Chapter 5

**Chap5\_6**

**题目要求：**自定义异常类，参考例子：补充例子源代码\chapter04\自定义异常。**程序代码：**

**package** chap5\_6;

/\*

\* 自定义异常类(继承运行时异常)

\*/

**public** **class** MyException **extends** RuntimeException {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* 错误编码

\*/

**private** String errorCode;

/\*\*

\* 消息是否为属性文件中的Key

\*/

**private** **boolean** propertiesKey = **true**;

/\*\*

\* 构造一个基本异常.

\*

\* **@param** message

\* 信息描述

\*/

**public** MyException(String message)

{

**super**(message);

}

/\*\*

\* 构造一个基本异常.

\*

\* **@param** errorCode

\* 错误编码

\* **@param** message

\* 信息描述

\*/

**public** MyException(String errorCode, String message)

{

**this**(errorCode, message, **true**);

}

/\*\*

\* 构造一个基本异常.

\*

\* **@param** errorCode

\* 错误编码

\* **@param** message

\* 信息描述

\*/

**public** MyException(String errorCode, String message, Throwable cause)

{

**this**(errorCode, message, cause, **true**);

}

/\*\*

\* 构造一个基本异常.

\*

\* **@param** errorCode

\* 错误编码

\* **@param** message

\* 信息描述

\* **@param** propertiesKey

\* 消息是否为属性文件中的Key

\*/

**public** MyException(String errorCode, String message, **boolean** propertiesKey)

{

**super**(message);

**this**.setErrorCode(errorCode);

**this**.setPropertiesKey(propertiesKey);

}

/\*\*

\* 构造一个基本异常.

\*

\* **@param** errorCode

\* 错误编码

\* **@param** message

\* 信息描述

\*/

**public** MyException(String errorCode, String message, Throwable cause, **boolean** propertiesKey)

{

**super**(message, cause);

**this**.setErrorCode(errorCode);

**this**.setPropertiesKey(propertiesKey);

}

/\*\*

\* 构造一个基本异常.

\*

\* **@param** message

\* 信息描述

\* **@param** cause

\* 根异常类（可以存入任何异常）

\*/

**public** MyException(String message, Throwable cause)

{

**super**(message, cause);

}

**public** String getErrorCode()

{

**return** errorCode;

}

**public** **void** setErrorCode(String errorCode)

{

**this**.errorCode = errorCode;

}

**public** **boolean** isPropertiesKey()

{

**return** propertiesKey;

}

**public** **void** setPropertiesKey(**boolean** propertiesKey)

{

**this**.propertiesKey = propertiesKey;

}

}

**程序流程：**

·创建运行时异常类，那么需要继承 RuntimeException 类

·写一个检查性异常类，则需要继承 Exception 类

**程序运行结果**：(由于输出过多，仅展示部分)

Male

Female

Exception in thread "main" chap5\_6.MyException: NA！

at chap5\_6.MyExceptionTest.main(MyExceptionTest.java:11)

**Chap5\_18**

**题目要求：**利用多线程（4个或以上）实现火车订票程序，利用同步使各线程余票正确

**程序代码：**

**package** chap5\_18;

**public** **class** Mainclass {

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

Station station1=**new** Station("Station1");

Station station2=**new** Station("Station2");

Station station3=**new** Station("Station3");

Station station4=**new** Station("Station4");

station1.start();

station2.start();

station3.start();

station4.start();

}

}

**package** chap5\_18;

**public** **class** Station **extends** Thread {

**public** Station(String name) {

**super**(name);

}

**static** **int** *tick* = 20;

**static** Object *ob* = "a";

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

**while** (*tick* > 0) {

**synchronized** (*ob*) {

**if** (*tick* > 0) {

System.***out***.println(getName() + " sells No. " + *tick* + " ticket");

*tick*--;

} **else** {

System.***out***.println("Selling out all tickets");

}

}

**try** {

*sleep*(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**程序流程：**

·创建一个站台类Station，继承Thread，重写run方法

·在run方法里面执行售票操作

·售票要使用同步锁

·创建主方法调用类

**程序运行结果**：

Station2 sells No. 20 ticket

Station4 sells No. 19 ticket

Station3 sells No. 18 ticket

Station1 sells No. 17 ticket

Station4 sells No. 16 ticket

Station3 sells No. 15 ticket

Station2 sells No. 14 ticket

Station1 sells No. 13 ticket

Station2 sells No. 12 ticket

Station3 sells No. 11 ticket

Station4 sells No. 10 ticket

Station1 sells No. 9 ticket

Station3 sells No. 8 ticket

Station4 sells No. 7 ticket

Station2 sells No. 6 ticket

Station1 sells No. 5 ticket

Station2 sells No. 4 ticket

Station4 sells No. 3 ticket

Station3 sells No. 2 ticket

Station1 sells No. 1 ticket

**Chap5\_19**

**题目要求：**仿照课本P122例5.9，将本章作业一个类，列出类名、修饰符名、接口名、方法名。

**程序代码：**

**package** chap5\_19;

**import** java.lang.reflect.\*;

**public** **class** chap\_class {

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

Class<chap\_class> c=chap\_class.**class**;

System.***out***.println(c.getName());

System.***out***.println(c.getSuperclass());

System.***out***.println(Modifier.*toString*(c.getModifiers()));

Type[] ifs=c.getInterfaces();

**if**(ifs.length!=0)

{

**for**(Type inf :ifs)

System.***out***.println(inf.toString());

}**else**{

System.***out***.println("No Interfaces");

}

Member[] mes=c.getMethods();

**if**((mes.length!=0)){

**for**(Member mef:mes){

System.***out***.println(mef.toString());

}

}

**else**{

System.***out***.println("No Member");

}

}

}

**程序流程：**

·列出类名、修饰符名、接口名、方法名。

**程序运行结果**：

chap5\_19.chap\_class

class java.lang.Object

public

No Interfaces

public static void chap5\_19.chap\_class.main(java.lang.String[])

public final void java.lang.Object.wait() throws java.lang.InterruptedException

public final void java.lang.Object.wait(long,int) throws java.lang.InterruptedException

public final native void java.lang.Object.wait(long) throws java.lang.InterruptedException

public boolean java.lang.Object.equals(java.lang.Object)

public java.lang.String java.lang.Object.toString()

public native int java.lang.Object.hashCode()

public final native java.lang.Class java.lang.Object.getClass()

public final native void java.lang.Object.notify()

public final native void java.lang.Object.notifyAll()

**Chapter 6**

**Chap6\_13**

**题目要求：**列出你的Java作业文件夹及合部子文件夹下所有.java文件信息，文件信息包括：路径、文件名、最后修改时间、文件字节数

**程序代码：**

package chap6\_13;

import java.io.File;

import java.io.FilenameFilter;

import java.util.\*;

public class FiLes {

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

System.out.println("Enter location：");

Scanner sc=new Scanner(System.in);

String fileName=sc.nextLine();

File file=new File(fileName);

boolean bl=file.exists();

if(bl){

fileDir(file);

}else

{

System.out.println("DONT EXIST.re enter：");

}}

public static void fileDir(File dir){

File[] files=dir.listFiles();

for(File file:files){

if(file.isDirectory()){

fileDir(file);

}

if(file.getName().endsWith(".java")){

System.out.println("Name "+file.getName());

System.out.println("Location "+file.getAbsolutePath());

System.out.println("Size "+file.length()+"bytes");

System.out.println("Edit time "+file.lastModified());

}

}

}}

**程序流程：**

直接调用main函数

**程序运行结果：**

Enter location：

D:\Gnomeek\eclipse-workspace\JAVA\_LEARN\src\chap6\_13

Name FiLes.java

Location D:\Gnomeek\eclipse-workspace\JAVA\_LEARN\src\chap6\_13\FiLes.java

Size 903bytes

Edit time 1507651900196

**Chap6\_16**

**题目要求：**创建一个带缓冲的输入流，从键盘输入一行行字符，然后把它们写入一个文件，直到用户输入字符串over

**程序代码：**

package chap6\_16;

import java.io.\*;

import java.util.\*;

public class NewFile{

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

System.out.println("Enter new file's name:");

Scanner sc=new Scanner(System.in);

FileWriter writer=new FileWriter("D:\\"+sc.nextLine()+".txt");

BufferedWriter bw=new BufferedWriter(writer);

String line;

while((line=sc.nextLine()).endsWith("over")==false)

{

bw.write(line);

bw.newLine();

}

bw.flush();

bw.close();

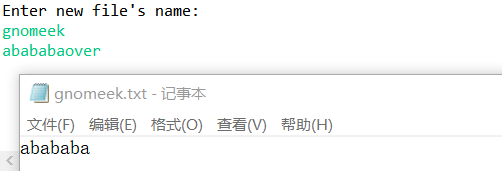
}

}

**程序流程：**

·输入内容导入文件

**程序运行结果：**

****

**Chap6\_17**

**题目要求：**编写一个程序，从一个文本文件众删除所有出现某个指定字符串的地方，文件名和要删除的字符串从命令行输入

**程序代码：**

package chap6\_17;

import java.io.\*;

import java.util.Scanner;

public class EditFile {

private static String message="";

public static void main(String[] args){

try {

editFile();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

private static void editFile()throws Exception{

System.out.println("Input files");

Scanner sc=new Scanner(System.in);

String filename=sc.nextLine();

File file=new File(filename);

if(file!=null&&!(filename.endsWith(".txt")))

{

System.out.println("plz enter right location");

return;

}

System.out.println("enter what to edit,"

+"stop edit to input \"stop\":");

String inputMessage="";

while(!inputMessage.equals("stop")){

inputMessage=sc.nextLine();

if(inputMessage!=null&&inputMessage.length()>0){

String[] editMessage=inputMessage.split(":");

if(editMessage!=null&&editMessage.length>1){

FileReader in=new FileReader(file);

char[] charArray=new char[1024];

int len=0;

StringBuffer sb=new StringBuffer();

while((len=in.read(charArray))!=-1){

sb.append(charArray);

}

message=sb.toString();

in.close();

message=message.replace(editMessage[0],editMessage[1]);

System.out.println("Success");

}

}

}

System.out.println("after edit is :"+message);

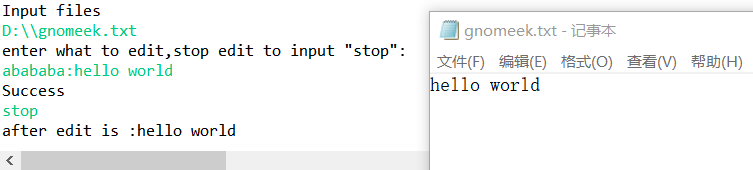
}

}

**程序流程：**

直接运行静态方法main

**程序运行结果：**

****

**Chapter 8**

**Chap8\_5**

**题目要求：**绘制静态图形

**程序代码：**

**package** chap8\_5;

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

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

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

**public** **class** Draw {

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

JFrame window = **new** DrawFrame();

window.setTitle("Draw the picture");

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.setBounds(100,100,600,400);

window.setVisible(**true**);

}

}

**class** DrawFrame **extends** JFrame{

**public** DrawFrame(){

add(**new** DrawComponent());

pack();

}

}

**class** DrawComponent **extends** JComponent{

**private** **static** **final** **int** ***DEAFULT\_WIDTH*** = 400;

**private** **static** **final** **int** ***DEFAULT\_HEIGHT*** = 400;

**public** **void** paintComponent(Graphics g){

Graphics2D g2 = (Graphics2D)g;

**double** leftx = 100;

**double** topy = 100;

**double** width = 200;

**double** height = 150;

Rectangle2D rect = **new** Rectangle2D.Double(leftx,topy ,width,height);

g2.draw(rect);

**double** centerx = rect.getCenterX();

**double** centery = rect.getCenterY();

**double** radius = 150;

Ellipse2D circle = **new** Ellipse2D.Double();

circle.setFrameFromCenter(centerx, centery,centerx+radius,centery+radius);

g2.draw(circle);

}

}

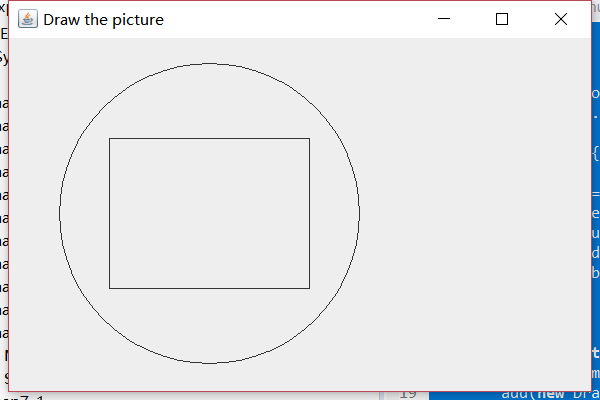
**程序流程：**

·创建窗口类

·创建图形类

·绘制图像

**程序输出结果：**



**Chap8\_12**

**题目要求：**利用双缓存，令动画不闪烁

**程序代码：**

/\*

\* 覆盖update方法

\*/

public void update(Graphics g){

ImageBuffer = createImage(this.getWidth(), this.getHeight());

GraImage = ImageBuffer.getGraphics(); paint(GraImage);

GraImage.dispose();

g.drawImage(ImageBuffer, 0, 0, this);

}

**程序流程：**

·截取上述过程，覆盖update(Graphics)函数

·在内存中创建一个与窗口大小相同的图形，并获得该图形的图形上下文

·再将图片的图形上下文作为参数调用paint(Graphics)函数

·再在update(Graphics)函数调用drawImage函数将创建的图形直接画在窗口上。

**程序运行结果：**

**无**

**Chap8\_14**

**题目要求：**用鼠标绘制图形

**程序代码：**

**package** chap8\_14;

**import** java.awt.BorderLayout;

**import** java.awt.Graphics;

**import** java.awt.event.MouseAdapter;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseMotionAdapter;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

@SuppressWarnings("serial")

**public** **class** DrawPanel **extends** JFrame{

**private** **static** **int** *m*;

**private** **static** **int**[][] *x* = **new** **int**[10][25];

**private** **static** **int**[][] *y* = **new** **int**[10][25];

**private** **static** **int**[] *n* = **new** **int**[25];

**public** DrawPanel(){

**this**.setBounds(100, 200, 300, 300);

**this**.setLayout(**new** BorderLayout());

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

}

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

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

DrawPanel t = **new** DrawPanel();

JPanel jp = **new** JPanel(){

**public** **void** paint(Graphics g){

g.clearRect(0, 0, **this**.getWidth(), **this**.getHeight());

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

**for**(**int** j = 1;j<=*n*[i];j++){

g.drawLine(*x*[i][j], *y*[i][j], *x*[i][j+1], *y*[i][j+1]);

}

}

}

};

jp.addMouseListener(**new** MouseAdapter(){

@Override

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

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

*n*[*m*]++;

*x*[*m*][*n*[*m*]] = e.getX();

*y*[*m*][*n*[*m*]] = e.getY();

*x*[*m*][*n*[*m*]+1] = *x*[*m*][*n*[*m*]];

*y*[*m*][*n*[*m*]+1] = *y*[*m*][*n*[*m*]];

**if**((*x*[*m*][*n*[*m*]]-*x*[*m*][1])\*(*x*[*m*][*n*[*m*]]-*x*[*m*][1])+(*y*[*m*][*n*[*m*]]-*y*[*m*][1])\*(*y*[*m*][*n*[*m*]]-*y*[*m*][1])<=16&&*n*[*m*]>1){

*x*[*m*][*n*[*m*]] = *x*[*m*][1];

*y*[*m*][*n*[*m*]] = *y*[*m*][1];

*m*++;

}

jp.repaint();

}

});

jp.addMouseMotionListener(**new** MouseMotionAdapter(){

@Override

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

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

*x*[*m*][*n*[*m*]+1] = e.getX();

*y*[*m*][*n*[*m*]+1] = e.getY();

jp.repaint();

}

});

t.getContentPane().add(jp, BorderLayout.***CENTER***);

t.setVisible(**true**);

}

}

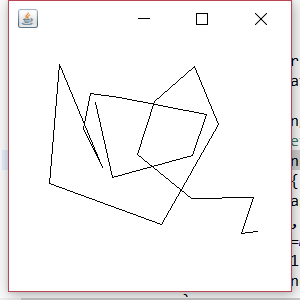
**程序流程：**

·实现左键按下，随着鼠标拖动出现直线

·再次单击左键，直线绘制结束

·不断连线构成图形

**程序运行结果：**



**Chapter 9**

**Chap9\_12**

**题目要求：**利用布局实现登录界面

**程序代码：**

**package chap9\_12;**

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

**import java.awt.\*; //导入必要的包**

**public class Login extends JFrame{**

**JTextField jTextField;**

**JPasswordField jPasswordField;**

**JLabel jLabel1,jLabel2;**

**JPanel jp1,jp2,jp3;**

**JButton jb1,jb2;**

**public Login(){**

**jTextField = new JTextField(12);**

**jPasswordField = new JPasswordField(13);**

**jLabel1 = new JLabel("Username");**

**jLabel2 = new JLabel("Password");**

**jb1 = new JButton("Enter");**

**jb2 = new JButton("Cancel");**

**jp1 = new JPanel();**

**jp2 = new JPanel();**

**jp3 = new JPanel();**

**this.setLayout(new GridLayout(3,1));**

**jp1.add(jLabel1);**

**jp1.add(jTextField);**

**jp2.add(jLabel2);**

**jp2.add(jPasswordField);**

**jp3.add(jb1);**

**jp3.add(jb2);**

**this.add(jp1);**

**this.add(jp2);**

**this.add(jp3);**

**this.setSize(300, 200);**

**this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**this.setVisible(true);**

**this.setTitle("Log In");**

**}**

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

**new Login();**

**}**

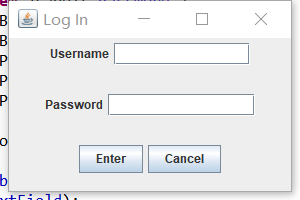
**}**

**程序流程：**

·定义文本框组件

·在面板上添加按钮

**程序输出结果：**



**Chap9\_13**

**题目要求：**用滑块调节字体大小

**程序代码：**

**package chap9\_13;**

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

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

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

**@SuppressWarnings("serial")**

**public class Fontsize extends JFrame implements MouseWheelListener{**

**private JLabel label;**

**private int x=20;**

**public Fontsize(){**

**label=new JLabel("赵",JLabel.CENTER);**

**label.setFont(new Font("微软雅黑",Font.PLAIN,x));**

**this.add(label);**

**this.pack();**

**this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**this.setSize(300, 300);**

**this.setVisible(true);**

**label.addMouseWheelListener(this); }**

**public void mouseWheelMoved(MouseWheelEvent e) {**

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

**x= x+e.getWheelRotation();**

**label.setFont(new Font("微软雅黑",Font.PLAIN,x));**

**this.repaint(); //刷新屏幕**

**}**

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

**new Fontsize();**

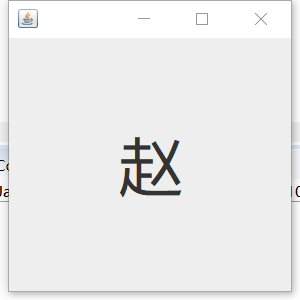
**}**

**程序流程：**

·读入文字

·实现滚轮放缩

**程序运行结果：**



**Chap9\_14**

**题目要求：**JTable 表格行增删

**程序代码：**

**package** chap9\_14;

**import** java.awt.BorderLayout;

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

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseAdapter;

**import** java.awt.event.MouseEvent;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextField;

**import** javax.swing.ListSelectionModel;

**import** javax.swing.table.DefaultTableModel;

**public** **class** JTableDefaultTableModelTest **extends** JFrame{

**private** DefaultTableModel tableModel;

**private** JTable table;

**private** JTextField aTextField;

**private** JTextField bTextField;

**public** JTableDefaultTableModelTest()

{

**super**();

setTitle("Table");

setBounds(100,100,500,400);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

String[] columnNames = {"A","B"};

String [][]tableVales={{"A1","B1"},{"A2","B2"},{"A3","B3"},{"A4","B4"},{"A5","B5"}};

tableModel = **new** DefaultTableModel(tableVales,columnNames);

table = **new** JTable(tableModel);

JScrollPane scrollPane = **new** JScrollPane(table);

getContentPane().add(scrollPane,BorderLayout.***CENTER***);

table.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);

table.addMouseListener(**new** MouseAdapter(){

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

**int** selectedRow = table.getSelectedRow();

Object oa = tableModel.getValueAt(selectedRow, 0);

Object ob = tableModel.getValueAt(selectedRow, 1);

aTextField.setText(oa.toString());

bTextField.setText(ob.toString());

}

});

scrollPane.setViewportView(table);

**final** JPanel panel = **new** JPanel();

getContentPane().add(panel,BorderLayout.***SOUTH***);

panel.add(**new** JLabel("A: "));

aTextField = **new** JTextField("A4",10);

panel.add(aTextField);

panel.add(**new** JLabel("B: "));

bTextField = **new** JTextField("B4",10);

panel.add(bTextField);

**final** JButton addButton = **new** JButton("Add");

addButton.addActionListener(**new** ActionListener(){

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

String []rowValues = {aTextField.getText(),bTextField.getText()};

tableModel.addRow(rowValues);

**int** rowCount = table.getRowCount() +1;

aTextField.setText("A"+rowCount);

bTextField.setText("B"+rowCount);

}

});

panel.add(addButton);

**final** JButton updateButton = **new** JButton("Edit");

updateButton.addActionListener(**new** ActionListener(){

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

**int** selectedRow = table.getSelectedRow();

**if**(selectedRow!= -1)

{

tableModel.setValueAt(aTextField.getText(), selectedRow, 0);

tableModel.setValueAt(bTextField.getText(), selectedRow, 1);

//table.setValueAt(arg0, arg1, arg2)

}

}

});

panel.add(updateButton);

**final** JButton delButton = **new** JButton("Delete");

delButton.addActionListener(**new** ActionListener(){

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

**int** selectedRow = table.getSelectedRow();

**if**(selectedRow!=-1)

{

tableModel.removeRow(selectedRow);

}

}

});

panel.add(delButton);

}

/\*\*

\* **@param** args

\*/

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

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

JTableDefaultTableModelTest jTableDefaultTableModelTest = **new** JTableDefaultTableModelTest();

jTableDefaultTableModelTest.setVisible(**true**);

}

}

**程序流程：**

·创建表格对象

·实现增删功能

**程序运行结果：**

