//判断读取的词是否已经存在歌词列表中

**for**(LyricWord c:lw){

**if**(line.equals(c.getWord())){

Flag = **false**;//如果存在，设置Flag为false，并增加本次词的个数

c.addnum();

**break**;//只要存在就查找循环

}

}

//判断标记值是否为真，如果为真，即!Flag为true，不执行下面的命令，直接继续下一次循环。

//如果为false，执行下面的语句，将新词加入到ArrayList中

**if**(!Flag)

**continue**;

w.setWord(line);

lw.add(w);

}

//调用Collections.sort命令，对歌词列表lw按照comparator的方式排序

Collections.*sort*(lw,comparator);

//个数从高到低排序

**for**(**int** i = lw.size()-1;i>=0;i--)

{

pw.println(lw.get(i));

println(lw.get(i));

}

rd.close();

pw.close();

}

**catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

//文档读取方法

**public** BufferedReader openFile(String prompt){

BufferedReader rd=**null**;

**while**(rd==**null**){

**try**{

String name=readLine(prompt);

rd=**new** BufferedReader(**new** FileReader(name));

}

**catch**(IOException ex){

println("文件不存在！");

}

}

**return** rd;

}

}

**Class10 课堂实例**

HashMapUse

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Set;

**import** java.util.Map.Entry;

**import** acm.program.ConsoleProgram;

**public** **class** HashMapUse **extends** ConsoleProgram{

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

HashMap<String, String> hashMap = **new** HashMap<String, String>();

hashMap.put("cn","中国");

hashMap.put("us","米国");

hashMap.put("en","英国");

println(hashMap);

println("cn"+hashMap.get("cn"));

println(hashMap.containsKey("cn"));

println(hashMap.keySet());

println(hashMap.isEmpty());

//hashMap.remove("cn");

//println(hashMap.containsKey("cn"));

//使用Entry遍历HashMap（首选，代码短，效率高，速度快）

Set<Entry<String, String>> sets = hashMap.entrySet();

**for**(Entry<String, String> entry : sets){

println(entry.getKey()+", ");

println(entry.getValue());

}

//使用Iterator遍历HashMap

Iterator<String> it = hashMap.keySet().iterator();

**while**(it.hasNext()){

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

println("key:"+key);

println("value:"+hashMap.get(key));

}

//使用Iterator+Entry遍历HashMap

Iterator<Entry<String, String>> it1 =hashMap.entrySet().iterator();

**while**(it1.hasNext()){

Entry<String, String> m=(Entry<String, String>)it1.next();

println("键：" + m.getKey() + "，值:" + m.getValue());

}

}

}

**interface**

**更多关于接口的信息http://blog.sina.com.cn/s/blog\_4adaacb7010006kz.html**

**interface** Runner //定义接口

{

**int** ***i***=3;

**public** **void** start();

**void** run();

**void** stop();

}

**interface** Eater **extends** Runner //接口间可以继承

{

**public** **final** **static** **int** ***j***=4;

**void** openMouth();

**void** upAndDown();

**void** goIn();

}

**class** TT **implements** Eater //引用接口

{

**public** **void** start()

{

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

}

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

{

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

}

**public** **void** stop()

{

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

}

**public** **void** openMouth()

{

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

}

**public** **void** upAndDown()

{

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

}

**public** **void** goIn()

{

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

}

}

**public** **class** TestInterface

{

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

{

Runner tt=**new** TT();//接口的引用指向实现的对象

System.***out***.println(tt.***i***);

System.***out***.println(Runner.***i***);

tt.start();

Eater ee=**new** TT();

System.***out***.println(ee.***j***);

System.***out***.println(Eater.***j***);

ee.start();

}

}

FirstButton

**import** java.awt.event.\*;

**import** javax.swing.\*;

**import** acm.program.\*;

**public** **class** FirstButton **extends** ConsoleProgram{

**public** **void** init(){

setFont("宋体-50");

JButton bt=**new** JButton("hi");

add(bt,***SOUTH***);

addActionListeners();

}

**public** **void** actionPerformed(ActionEvent e){

String cmd=e.getActionCommand();

**if**(cmd.equals("hi")){

println("hello world");

}

}

}

ThreeButtons

**import** java.awt.event.ActionEvent;

**import** javax.swing.JButton;

**import** acm.program.ConsoleProgram;

**public** **class** ThreeButtons **extends** ConsoleProgram{

**public** **void** init(){

setFont("宋体-50");

JButton bt=**new** JButton("hello");

JButton bt1=**new** JButton("java class");

add(bt,***SOUTH***);

add(bt1,***SOUTH***);

add(**new** JButton("welcome"),***EAST***);

addActionListeners();

}

**public** **void** actionPerformed(ActionEvent e){

String cmd=e.getActionCommand();

**if**(cmd.equals("hello")){

println("大家好");

}

**else** **if**(cmd.equals("java class")){

println("java基础课程");

}

**else** **if**(cmd.equals("welcome")){

println("欢迎你！");

}

}

}

InteractiveDrawHP

**import** acm.program.\*;

**import** acm.graphics.\*;

**import** java.awt.\*;

**import** javax.swing.\*;

**public** **class** InteractiveDrawHP **extends** GraphicsProgram {

**public** **void** init() {

//清除界面

add(**new** JButton("重置"), ***SOUTH***);

// 检查选择的是证明还是北面

checkbox = **new** JCheckBox("正面");

checkbox.setSelected(**true**);

add(checkbox, ***SOUTH***);

initRadioButtons();

initColorChooser();

// 获得鼠标动作

addMouseListeners();

// 获得交互动作

addActionListeners();

}

**private** **void** initRadioButtons() {

// 控制大小的单选按钮

ButtonGroup sizeBG = **new** ButtonGroup();

smallRB = **new** JRadioButton("小");

medRB = **new** JRadioButton("中");

largeRB = **new** JRadioButton("大");

// 将单选按钮添加到界面上

sizeBG.add(smallRB);

sizeBG.add(medRB);

sizeBG.add(largeRB);

// 设定初始选择的单选按钮

medRB.setSelected(**true**);

// 将单选按钮添加到页面上

add(smallRB, ***SOUTH***);

add(medRB, ***SOUTH***);

add(largeRB, ***SOUTH***);

}

**private** **void** initColorChooser() {

// 制作控制颜色的下拉框

pickColor = **new** JComboBox();

pickColor.addItem("Black");

pickColor.addItem("Blue");

pickColor.addItem("Green");

pickColor.addItem("Red");

// 下拉框不允许用户编辑

pickColor.setEditable(**false**);

// 设定初始选择颜色

pickColor.setSelectedItem("Black");

// 为下拉框制作显示标签，与之前添加的组件用空格分隔

add(**new** JLabel(" Color:"), ***SOUTH***);

// 将下拉框添加到界面

add(pickColor, ***SOUTH***);

}

// 根据单选框返回图像大小

**private** **double** getDiamSize() {

**double** size = 0;

**if** (smallRB.isSelected()) {

size = ***SMALL\_DIAM***;

} **else** **if** (medRB.isSelected()) {

size = ***MED\_DIAM***;

} **else** **if** (largeRB.isSelected()) {

size = ***LARGE\_DIAM***;

}

**return** size;

}

// 根据下拉框返回颜色

**private** Color getCurrentColor() {

String name = (String) pickColor.getSelectedItem();

**if** (name.equals("Blue")) {

**return** Color.***BLUE***;

} **else** **if** (name.equals("Green")) {

**return** Color.***GREEN***;

} **else** **if** (name.equals("Red")) {

**return** Color.***RED***;

} **else** **return** Color.***BLACK***;

}

// 用户点击鼠标时执行以下程序

**public** **void** mouseClicked(MouseEvent e) {

GObject obj;

**double** diam = getDiamSize();

**if** (checkbox.isSelected()) {

obj = **new** HalloweenPumpkin(diam, diam,getCurrentColor());

} **else** {

obj = **new** GOval(diam, diam);

}

add(obj, e.getX(), e.getY());

}

// 当点击按钮时执行以下程序

**public** **void** actionPerformed(ActionEvent e) {

**if** (e.getActionCommand().equals("Clear")) {

removeAll(); // 清除画布中的一切元素

}

}

/\* 私有常量 \*/

**private** **static** **final** **double** ***SMALL\_DIAM*** = 50;

**private** **static** **final** **double** ***MED\_DIAM*** = 80;

**private** **static** **final** **double** ***LARGE\_DIAM*** = 100;

/\* 私有实例变量 \*/

**private** JCheckBox checkbox;

**private** JRadioButton smallRB;

**private** JRadioButton medRB;

**private** JRadioButton largeRB;

**private** JComboBox pickColor;

}

HalloweenPumpkin类

**import** java.awt.Color;

**import** acm.graphics.\*;

/\*

\* 圣诞节南瓜头类

\* 集成GCompound

\*/

**public** **class** HalloweenPumpkin **extends** GCompound{

**public** HalloweenPumpkin(**double** width,**double** height,Color c){//类的构造函数，给出图像的宽和高

//构造函数部分，初始化HalloweenPumpkin类

face=**new** GOval(width,height-height\*HATSIZE);//脸的位置要下移，高度减去帽子的高度

hat=**new** GRect(width\*HATSIZE,height\*HATSIZE);//大小由之前设置的比例参数决定

//三角形和多边形不能直接声明得到相应的图像，我们建立相应的create方法

lefteye=createEyes(width\*EYESIZE,height\*EYESIZE);//大小由之前设置的比例参数决定

righteye=createEyes(width\*EYESIZE,height\*EYESIZE);//大小由之前设置的比例参数决定

nose=createNose(width\*NOSESIZE,width\*NOSESIZE);//大小由之前设置的比例参数决定

mouse=createMouse(width\*0.5,height\*MOUSESIZE);//大小由之前设置的比例参数决定

//填充各个组件的颜色

hat.setFilled(**true**);

hat.setFillColor(c);

lefteye.setFilled(**true**);

righteye.setFilled(**true**);

mouse.setFilled(**true**);

nose.setFilled(**true**);

face.setFillColor(Color.***orange***);

face.setFilled(**true**);

/\*将各个组件添加到GCompound类中，不是添加到画布

\* 大家可以把GCompound看作是一个小画布，同样使用的是add方法

\* 但与画布中调用的add方法不同，这里add针对的是GCompound

\* add的调用方法，add(GObject,x,y)

\* GObject表示添加的对象，x和y是添加到小画布的位置

\*

\*/

add(face,0,height\*HATSIZE);//脸的初始点x轴最左边，y轴是帽子下边

//左眼的初始点x在宽度的四分之一处减去眼大小的一半，y在高度减去帽子高度的四分之一再加上眼大小的一半加上帽子的高度

add(lefteye,0.25\*width-width\*EYESIZE/2,0.25\*(height-height\*HATSIZE)+height\*EYESIZE/2+height\*HATSIZE);

//右眼的初始点x在宽度的四分之三处减去眼大小的一半，y在高度减去帽子高度的四分之一再加上眼大小的一半加上帽子的高度

add(righteye,0.75\*width-width\*EYESIZE/2,0.25\*(height-height\*HATSIZE)+height\*EYESIZE/2+height\*HATSIZE);

//鼻子在脸部的中央

add(nose,width/2-width\*NOSESIZE/2,height/2-width\*NOSESIZE/2+height\*HATSIZE);

//嘴巴的初始点x在宽的四分之一，高度在脸的四分之三加帽子高度

add(mouse,width\*0.25,(height-height\*HATSIZE)\*0.75+height\*HATSIZE);

//帽子的初始点x在宽度减去帽子宽带的一半，y为0

add(hat,(width-width\*HATSIZE)/2,0);

}

//初始化变量

**private** **double** NOSESIZE=0.15;//占整个图像的比例

**private** **double** EYESIZE=0.15;

**private** **double** MOUSESIZE=0.15;

**private** **double** HATSIZE=0.1;

**private** GOval face;//南瓜脸

**private** GPolygon lefteye,righteye;//眼镜，三角形

**private** GPolygon nose;//鼻子，三角形

**private** GPolygon mouse;//嘴巴，多边形

**private** GRect hat;//帽子，矩形

//创建眼睛，倒三角形，使用GPolygon

**private** GPolygon createEyes(**double** w,**double** h){

//声明一个GPolygon的对象

GPolygon eye=**new** GPolygon();

eye.addVertex(0, 0);//第一个点

eye.addVertex(w, 0);

eye.addVertex(w/2, h);

**return** eye;

};

//创建鼻子，三角形

**private** GPolygon createNose(**double** w,**double** h){

GPolygon nose=**new** GPolygon();

nose.addVertex(w/2, 0);

nose.addVertex(0, h);

nose.addVertex(w, h);

**return** nose;

}

//嘴巴，多边形

**private** GPolygon createMouse(**double** w,**double** h){

GPolygon mouse=**new** GPolygon();

//确定起始点，w和h是嘴的大小

**int** startx=0;

//画上嘴唇，共11个点，循环先画前10个

**for**(**int** i=0;i<5;i++){

mouse.addVertex(startx, h/3);//点之间的高度为嘴巴高的三分之一

startx+=w/10;//画一个点向前移动嘴巴宽度的十分之一

mouse.addVertex(startx, 0);

startx+=w/10;//画一个点向前移动嘴巴宽度的十分之一

}

//加上最后一个点

mouse.addVertex(startx, h/3);

//画下嘴唇，共9个点

startx-=(w/10)\*2;//向后倒退两个宽度

**for**(**int** j=0;j<4;j++){

mouse.addVertex(startx, h);

startx-=w/10;//每画一个点向后倒退两个点的宽度

mouse.addVertex(startx, h/3\*2);

startx-=w/10;

}

//加上最后一个点

mouse.addVertex(startx, h/3);

**return** mouse;

}

}