# 实训1 认识Java

1．阅读程序

（1）① 第1行注释，表示程序名；第2行创建一个公共类Practice1\_1；第3行定义主方法main()；第4行输出“ \* ”；第5行输出“ \*\*\* ”；第6行输出“\*\*\*\*\*”；

② 输出由“\*”组成的三角形。

\*

\*\*\*

\*\*\*\*\*

（2）x + y = 5

x \* y = 6

x / y = 1

（3）您的身高为：1.72

您的体重为：70

您的BMI指数为：23.661438615467823

2．编程训练

（1）

/\* Practice2\_1.java \*/

public class Practice2\_1 {

public static void main(String[] args) {

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

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

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

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

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

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

}

}

（2）

/\* Practice2\_2.java \*/

public class Practice2\_2 {

public static void main(String[] args) {

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

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

System.out.println(" | ：H-a-p-p-y： |");

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

System.out.println("| ：B-i-r-t-h-d-a-y：|");

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

}

}

# 实训2 Java语言基础

1．阅读程序

（1）45.23强制转换成int的结果：45

456.6F强制转换成long的结果：456

97.14强制转换成char的结果：a

（2）x的值为：2.6666666666666665

y的值为：false

b的值为：4 c的值为：4

（3）输入的数为：37

输出的数为：69.0

输出的数为：98.60000000000001

（4）'a'和'b'相加后的结果：195

2．程序填空

① int a = num / 1000;

② int b = (num % 1000) / 100;

③ int c = (num % 100) / 10;

④ int d = num % 10;

3．编程训练

（1）

|  |  |  |  |
| --- | --- | --- | --- |
| 变量描述 | 变量名称 | 数据类型 | 变量初始值 |
| 商品优良等级： | level | char | ‘优’ |
| 商品单价 | price | float | 45.5f |
| 销售总数 | amount | int | 5000 |
| 是否有库 | haveProduct | boolean | true |

/\* Practice3\_1.java \*/

public class Practice3\_1 {

public static void main(String args[]) {

//变量声明及初始化

char level='优';

float price=45.5F;

int amount=5000;

boolean haveProduct=true;

//输出变量的值

System.out.println("商品等级："+level);

System.out.println("商品单价："+price);

System.out.println("目前销售总数："+amount);

System.out.println("现在是否有库存："+haveProduct);

}

}

（2）

/\* Practice3\_2.java \*/

import java.util.Scanner;

public class Practice3\_2 {

public static void main(String args[]){

Scanner scan=new Scanner(System.in); //创建Scanner对象

System.out.println("请输入一个整数：");

int num=scan.nextInt(); //扫描为int型数据

String check=(num%2!=0)?"您输入的整数是奇数":"您输入的整数是偶数";

System.out.println(check); //输出奇偶判断结果

scan.close();

}

}

3．

/\* Practice3\_3.java \*/

public class Practice3\_3 {

public static void main(String[] args) {

int password = 596384; //原密码

int key = 7; //加密参数

System.out.println("原密码：" + password); //输出原密码

password = password << key; //将原密码左移，生成新密码

System.out.println("经过左移运算加密后的结果：" + password);//输出新密码

password = password >> key; //将新密码右移，还原回原来的密码

System.out.println("经过右移运算还原的结果：" + password); //输出原密码

}

}

# 实训3 流程控制

1．阅读程序

（1）程序实现的功能如下：

 

（2）当消费金额未超过200元，无优惠。

当消费金额超过200元，但未超过600元，可享8.5折优惠。

当消费金额超过600元，但未超过1000元，可享7折优惠。

当消费金额超过1000元，可享6折优惠。

2．动手纠错

（1）for (int i = 2; i < num; i++)修改为for (int i = 2; i <= num; i++)

（2）① int k = 50;修改为int k = 1;

② while (k = 0)修改为while (k <= 50)

③ 在while中最后添加语句k++;

（3）① if (a / 9 == 0)修改为if (a % 9 == 0)

② continue;修改为break;

3．编程训练

（1）

/\* Practice3\_1.java \*/

public class Practice3\_1 {

public static void main(String[] args) {

int count = 0; //定义计数变量count，并赋值为0

for (int i = 2; i< 1000; i++) { //循环，当i大于等于1000时结束循环

int b = i / 100; //计算百位数

int s = (i - 100 \* b) / 10; //计算十位数

int g = i % 10; //计算个位数

if (i == g\*g\*g + s\*s\*s + b\*b\*b) { //判断数字的立方和是否等于该数本身

//如果是，计数变量增加1，并输出该数

count++;

System.out.print(i + " ");

}

}

System.out.println(); //换行

System.out.println("水仙花数总共有"+count+"个"); //输出水仙花总个数

}

}

（2）

/\* Practice3\_2.java \*/

import java.util.Scanner;

public class Practice3\_2 {

public static void main(String[] args) {

double weight, price; //定义重量变量和价格变量

Scanner scan = new Scanner(System.in);

System.out.print("请您输入行李的重量：");

weight = scan.nextDouble(); //扫描为double型数据并赋值给n

if (weight < 0) {

System.out.print("您输入的数据有误！");

}

else if (weight <= 20) {

System.out.print("您可以免费托运行李！");

}

else if (weight <= 30) {

price = (weight - 20) \* 5;

System.out.print("您的托运费用为" + price + "元！");

}

else if (weight <= 40) {

price = (30 - 20) \* 5 + (weight - 30) \* 10;

System.out.print("您的托运费用为" + price + "元！");

}

else if(weight <= 50) {

price = (30 - 20) \* 5 + (40 - 30) \* 10 + (weight - 40) \* 15;

System.out.print("您的托运费用为" + price + "元！");

}

else {

System.out.print("您托运的行李超出了最高上限，不允许托运！");

}

scan.close();

}

}

（3）

/\* Practice3\_3.java \*/

public class Practice3\_3 {

public static void main(String[] args) {

int i, j, k;

int n = 0; //定义情况数

//i表示大马数，由于大马驮3担，所以它的取值小于100/3=33

for (i = 1; i < 33; i++)

//j表示中马数，由于大马驮2担，所以它的取值小于100/2=50

for (j = 1; j < 50; j++) {

k = 100 - i - j; //k表示小马数，计算小马数

//判断总担数量是否为100

//由于两匹小马驮一担，所以必须同时满足k为偶数

if ((i \* 3 + j \* 2 + k / 2 == 100) && (k % 2 == 0)) {

n++; //情况数增加1

//输出每种马匹数

System.out.println("大马"+i+"匹,中马"+j+"匹,小马"+k+"匹");

}

}

System.out.print("一共有" + n + "种情况"); //输出情况数

}

}

（4）

/\* Practice3\_4.java \*/

import java.util.Scanner;

public class Practice3\_4 {

static final int FGVALUE = 20; //定义仙女草的售价

static final int GSVALUE = 16; //定义银河梭的售价

static final int FGLIFE = 30; //定义仙女草增加的生命力

static final int GSLIFE = 20; //定义银河梭增加的生命力

public static void main(String[] args) {

int goldcoin = 100; //定义金币的数量

int fg\_num = 0, gs\_num = 0; //定义仙女草与银河梭的数量

int max\_life = 0; //定义最大生命力值

System.out.print("请输入金币的数量为：");

Scanner scan = new Scanner(System.in); //根据Scanner类创建对象

goldcoin = scan.nextInt();//方法nextInt()返回键盘上输入的一个整数

for (int fg\_loop = 0; fg\_loop <= (goldcoin / FGVALUE); fg\_loop++)

for (int gs\_loop = 0; gs\_loop <= (goldcoin / GSVALUE); gs\_loop++)

if (((fg\_loop \* FGVALUE + gs\_loop \* GSVALUE) <= goldcoin)&& ((FGLIFE \* fg\_loop + GSLIFE \* gs\_loop) > max\_life)) {

fg\_num = fg\_loop; //记录仙女草的数量

gs\_num = gs\_loop; //记录银河梭的数量

//记录增加的最大生命力

max\_life = FGLIFE \* fg\_loop + GSLIFE \* gs\_loop;

}

System.out.println("购买的宝物最多能增加你" + max\_life + "个生命力！");

System.out.println("购买仙女草的数量为" + fg\_num);

System.out.println("购买银河梭的数量为" + gs\_num);

scan.close();

}

}

（5）

/\* Practice3\_5.java \*/

public class Practice3\_5 {

public static void main(String[] args) {

int x= 17; //定义牛总数变量x，由于最后生下11头牛

//且国王有6个儿子，故赋初值为17

while (true) //循环

{

//判断牛总数是否同时满足被4、5、6、7、8、9整除

if(x%4==0&&x%5==0&&x%6==0&&x%7==0&&x%8==0&&x%9==0)

//判断最后剩下的牛头数是否等于11

if(x - x / 4 - x / 5 - x / 6 - x / 7 - x / 8 - x / 9 == 11) {

System.out.println(x); //如果是，输出x

break; //退出循环

}

x++; //x增加1

}

}

}

# 实训4 Java面向对象编程基础

1．阅读程序

（1）Circle1:x=400,y=100,r=50

Circle2:x=400,y=100,r=150

2．程序填空

① RectConstructor rect1 = new RectConstructor(10, 20);

② return length \* width;

③ this.length = length;

④ this.width = width;

3．动手纠错

GasStation gs = null;修改为GasStation gs = new GasStation();

public class GasStation修改为class GasStation

4．编程训练

（1）

/\* EncryptTest.java \*/

package chapter4.practice4\_1;

import java.util.Scanner;

class Encrypt {

int code;

Encrypt(int code) {

this.code = code;

}

int encryptCode() {

int qw = code / 1000; // 取千位数

int bw = (code - qw \*1000) / 100; // 取百位数

int sw = (code - qw \*1000 - bw \* 100) / 10; // 取十位数

int gw = code % 10; // 取个位数

qw = (qw + 5) % 10; // 千位数加5除以10

bw = (bw + 5) % 10; // 百位数加5除以10

sw = (sw + 5) % 10; // 十位数加5除以10

gw = (gw + 5) % 10; // 个位数加5除以10

int temp; // 定义中间量

// 第一位和第四位交换

temp = qw;

qw = gw;

gw = temp;

// 第二位和第三位交换

temp = bw;

bw = sw;

sw = temp;

// 返回加密后的四位整数

return qw \* 1000 + bw \* 100 + sw \* 10 + gw;

}

}

public class EncryptTest {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.print("请输入一个四位整数：");

int code = scan.nextInt(); // 扫描为int型数据并赋值给code

Encrypt c = new Encrypt(code); // 创建Encrypt类的对象

System.out.println("加密后的四位整数为：" + c.encryptCode());

scan.close(); // 关闭扫描器

}

}

（2）

/\* Student.java \*/

package chapter4.practice4\_2;

public class Student {

String sNo;

String sName;

char sSex;

int sAge;

double sJava;

// 定义有参构造方法，初始化成员变量

Student(String sNo, String sName, char sSex, int sAge, double sJava) {

this.sNo = sNo;

this.sName = sName;

this.sSex = sSex;

this.sAge = sAge;

this.sJava = sJava;

}

String getNo( ) { // 获取学号

return sNo;

}

String getName( ) { // 获取姓名

return sName;

}

char getSex( ) { // 获取性别

return sSex;

}

int getAge( ) { // 获取年龄

return sAge;

}

double getJava( ) { // 获取Java课程成绩

return sJava;

}

// 显示学生信息

public void display() {

System.out.println("学号：" + sNo + " " + "姓名：" + sName + " " + "性别：" + sSex + " " + "年龄：" + sAge + " " + "Java课程成绩：" + sJava);

}

public static void main(String[] args) {

// 创建对象并初始化对象

Student s1 = new Student("001", "张一", '男', 18, 80);

Student s2 = new Student("002", "李二", '女', 19, 75.5);

Student s3 = new Student("003", "刘三", '女', 20, 90);

Student s4 = new Student("004", "赵四", '男', 18, 88);

Student s5 = new Student("005", "钱五", '男', 20, 70);

s1.display(); // 调用display方法

s2.display();

s3.display();

s4.display();

s5.display();

}

}

（3）

/\* Student.java \*/

package chapter4.practice4\_2;

// 导入chapter4.practice4\_2.math.ComputeMaxMin类

import chapter4.practice4\_2.math.ComputeMaxMin;

public class Student {

String sNo;

String sName;

char sSex;

int sAge;

double sJava;

// 定义有参构造方法，初始化成员变量

Student(String sNo, String sName, char sSex, int sAge, double sJava) {

this.sNo = sNo;

this.sName = sName;

this.sSex = sSex;

this.sAge = sAge;

this.sJava = sJava;

}

String getNo( ) { // 获取学号

return sNo;

}

String getName( ) { // 获取姓名

return sName;

}

char getSex( ) { // 获取性别

return sSex;

}

int getAge( ) { // 获取年龄

return sAge;

}

double getJava( ) { // 获取Java课程成绩

return sJava;

}

// 显示学生信息

public void display() {

System.out.println("学号：" + sNo + " " + "姓名：" + sName + " " + "性别：" + sSex + " " + "年龄：" + sAge + " " + "Java课程成绩：" + sJava);

}

public static void main(String[] args) {

// 创建Student对象并初始化对象

Student s1 = new Student("001", "张一", '男', 18, 80);

Student s2 = new Student("002", "李二", '女', 19, 75.5);

Student s3 = new Student("003", "刘三", '女', 20, 90);

Student s4 = new Student("004", "赵四", '男', 18, 88);

Student s5 = new Student("005", "钱五", '男', 20, 70);

s1.display(); // 调用display方法

s2.display();

s3.display();

s4.display();

s5.display();

// 创建ComputeMaxMin类对象并初始化对象

ComputeMaxMin c= new ComputeMaxMin(s1.sJava, s2.sJava, s3.sJava, s4.sJava, s5.sJava);

System.out.println("Java课程成绩最大值：" + c.max());// 调用max方法并输出Java课程成绩最大值

System.out.println("Java课程成绩最小值：" + c.min());// 调用min方法并输出Java课程成绩最小值

}

}

/\*ComputeMaxMin.java\*/

package chapter4.practice4\_2.math;

public class ComputeMaxMin {

double num1, num2, num3, num4, num5;

// 定义有参构造方法，并初始化成员变量

public ComputeMaxMin(double num1, double num2, double num3, double num4, double num5) {

this.num1 = num1;

this.num2 = num2;

this.num3 = num3;

this.num4 = num4;

this.num5 = num5;

}

public double max() { // 定义方法计算最大值

double max;

max = Math.max(num1, num2);// 调用Math类中max方法计算两个数的较大值

max = Math.max(max, num3);

max = Math.max(max, num4);

max = Math.max(max, num5);

return max; // 返回最大值

}

public double min() { // 定义方法计算最小值

double min;

min = Math.min(num1, num2);// 调用Math类中的min方法计算两个数的较小值

min = Math.min(min, num3);

min = Math.min(min, num4);

min = Math.min(min, num5);

return min; // 返回最小值

}

}

（4）

/\* ComputeRootTest.java\*/

package chapter4.practice4\_4;

public class ComputeRootTest {

public static void main(String[] args) {

ComputeRoot y1, y2; // 声明对象

y1 = new ComputeRoot(1, 5, 10); // 使用有参构造方法创建对象

y1.showRoot(); // 调用showRoot方法

y2 = new ComputeRoot(1, 10, 9);

y2.showRoot();

}

}

class ComputeRoot {

double x1, x2;

private double a, b, c;

// 带参数构造方法初始化成员变量

public ComputeRoot(double a, double b, double c) {

this.a = a;

this.b = b;

this.c = c;

}

// 计算方程实根的方法

void real\_root(double delt) {

x1 = (-b + Math.sqrt(delt)) / (2 \* a);

x2 = (-b - Math.sqrt(delt)) / (2 \* a);

System.out.println("delt>=0，方程的实根为\nx1=" + x1); // 输出方程的实根

System.out.println("x2=" + x2);

}

// 计算方程虚根的方法

void imag\_root(double delt) {

double real, imag;

real = (-b) / (2 \* a);

imag = Math.sqrt(-delt) / (2 \* a);

System.out.println("delt<0，方程的虚根为\nx1=" + real + "+" + imag + "i");

System.out.println("x2=" + real + "-" + imag + "i");

}

// 根据判别式delt的值，选择调用的方法

void showRoot() {

double delt = b \* b - 4 \* a \* c;

if (delt >= 0)

real\_root(delt);

else

imag\_root(delt);

}

}

# 实训5 Java面向对象高级特性

1．阅读程序

同学：老师好

老师：同学们好

老师：开始上课

同学：开始记笔记

2．动手纠错

public abstract void shop() {}修改为public abstract void shop();

JDMarket和WuMeiMarket类中的void shop()修改为public void shop()

3．编程训练

（1）

/\* Fruit.java \*/

package chapter5.practice3\_1;

public class Fruit {

private String name; // 定义水果名name

private String area; // 定义水果产地area

private float price; // 定义水果单价price

void setName(String name) { //给name赋值

this.name = name;

}

String getName() { //获取name

return name;

}

void setArea(String area) { //给area赋值

this.area = area;

}

String getArea() { //获取area

return area;

}

void setPrice(float price) { //给price赋值

this.price = price;

}

float getPrice() { //获取price

return price;

}

}

/\* FruitTest.java \*/

package chapter5.practice3\_1;

public class FruitTest {

public static void main(String[] args) {

Fruit f1 = new Fruit(); // 创建水果类对象f1

f1.setName("苹果"); // 调用setName()方法，给name赋值

f1.setArea("山东烟台"); // 调用setArea()方法，给area赋值

f1.setPrice(5.99f); // 调用setPrice()方法，给price赋值

// 输出f1的name、area和price

System.out.println(f1.getName() + " " + f1.getArea() + " " + f1.getPrice() + "元/斤");

Fruit f2 = new Fruit(); // 创建水果类对象f2

f2.setName("爱媛果冻橙"); // 调用setName()方法，给name赋值

f2.setArea("四川眉山"); // 调用setArea()方法，给area赋值

f2.setPrice(7.99f); // 调用setPrice()方法，给price赋值

// 输出f2的name、area和price

System.out.println(f2.getName() + " " + f2.getArea() + " " + f2.getPrice() + "元/斤");

Fruit f3 = new Fruit(); // 创建水果类对象f3

f3.setName("车厘子"); // 调用setName()方法，给name赋值

f3.setArea("智利进口"); // 调用setArea()方法，给area赋值

f3.setPrice(39.99f); // 调用setPrice()方法，给price赋值

// 输出f3的name、area和price

System.out.println(f3.getName() + " " + f3.getArea() + " " + f3.getPrice() + "元/斤");

}

}

（2）

/\* SonLikeDo.java \*/

package chapter5.practice3\_2;

interface DadLikeDo { // 定义“爸爸喜欢做的事”接口

void ride(); // 骑车的方法

void basketball(); // 打篮球的方法

}

interface MomLikeDo { // 定义“妈妈喜欢做的事”接口

void sing(); // 唱歌的方法

void dance(); // 跳舞的方法

}

// 继承DadLikeDo接口和MomLikeDo接口

public class SonLikeDo implements DadLikeDo, MomLikeDo {

public void ride() { // 实现ride()方法

System.out.println("骑车");

}

public void basketball() { // 实现basketball()方法

System.out.println("打篮球");

}

public void sing() { // 实现sing()方法

System.out.println("唱歌");

}

public void dance() { // 实现dance()方法

System.out.println("跳舞");

}

public static void main(String[] args) {

SonLikeDo son = new SonLikeDo(); // 通过子类创建接口对象

System.out.println("儿子喜欢做的事有：");

// 子类对象调用DadLikeDo和MomLikeDo两个接口被实现的所有方法

son.ride();

son.basketball();

son.sing();

son.dance();

}

}

（3）

/\* Factory.java \*/

package chapter5.practice3\_3;

public abstract class Factory {

String name; // 定义工厂名name

public String getName() { // 设置工厂名

return name;

}

public void setName(String name) { // 获取工厂名

this.name = name;

}

public abstract String produce(); // 定义抽象方法，返回产品名

}

class PhonePlant extends Factory { // 定义PhonePlant类，继承Factory类

String productsName; // 定义产品名productsName

public PhonePlant(String productsName) { // 定义有参构造方法，传递产品名

this.productsName = productsName;

}

@Override

public String produce() { // 重写produce()方法

return this.productsName;

}

}

class ClothsFactory extends Factory { // 定义ClothsFactory类，继承Factory类

String productsName; // 定义产品名productsName

public ClothsFactory(String productsName) {// 定义有参构造方法，传递产品名

this.productsName = productsName;

}

@Override

public String produce() { // 重写produce()方法

return this.productsName;

}

}

/\* FactoryTest \*/

package chapter5.practice3\_3;

public class FactoryTest {

public static void main(String[] args) {

// 定义PhonePlant类对象

PhonePlant phonePlant = new PhonePlant("手机");

phonePlant.setName("手机厂"); // 设置工厂名

// 输出产品

System.out.println(phonePlant.getName() + "生产的是"

+ phonePlant.productsName);

// 定义ClothsFactory类对象

ClothsFactory clothsFactory = new ClothsFactory("衣服");

clothsFactory.setName("服饰厂"); // 设置工厂名

// 输出产品

System.out.println(clothsFactory.getName() + "生产的是"

+ clothsFactory.productsName);

}

}

# 实训6 数组和字符串

1．阅读程序

64

2．程序填空

① int[][] result = new int [2][3];

② i < 2

③ j < 3

3．动手纠错

（1）String[] product= new String修改为String[] product= new String[]

for (int i = 0; i < product.length(); i++)修改为for (int i = 0; i < product.length; i++)

（2）两处的for(int i = 0; c[i] != '\0'; i++)都修改为for(int i = 0; i < c.length; i++)

if((c[i] >= 'A' && c[i] <= 'Z') && (c[i] >= 'a' && c[i] <= 'z'))修改为

if((c[i] >= 'A' && c[i] <= 'Z') || (c[i] >= 'a' && c[i] <= 'z'))

4．编程训练

（1）

/\* DuotoneBall.java \*/

package chapter6.practice4\_1;

public class DuotoneBall {

public static void main(String[] args) {

int[] nums = new int[7]; // 初始化一个int类型的数组

for (int j = 0; j < 6; j++) { // 产生6个红球

// 给数组中的前6个元素赋值（1-33之间的随机数）

nums[j] = (int) (Math.random() \* 33) + 1;

// 比较每个数组中的元素是否相等；如果相等，就要跳出，重复生成

for (int k = 0; k < j; k++) {

if (nums[j] == nums[k] && j != 0) {// 判断“元素是否相等”的条件

j--;

break;

}

}

}

// 产生1个蓝球，给数组中的后1两个元素赋值（1-16之间的随机数）

nums[6] = (int) (Math.random() \* 16) + 1;

System.out.println("机选双色球的号码："); // 提示信息

for (int j = 0; j < nums.length; j++) { // 遍历数组

System.out.print(nums[j] + " "); // 显示数组中的每个元素

}

}

}

（2）

/\* NumberManager.java \*/

package chapter6.practice4\_2;

public class NumberManager {

/\* 统计每个数字（0，1，…，8，9）出现的频率 \*/

public static int[] countNumbers(String numbers) {

int[] numberArray = new int[10];

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

/\*numbers[i].charAt(index)方法返回索引位置处的字符，

它与字符'0'的差值为对应数字符的数值。

例如，'1'-'0'=1，'5'-'0'=5等。因此，如果字符为'5'，

则相当于numberArray[5]加1，其他字符与此类似。

\*/

numberArray[numbers.charAt(i) - '0']++;

}

return numberArray;

}

/\*打印整型数组\*/

private static void printArray(int[] numArr) {

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

System.out.printf(i + ":" + numArr[i] + ",");

}

System.out.println();

}

/\*替换数字\*/

public static String replaceNumbers(String numbers, int[] numberCounts) {

String results;

int replaceNum = getMaxNumber(numberCounts); // 得到最大数的索引下标

char c = (char)(replaceNum + '0'); // 将索引转化为字符

results = numbers.replace(c, '8');

return results;

}

/\*得到出现频率最高数字的下标\*/

private static int getMaxNumber(int[] numberCounts) {

int currenMaxNumber = -1;

int index = -1;

//获取numberCounts数组中的最大数

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

if (numberCounts[i] > currenMaxNumber) { // 记录当前最大数

currenMaxNumber = numberCounts[i]; // 记录最大数的下标

index = i;

}

}

return index;

}

public static void main(String[] args) {

String numbers = "13701192543";

System.out.println("原始电话号码：" + numbers);

// 统计各数字串中数字字符'0'、'1'...'9'出现的频率

int[] numArr = countNumbers(numbers);

System.out.println("各数字在电话号码中出现的频率：");

// 打印数字字符'0'、'1'...'9'出现的频率

printArray(numArr);

// 将数字串中高频数字符与字符'8'互换

String newArr = replaceNumbers(numbers, numArr);

System.out.println("高频数字与8互换后的电话号码: " + newArr);

}

}

（3）

/\* ChangeAge.java \*/

package chapter6.practice4\_3;

import java.util.Scanner; //导入Scanner类

public class ChangeAge {

public static void main(String[] args) {

// 创建扫描器对象，实现控制台输入

Scanner scan = new Scanner(System.in);

// 提示信息

System.out.print("用户名：");

// 接收控制台输入的用户名

String userName = scan.next();

int userAge = 25;

System.out.println("年龄：" + userAge);

boolean flag = true;

while (flag) {

System.out.println("======选项条件======");

// 控制台输出选项信息

System.out.println("1.确认 2.更改");

// 提示信息

System.out.print("请选择：");

// 接收控制台输入的数字

int choiceNum = scan.nextInt();

// 创建一个有参的、字符序列可变的字符串对象

StringBuffer sbf = new StringBuffer("用户" + userName + "的年龄是"

+ userAge);

// 以控制台输入的数字为参数的多分支语句

switch (choiceNum) {

case 1:// 控制台输入1时

// 提示信息

System.out.println("======确认信息======");

// 控制台输出字符串对象

System.out.println(sbf);

flag = false;

break;

case 2:// 控制台输入2时

// 提示信息

System.out.println("======更改信息======");

// 提示信息

System.out.print("年龄：");

// 接收控制台输入的年龄

userAge = scan.nextInt();

break;

}

}

// 关闭控制台输入

scan.close();

}

}

# 实训7 泛型和集合

1．阅读程序

size = 4

10 15 18 20

2．程序填空

① shelf.add(0, desk.get(1));

② shelf.set(2, desk.get(0));

③ i < shelf.size()

3．编程训练

（1）

/\* Joseph.java \*/

package chapter7.practice3\_1;

import java.util.LinkedList;

public class Joseph {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>(); // 创建int型List对象list

for(int i=0;i<500;i++) { // 循环500次

list.add(i); // 使用add()方法将i添加到list中

}

int index=0; // 定义index，并赋为0

int count=0; // 定义count，并赋为0

while(list.size()>1) { // 循环

count++; // count加1

if(count%3==0) { // 如果count能被3整除

list.remove(index); // 删除list中索引为index的元素

index--; // index减1

}

index ++; // index加1

if(index ==list.size()) { // 如果index等于list的长度

index =0; // index赋为0

}

}

System.out.println(list); // 输出list

}

}

（2）

/\* StudentManage.java \*/

package chapter7.practice3\_2;

import java.util.\*;

class Student {

String name; // 定义姓名name

char sex; // 定义性别sex

int age; // 定义年龄age

// 定义有参构造方法，初始化stuName、stuSex和stuAge

public Student(String name, char sex, int age) {

this.name = name;

this.sex = sex;

this. age = age;

}

}

public class StudentManage {

public static void main(String[] args) {

// 创建类型为Student类的ArrayList对象stuList1

List<Student> stuList1 = new ArrayList<Student>();

// 向stuList1中添加3个Student对象

stuList1.add(new Student("张三丰", '男', 7));

stuList1.add(new Student("杨过", '男', 9));

stuList1.add(new Student("郭靖", '女', 8));

// 遍历stuList1

for (int i = 0; i < stuList1.size(); i++) {

Student s = stuList1.get(i);

// 调用get()方法获取stuList1中的元素并输出

System.out.println(s.name + "\t" + s.sex + "\t" + s.age + "\t");

}

}

}

（3）

/\* StudentManage.java \*/

package chapter7.practice3\_3;

import java.util.\*;

class Student {

String name; // 定义姓名name

char sex; // 定义性别sex

int age; // 定义年龄age

// 定义有参构造方法，初始化stuName、stuSex和stuAge

public Student(String name, char sex, int age) {

this.name = name;

this.sex = sex;

this. age = age;

}

}

public class StudentManage {

public static void main(String[] args) {

// 创建类型为Student类的ArrayList对象stuList1、stuList2和stuList3

List<Student> stuList1 = new ArrayList<Student>();

List<Student> stuList2 = new ArrayList<Student>();

List<Student> stuList3 = new ArrayList<Student>();

// 创建键为String型，值为List类型的Map对象stuGrade

Map<String, List<Student>> stuGrade = new HashMap<String, List<Student>>();

// 向stuList1中添加3个Student对象

stuList1.add(new Student("张三丰", '男', 7));

stuList1.add(new Student("杨过", '男', 9));

stuList1.add(new Student("郭靖", '女', 8));

// 向stuList2中添加3个Student对象

stuList2.add(new Student("刘备", '男', 10));

stuList2.add(new Student("张飞", '男', 8));

stuList2.add(new Student("诸葛亮", '男', 9));

// 向stuList3中添加3个Student对象

stuList3.add(new Student("许嵩", '男', 11));

stuList3.add(new Student("刘能", '男', 12));

stuList3.add(new Student("马三", '男', 10));

// 向stuGrade中添加3个键-值对

stuGrade.put("三年级一班", stuList1);

stuGrade.put("三年级二班", stuList2);

stuGrade.put("四年级一班", stuList3);

Scanner scan = new Scanner(System.in);

System.out.print("请输入班级名称：");

String inputClass = scan.nextLine();

System.out.println(inputClass + "的学生列表：");

// 使用get方法获取输入键对应的值，并赋给stulist

List<Student> stulist = stuGrade.get(inputClass);

// 遍历stulist

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

Student s = stulist.get(i);

// 调用get()方法获取stulist中的元素并输出

System.out.println(s.name + "\t" + s.sex + "\t" + s.age + "\t");

}

scan.close(); // 关闭扫描器

}

}

# 实训8 异常和断言

1．程序填空

① double result = leftMoney - drawMoney;

② System.out.println("您账户的余额不足！");

③ System.out.println("您输入的取款金额不是整数！");

2．动手纠错

public static void pay(double weight)修改为public static void pay(double weight) throws OverloadException

float money = weight \* 3.98;修改为float money = (float) (weight \* 3.98);

double weight = scan.next();修改为double weight = scan.nextDouble();

3．编程训练

（1）

/\* FingerGuess.java \*/

package chapter8.practice3\_1;

import java.util.Scanner;

public class FingerGuess {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

while (true)

try {

System.out.print("请用户1输入：0(剪刀) 1(石头) 2(布)：");

int player1 = scan.nextInt();// 获取用户1输入的信息并赋值

System.out.print("请用户2输入：0(剪刀) 1(石头) 2(布)：");

int player2 = scan.nextInt();// 获取用户2输入的信息并赋值

// 断言play1和play2必须是0,1或2，如果不是抛出异常

assert (player1 >= 0 && player1 < 3) && (player2 >= 0 && player2 < 3)

: ("play1和play2必须是0,1或2");

// 用户1所有能获胜的判断条件

if (((player1 == 0) && (player2 == 2)) ||

((player1 == 1) && (player2 == 0)) ||

((player1 == 2) && (player2 == 1)))

System.out.println("用户1获得胜利");// 输出“用户1获得胜利”

else if (player1 == player2) // 用户1输入与用户2相同时

System.out.println("平局，再来一局");// 输出“平局，再来一局”

else // 用户2获胜

System.out.println("用户2获得胜利");// 输出“用户2获得胜利”

break; //跳出循环

} catch(AssertionError e) {

System.out.println("异常信息：" + e.getMessage());

}

scan.close();

}

}

（2）

/\* InputChar.java \*/

package chapter8.practice3\_2;

import java.util.Scanner;

public class InputChar {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in); // 创建Scanner对象scan

while (true) { // 循环

try {

System.out.print("请输入一个小写字符：");

String s = scan.nextLine();

if (s.length() == 1) { // 如果字符串的长度等于1

char c = s.charAt(0); // 获取s的第1个字符，并赋给c

if ('a'<= c && c<='z') { // 如果c是小写字母

System.out.println(c); // 输出c

break; // 退出循环

}

else

// 如果c不是小写字母，抛出异常，并设置异常信息

throw new Exception("输入的字符应为小写字符");

}

else

// 如果字符串的长度不等于1，抛出异常，并设置异常信息

throw new Exception("只需输入一个字符");

} catch(Exception e) { // 捕获异常

System.out.println(e.getMessage()); // 输出异常信息

}

}

scan.close(); // 关闭扫描器

}

}

（3）

/\* Weight.java \*/

package chapter8.practice3\_3;

import java.util.\*;

class HeightException extends Exception {}

public class Weight {

public static void main(String[] args) {

System.out.println("请输入身高："); // 输出提示

Scanner scan = new Scanner(System.in);

try {

float height = scan.nextFloat();

if (height < 0.3 || height > 2.5) // 如果height小于30或大于250

throw new HeightException(); // 抛出HeightException异常

System.out.println("请输入体重："); // 输出提示

float weight = scan.nextFloat();

double BMI = weight/Math.pow(height, 2); // 计算BMI指数

// 如果BMI小于18.5，输出“消瘦”

if (BMI < 18.5)

System.out.println("消瘦");

// 如果BMI小于24且大于等于18.5，输出“正常”

else if (BMI < 24)

System.out.println("正常");

// 如果BMI小于28且大于等于24，输出“超重”

else if (BMI < 28)

System.out.println("正常");

else // 如果BMI大于等于28，输出“肥胖”

System.out.println("肥胖");

} catch(InputMismatchException e) { // 捕获输入数据类型异常

System.out.println("输入异常，请输入浮点数或整数！");

} catch(HeightException e) { // 捕获输入数值异常

System.out.println("您输入的身高有误！");

} finally {

scan.close(); // 关闭扫描器

}

}

}

# 实训9 I/O流和文件操作

1．程序填空

① fw = new FileWriter(fileName, true);

② i < content.length

③ fw.write(content[i] + "\n");

2．动手纠错

FileOutputStream fos = new FileOutputStream(f);修改为FileOutputStream fos = new FileOutputStream(f, true);

fos.write(str);修改为fos.write(str.getBytes());

3．编程训练

（1）

/\* ExtractFile.java \*/

package chapter9.practice3\_1;

import java.io.\*;

public class ExtractFile {

public static void main(String[] args) {

InputStreamReader filein = null;

BufferedReader bufin = null;

FileWriter fileout = null;

BufferedWriter bufout = null;

String originalText = "";

File file = new File("园丁集.txt"); // 创建文件对象file

try {

// 使用InputStreamReader创建输入流，设置编码为UTF-8

filein = new InputStreamReader(new FileInputStream(file), "UTF-8");

bufin = new BufferedReader(filein); // 创建缓冲输入流

String s = null;

// 读取文件，并判断读取的字符串不为空

while ((s = bufin.readLine()) != null) {

// 判断读取的一行是否为原文，如果是则连接到originalText

if (s.startsWith("【原文】"))

originalText += s + "\n";

}

bufin.close();

filein.close();

} catch (IOException e) {

System.out.println(e);

}

file = new File("园丁\_原文.txt"); // 创建文件对象file

try {

fileout = new FileWriter(file); // 创建文件字符输入流

bufout = new BufferedWriter(fileout); // 创建缓冲字符流

bufout.write(originalText); // 将originalText写入文件

bufout.close();

fileout.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

（2）

/\* StudentSystem.java \*/

package chapter9.practice3\_2;

import java.io.\*;

import java.util.\*;

public class StudentSystem {

static Scanner scan = new Scanner(System.in);

static ArrayList<HashMap<String, String>> stuInfos = new ArrayList<>();

// 输出功能信息

public static void printMenu() {

System.out.println("\*\*\*\*\*\*\*\*学生管理系统\*\*\*\*\*\*\*\*");

System.out.println("1.添加学生信息");

System.out.println("2.删除学生信息");

System.out.println("3.显示所有学生信息");

System.out.println("4.保存数据");

System.out.println("5.恢复数据");

System.out.println("0.退出系统");

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

}

// 从键盘输入新学生的学号、姓名和性别

// 将每个学生的信息添加到Map中，然后将Map添加到List中

public static void addStuInfo() {

HashMap<String, String> newInfo = new HashMap<>();

System.out.print("请输入新学生的学号：");

newInfo.put("num", scan.next());

System.out.print("请输入新学生的姓名：");

newInfo.put("name", scan.next());

System.out.print("请输入新学生的性别（男/女）：");

newInfo.put("sex", scan.next());

stuInfos.add(newInfo);

}

// 根据输入的学号获取在列表中的索引，然后删除此学生信息

public static void delStuInfo() {

if (stuInfos.size() > 0) {

System.out.print("请输入要删除的学生的学号：");

String num = scan.next();

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

if (stuInfos.get(i).containsValue(num)) {

stuInfos.remove(i);

break;

}

}

} else

System.out.println("没有该学生信息！");

}

// 显示所有学生的信息

public static void showStuInfo() {

System.out.println("序号\t学号\t姓名\t性别");

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

System.out.println((i+1) + "\t" + stuInfos.get(i).get("num") + "\t"

+ stuInfos.get(i).get("name") + "\t" + stuInfos.get(i).get("sex"));

}

}

// 将学生信息保存到文件中

public static void saveStuInfo() {

try {

// 使用OutputStreamWriter创建文件输出流，并设置编码为UTF-8

OutputStreamWriter fw;

fw = new OutputStreamWriter(new FileOutputStream("StudentInfo.txt"), "UTF-8");

//fw = new FileWriter("StudentInfo.txt", true);

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

fw.write((i+1) + "\t" + stuInfos.get(i).get("num") + "\t"

+ stuInfos.get(i).get("name") + "\t"

+ stuInfos.get(i).get("sex") + "\n");

}

fw.close();

} catch(IOException e) {

e.printStackTrace();

}

}

// 从文件中读取学生信息

// 将每个学生的信息存储到Map中，然后将Map存储到List中

public static void recoverStuInfo() {

try {

// 使用InputStreamReader创建文件输入流，并设置编码为UTF-8

InputStreamReader fr;

fr = new InputStreamReader(new FileInputStream("StudentInfo.txt"), "UTF-8");

char[] cBuff = new char[1024];

int len = fr.read(cBuff); // 读取数据，并保存在字符数组cBuff中

String[] sBuff = new String(cBuff, 0, len).split("\n");

String[] s;

System.out.println("序号\t学号\t姓名\t性别");

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

HashMap<String, String> newInfo = new HashMap<>();

System.out.println(sBuff[i]);

s = sBuff[i].split("\t");

newInfo.put("num", s[1]);

newInfo.put("name", s[2]);

newInfo.put("sex", s[3]);

stuInfos.add(newInfo);

}

fr.close();

} catch(IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

// 循环，根据输入的数字，调用不同的方法实现不同的功能

while (true) {

printMenu();

System.out.print("请输入功能对应的数字：");

String key = scan.next();

if (key.equals("0"))

break;

else if (key.equals("1"))

addStuInfo();

else if (key.equals("2"))

delStuInfo();

else if (key.equals("3"))

showStuInfo();

else if (key.equals("4"))

saveStuInfo();

else if (key.equals("5"))

recoverStuInfo();

else

System.out.println("Error!");

}

}

}

（3）

/\* FolderTest.java \*/

package chapter9.practice3\_3;

import java.io.File;

public class FolderTest {

public static void main(String[] args) {

String path = "D:/Test"; // 声明文件夹Test所在的目录

for (int i = 1; i <= 10; i++) { // 循环获得i值，并用i命名新的文件夹

File folder = new File(path + "/" + i); // 根据新的目录创建File对象

if (!folder.exists()) { // 文件夹不存在

folder.mkdirs(); // 创建新的文件夹(包括不存在的父文件夹)

}

}

System.out.println("文件夹创建成功，请打开D盘查看！"

+ "\n\nD盘文件及文件夹列表如下：");

File file = new File("D:/"); // 根据路径名创建File对象

File[] files = file.listFiles(); // 获得D盘的所有文件和文件夹

for (File folder : files) { // 遍历files数组

if (folder.isFile()) // 判断是否为文件

// 输出D盘下所有文件的名称

System.out.println(folder.getName() + " 文件");

else if (folder.isDirectory()) // 判断是否为文件夹

// 输出D盘下所有文件夹的名称

System.out.println(folder.getName() + " 文件夹");

}

}

}

# 实训10 多线程

1．程序填空

① sleep(500);

② NumOutput t = new NumOutput();

③ t.start();

2．动手纠错

public void subtraction()修改为public synchronized void subtraction()

System.out.println(Thread.currentThread().getName() + "说：num的值为" + num);修改为System.out.println(Thread.currentThread().getName() + "说：num的值为" + num--);

3．编程训练

（1）

/\* Race.java \*/

package chapter10.practice3\_1;

class Rabbit implements Runnable { // 创建兔子跑步线程类Rabbit

@Override

public void run() {

// 创建乌龟跑步线程对象

Tortoise runTortoise = new Tortoise();

Thread tortoise = new Thread(runTortoise);

tortoise.start(); // 启动乌龟跑步线程

for (int i = 1; i <= 10; i++) { // 循环10次模拟赛跑的过程

try {

Thread.sleep(100); // 线程休眠0.1秒，模拟兔子在跑步

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("兔子跑了" + i \* 10 + "米"); // 显示兔子的跑步距离

if (i == 8) {

System.out.println("兔子在睡觉"); // 当跑了80米时开始睡觉

try {

tortoise.join(); // Tortoise线程加入

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

System.out.println( "兔子到达终点"); // 显示兔子到达了终点

}

}

class Tortoise implements Runnable { // 创建乌龟跑步线程类Tortoise

@Override

public void run() {

for (int i = 1; i <= 10; i++) {

try {

Thread.sleep(500); // 线程休眠0.5秒，模拟乌龟在跑步

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("乌龟跑了" + i \* 10 + "米"); // 显示兔子的跑步距离

}

System.out.println("乌龟到达终点");

}

}

public class Race {

public static void main(String[] args) {

// 创建乌龟跑步线程对象

Runnable runRabbit = new Rabbit();

Thread rabbit = new Thread(runRabbit);

rabbit.start(); // 启动兔子跑步线程

}

}

（2）

/\* RandomLottery.java \*/

package chapter10.practice3\_2;

import java.util.Random;

class RandomNunm extends Thread{

String name;

RandomNunm(String name) { // 有参构造方法，参数为线程名

this.name = name;

}

int[] array = new int[3]; // 定义长度为3的数组

public void run( ) {

Random random = new Random(); // 创建随机数对象random

for (int i = 0; i< 3; i++) {

array[i] = random.nextInt(10) + 1; // 生成1～10的随机数

try {

sleep(100); // 线程休眠0.1s

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println(name + "：" + array[0] + " " + array[1] + " " + array[2] + " ");

}

}

public class RandomLottery {

public static void main(String[] args) {

RandomNunm[] t = new RandomNunm[10]; // 定义线程数组，长度为10

// 循环创建RandomNunm类对象，通过有参构造方法命名线程，并启动线程

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

t[i] = new RandomNunm("线程" + String.valueOf(i + 1));

t[i].start();

}

}

}

（3）

/\* Children.java \*/

package chapter10.practice3\_3;

class WatchingTV extends Thread {

int minutes = 0;

@Override

public void run() {

while (true) {

try {

sleep(60000); // 线程休眠1分钟

System.out.println("小朋友看电视时长：" + (++minutes) + "分钟");

if (minutes == 20) {

interrupt(); // 线程中断

}

} catch (InterruptedException e) {

System.out.println("妈妈让他去写作业，小朋友关掉电视去写作业");

break; // 退出循环

}

}

}

}

public class Children {

public static void main(String[] args) {

new WatchingTV().start(); // 创建WatchingTV类对象，并启动线程

}

}

# 实训11 GUI编程

编程训练

（1）

/\* TraficLights.java \*/

package chapter11.practice1;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class TraficLights extends JFrame { // 创建一个“交通灯”类，并继承JFrame

public static void main(String[] args) {

TraficLights frame = new TraficLights(); // 创建TraficLights对象frame

frame.setLocationRelativeTo(null); // 在屏幕中居中显示

frame.setVisible(true); // 设置frame可见

}

// 创建JFrame窗体

public TraficLights() { // TraficLights的构造方法

setResizable(false); // 设置窗体不可改变大小

setTitle("交通灯"); // 设置窗体题目

setSize(220, 300); // 设置窗体大小

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);// 设置默认关闭方法

Container c = getContentPane(); // 创建Container类对象c

c.setLayout(new GridLayout(1, 0, 0, 0)); // 设置窗体布局为网格布局

// 创建面板ImagePanel置于c中，设置ImagePanel的背景色、边界布局

JPanel ImagePanel = new JPanel();

ImagePanel.setBackground(Color.WHITE);

c.add(ImagePanel);

ImagePanel.setLayout(new BorderLayout(0, 0));

// 创建标签lblImage置于ImagePanel的中间，设置标签的背景色、图标

JLabel lblImage = new JLabel("");

lblImage.setBackground(Color.WHITE);

lblImage.setIcon(new ImageIcon(TraficLights.class.getResource("green.png")));

ImagePanel.add(lblImage, BorderLayout.CENTER);

// 创建面板btnPanel置于c中，取消btnPanel的布局

JPanel btnPanel = new JPanel();

c.add(btnPanel);

btnPanel.setLayout(null);

// 创建单选按钮rbtnRed置于btnPanel中，设置单选按钮的文本、大小

// 添加动作监听事件，发生操作时，标签设置为红灯图标

JRadioButton rbtnRed = new JRadioButton("红灯");

rbtnRed.addActionListener(new ActionListener() { // 添加动作监听的事件

public void actionPerformed(ActionEvent e) { // 发生操作时

// 标签中添加图片

lblImage.setIcon(new ImageIcon(TraficLights.class.getResource("red.png")));

}

});

rbtnRed.setBounds(20, 51, 60, 20);

btnPanel.add(rbtnRed);

// 创建单选按钮rbtnYellow置于btnPanel中，设置单选按钮的文本、大小

// 添加动作监听事件，发生操作时，标签设置为黄灯图标

JRadioButton rbtnYellow = new JRadioButton("黄灯");

rbtnYellow.addActionListener(new ActionListener() { // 添加动作监听的事件

public void actionPerformed(ActionEvent e) { // 发生操作时

// 标签中添加图片

lblImage.setIcon(new ImageIcon(TraficLights.class.getResource("yellow.png")));

}

});

rbtnYellow.setBounds(20, 117, 60, 20);

btnPanel.add(rbtnYellow);

// 创建单选按钮rbtnGreen置于btnPanel中，设置单选按钮的文本、大小

// 添加动作监听事件，发生操作时，标签设置为绿灯图标

JRadioButton rbtnGreen = new JRadioButton("绿灯");

rbtnGreen.setSelected(true); // 设置按钮rbtnGreen为默认的单选按钮

rbtnGreen.addActionListener(new ActionListener() { // 添加动作监听的事件

public void actionPerformed(ActionEvent e) { // 发生操作时

// 标签中添加图片

lblImage.setIcon(new ImageIcon(TraficLights.class.getResource("green.png")));

}

});

rbtnGreen.setBounds(20, 182, 60, 20);

btnPanel.add(rbtnGreen);

// 创建ButtonGroup对象group

// 把单选按钮rbtnRed、rbtnYellow、rbtnGreen放进group中

ButtonGroup group = new ButtonGroup();

group.add(rbtnRed);

group.add(rbtnYellow);

group.add(rbtnGreen);

}

}

（2）

/\* KeyBoard.java \*/

package chapter11.practice2;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

public class KeyBoard extends Frame { // 创建“键盘”类继承Frame

// 声明窗体中的成员组件

private TextField textField;

private Button btnQ;

private Button btnW;

private Button btnE;

private Button btnR;

private Button btnT;

private Button btnY;

private Button btnU;

private Button btnI;

private Button btnO;

private Button btnP;

private Button btnA;

private Button btnS;

private Button btnD;

private Button btnF;

private Button btnG;

private Button btnH;

private Button btnJ;

private Button btnK;

private Button btnL;

private Button btnZ;

private Button btnX;

private Button btnC;

private Button btnV;

private Button btnB;

private Button btnN;

private Button btnM;

private Panel panel;

Color red = Color.RED; // 定义Color对象，表示按下键的颜色，设置为红色

// 定义Color对象，用来表示释放键的颜色，设置为浅灰色

Color gray = Color.LIGHT\_GRAY;

// 定义Font对象，表示字体，设置字体、样式和大小

Font font = new Font("Times New Roman", Font.PLAIN, 16);

// 定义列表btns，实现存储所有的按键ID

ArrayList<Button> btns=new ArrayList<Button>();

// 自定义方法，实现将容器中的所有Button组件添加到列表中

private void addButtons(){

for(Component cmp : panel.getComponents()){ // 遍历面板中的所有组件

if(cmp instanceof Button){ // 如果组件为Button类型

btns.add((Button)cmp); // 将Button组件添加到列表中

}

}

}

public static void main(String[] args) {

KeyBoard frame = new KeyBoard(); // 创建KeyBoard对象

frame.setLocationRelativeTo(null); // 在屏幕中居中显示

frame.setVisible(true); // 使frame可见

frame.addButtons(); // 初始化存储所有按键的集合

}

// 创建Frame窗体

public KeyBoard() { // KeyBoard的构造方法

setTitle("虚拟键盘（模拟键盘的按下与释放）"); // 设置窗体标题

setResizable(false); //不可改变窗体宽高

setSize(556, 280); //设置窗体的宽高

// 为窗体注册窗口事件侦听器

addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法，关闭当前窗口

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

// 设置窗体的背景色，并设置为边界布局

setBackground(Color.WHITE);

setLayout(new BorderLayout());

// 为关闭按钮添加窗口侦听器，关闭窗口

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

// 创建面板panel，添加到窗体中

// 放置在Center位置，并取消面板的布局管理器

panel = new Panel();

add(panel, BorderLayout.CENTER);

panel.setLayout(null);

// 创建按钮btnA～btnZ，添加到panel中

// 设置按钮的背景色、位置、宽高以及按钮中文字的字体

btnQ = new Button("Q");

btnQ.setBackground(gray);

btnQ.setFont(font);

btnQ.setBounds(0, 20, 47, 45);

panel.add(btnQ);

btnW = new Button("W");

btnW.setBackground(gray);

btnW.setFont(font);

btnW.setBounds(55, 20, 49, 45);

panel.add(btnW);

btnE = new Button("E");

btnE.setBackground(gray);

btnE.setFont(font);

btnE.setBounds(110, 20, 45, 45);

panel.add(btnE);

btnR = new Button("R");

btnR.setBackground(gray);

btnR.setFont(font);

btnR.setBounds(165, 20, 45, 45);

panel.add(btnR);

btnT = new Button("T");

btnT.setFont(font);

btnT.setBackground(gray);

btnT.setBounds(220, 20, 45, 45);

panel.add(btnT);

btnY = new Button("Y");

btnY.setFont(font);

btnY.setBackground(gray);

btnY.setBounds(275, 20, 45, 45);

panel.add(btnY);

btnU = new Button("U");

btnU.setFont(font);

btnU.setBackground(gray);

btnU.setBounds(330, 20, 45, 45);

panel.add(btnU);

btnI = new Button("I");

btnI.setFont(font);

btnI.setBackground(gray);

btnI.setBounds(385, 20, 45, 45);

panel.add(btnI);

btnO = new Button("O");

btnO.setFont(font);

btnO.setBackground(gray);

btnO.setBounds(440, 20, 46, 45);

panel.add(btnO);

btnP = new Button("P");

btnP.setFont(font);

btnP.setBackground(gray);

btnP.setBounds(495, 20, 45, 45);

panel.add(btnP);

btnA = new Button("A");

btnA.setFont(font);

btnA.setBackground(gray);

btnA.setBounds(23, 85, 45, 45);

panel.add(btnA);

btnS = new Button("S");

btnS.setFont(font);

btnS.setBackground(gray);

btnS.setBounds(82, 85, 45, 45);

panel.add(btnS);

btnD = new Button("D");

btnD.setBackground(gray);

btnD.setFont(font);

btnD.setBounds(137, 85, 45, 45);

panel.add(btnD);

btnF = new Button("F");

btnF.setBackground(gray);

btnF.setFont(font);

btnF.setBounds(195, 85, 45, 45);

panel.add(btnF);

btnG = new Button("G");

btnG.setFont(font);

btnG.setBackground(gray);

btnG.setBounds(251, 85, 45, 45);

panel.add(btnG);

btnH = new Button("H");

btnH.setFont(font);

btnH.setBackground(gray);

btnH.setBounds(306, 85, 45, 45);

panel.add(btnH);

btnJ = new Button("J");

btnJ.setFont(font);

btnJ.setBackground(gray);

btnJ.setBounds(361, 85, 45, 45);

panel.add(btnJ);

btnK = new Button("K");

btnK.setFont(font);

btnK.setBackground(gray);

btnK.setBounds(416, 85, 47, 45);

panel.add(btnK);

btnL = new Button("L");

btnL.setFont(font);

btnL.setBackground(gray);

btnL.setBounds(471, 85, 45, 45);

panel.add(btnL);

btnZ = new Button("Z");

btnZ.setFont(font);

btnZ.setBackground(gray);

btnZ.setBounds(39, 150, 45, 45);

panel.add(btnZ);

btnX = new Button("X");

btnX.setFont(font);

btnX.setBackground(gray);

btnX.setBounds(107, 150, 45, 45);

panel.add(btnX);

btnC = new Button("C");

btnC.setFont(font);

btnC.setBackground(gray);

btnC.setBounds(178, 150, 45, 45);

panel.add(btnC);

btnV = new Button("V");

btnV.setFont(font);

btnV.setBackground(gray);

btnV.setBounds(250, 150, 45, 45);

panel.add(btnV);

btnB = new Button("B");

btnB.setFont(font);

btnB.setBackground(gray);

btnB.setBounds(315, 150, 45, 45);

panel.add(btnB);

btnN = new Button("N");

btnN.setFont(font);

btnN.setBackground(gray);

btnN.setBounds(382, 150, 47, 45);

panel.add(btnN);

btnM = new Button("M");

btnM.setFont(font);

btnM.setBackground(gray);

btnM.setBounds(449, 150, 48, 45);

panel.add(btnM);

// 创建文本框textField，添加到panel中，并放置在窗体的North区域

textField = new TextField();

add(textField, BorderLayout.NORTH);

textField.addKeyListener(new KeyAdapter() { // 文本框添加键盘事件的监听

char word;

@Override

public void keyPressed(KeyEvent e) { // 按键被按下时被触发

word=e.getKeyChar(); // 获取按下键表示的字符

for(int i=0;i<btns.size();i++){ // 遍历存储按钮ID的ArrayList

//判断按键是否与遍历到的按钮的文本相同

if(String.valueOf(word).equalsIgnoreCase(btns.get(i).getLabel())){

btns.get(i).setBackground(red);//将指定按钮颜色设置为红色

}

}

}

@Override

public void keyReleased(KeyEvent e) { // 按键被释放时被触发

word=e.getKeyChar(); // 获取释放键表示的字符

for(int i=0;i<btns.size();i++){ // 遍历存储按键ID的ArrayList集合

// 判断按键是否与遍历到的按钮的文本相同

if(String.valueOf(word).equalsIgnoreCase(btns.get(i).getLabel())){

// 将指定按钮颜色设置为浅灰色

btns.get(i).setBackground(gray);

}

}

}

});

}

}

（3）

/\* LuckyViewer.java \*/

package chapter11.practice3;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class LuckyViewer extends JFrame {

// 声明窗体上的组件

private JLabel inputLabel;

private JLabel resultLabel;

private JTextField nameField;

private JTextArea nameArea;

private JTextArea resultArea;

private JButton btnExtract;

public static void main(String[] args) {

LuckyViewer frame = new LuckyViewer();

frame.setVisible(true);

}

public LuckyViewer() {

// 设置窗体的标签、大小和关闭

setTitle("抽取幸运观众");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(498, 300);

// 创建Container类对象c，并取消布局管理器

Container c = getContentPane();

c.setLayout(null);

// 创建面板namePanel，添加到c中

// 设置面板的位置和大小，取消布局管理器

JPanel namePanel = new JPanel();

namePanel.setBounds(10, 10, 174, 242);

c.add(namePanel);

namePanel.setLayout(null);

// 创建输入提示标签，添加到namePanel中，并设置位置和大小

inputLabel = new JLabel("输入在场观众姓名并按回车");

inputLabel.setBounds(0, 0, 170, 15);

namePanel.add(inputLabel);

// 创建输入姓名文本框，添加到namePanel中

// 设置位置和大小，以及添加按键事件侦听器

nameField = new JTextField();

nameField.addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

textField\_keyPressed(e);

}

});

nameField.setBounds(0, 20, 170, 20);

namePanel.add(nameField);

// 创建姓名列表文本区，添加到滚动面板scrollPane1中

// 将scrollPane1添加到c中，并设置scrollPane1的位置和大小

nameArea = new JTextArea();

nameArea.setEditable(false);

JScrollPane scrollPane1 = new JScrollPane(nameArea);

scrollPane1.setBounds(0, 45, 170, 200);

namePanel.add(scrollPane1);

// 创建面板resultPanel，添加到c中，设置位置和大小，以及设置为边界布局

JPanel resultPanel = new JPanel();

resultPanel.setBounds(183, 10, 219, 242);

c.add(resultPanel);

resultPanel.setLayout(new BorderLayout(0, 0));

// 创建抽取的幸运观众提示标签，添加到resultPanel的North区域

resultLabel = new JLabel("抽取的幸运观众");

resultPanel.add(resultLabel, BorderLayout.NORTH);

// 创建抽奖信息文本区，添加到滚动面板scrollPane2中

// 将scrollPane2添加到c中

resultArea = new JTextArea();

resultArea.setEditable(false);

JScrollPane scrollPane2 = new JScrollPane(resultArea);

resultPanel.add(scrollPane2);

// 创建抽取按钮，添加到c中，并设置位置和大小，以及添加动作事件侦听器

btnExtract = new JButton("抽取");

btnExtract.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnExtract\_actionPerformed(e);

}

});

btnExtract.setBounds(407, 215, 63, 25);

c.add(btnExtract);

//设置窗体在屏幕上居中放置，并不能改变大小

setLocationRelativeTo(null);

setResizable(false);

}

protected void textField\_keyPressed(KeyEvent e) {

if (e.getKeyChar() != '\n') // 不是回车字符不做处理

return;

String name = nameField.getText();

if (name.isEmpty()) // 如果文本框没有字符串不做处理

return;

nameArea.append(name + "\n"); // 把输入的姓名与回车符添加到姓名列表

nameField.selectAll(); // 选择文本框所有字符

}

protected void btnExtract\_actionPerformed(ActionEvent e) {

String perstring = nameArea.getText(); // 获取姓名列表文本

String[] personnelArray = perstring.split("\n"); // 获取姓名数组

int index = (int) (Math.random() \* personnelArray.length); // 生成随机数组索引

// 定义包含格式参数的中奖信息

String formatArg = "本次抽取的幸运观众：\n\t %1$s\n我们将颁发奖品：\n\t牛奶一箱\n\t鸡蛋一盒";

// 为中奖信息添加人员参数

String info = String.format(formatArg, personnelArray[index]);

resultArea.setText(info); // 在文本区显示中间信息

}

}

# 实训12 Java网络编程

编程训练

（1）

/\* GetAddress.java \*/

package chapter12.practice1;

import java.net.\*;

public class GetAddress {

public static void main(String args[]) {

InetAddress ip = null;

try {

// 获取指定主机的InetAddress对象

ip = InetAddress.getByName("www.163.com");

// 获取主机名并输出

System.out.println("主机名：" + ip.getHostName());

// 获取主机IP地址并输出

System.out.println("主机IP地址：" + ip.getHostAddress());

//获取本地主机名并输出

System.out.println("本地主机名:"

+ InetAddress.getLocalHost().getHostName());

// 获取本地主机IP地址

System.out.println("本机IP地址："

+ InetAddress.getLocalHost().getHostAddress());

} catch (UnknownHostException e) {

e.printStackTrace();

}

}

}

（2）

/\* TransferPhotoServer.java \*/

package chapter12.practice2;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

import javax.imageio.ImageIO;

import javax.swing.\*;

public class TransferPhotoServer extends JFrame {

private Image sendImg = null; // 声明发送的图像对象sendImg

private Image receiveImg = null; // 声明接收的图像对象receiveImg

// 声明发送的图像面板对象sendImagePanel

private SendImagePanel sendImagePanel = null;

// 声明接收图像面板对象receiveImagePanel

private ReceiveImagePanel receiveImagePanel = null;

private File imgFile = null; // 声明选择图片的File对象

private JTextField tf\_path; // 声明路径文本框tf\_path

private DataOutputStream out = null; // 声明数据字节输出流对象out

private DataInputStream in = null; // 创建数据字节输入流对象in

private ServerSocket server; // 声明ServerSocket对象server

private Socket socket; // 声明Socket对象socket

private long lengths = -1; // 声明图片文件的大小lengths

public void getServer() {

try {

server = new ServerSocket(1978); // 实例化Socket对象

while (true) { // 如果套接字是连接状态

socket = server.accept(); // 实例化Socket对象

out = new DataOutputStream(socket.getOutputStream()); // 获得out

in = new DataInputStream(socket.getInputStream()); // 获得in

getClientInfo(); // 调用getClientInfo()方法

}

} catch (Exception e) {

e.printStackTrace(); // 输出异常信息

}

}

private void getClientInfo() {

while (true) {

try {

long lengths = in.readLong(); // 读取图片文件的长度

byte[] bt = new byte[(int) lengths]; // 创建字节数组

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

bt[i] = in.readByte(); // 读取字节信息并存储到字节数组

}

receiveImg = new ImageIcon(bt).getImage(); // 创建图像对象

receiveImagePanel.repaint(); // 重新绘制图像

} catch (Exception e) {

}

}

}

public static void main(String[] args) {

// 创建TransferPhotoServer类对象

TransferPhotoServer frame = new TransferPhotoServer();

frame.getServer(); // 调用方法

}

public TransferPhotoServer() {

super("服务器");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(500, 400, 400, 300);

// 创建面板panel，添加路径标签和文本框、选择图片按钮、发送按钮

final JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

final JLabel label = new JLabel("路径：");

panel.add(label);

tf\_path = new JTextField(10);

panel.add(tf\_path);

sendImagePanel = new SendImagePanel();

receiveImagePanel = new ReceiveImagePanel();

final JButton btnSelect = new JButton("选择图片");

// 为选择图片按钮注册事件侦听器

btnSelect.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

JFileChooser fileChooser = new JFileChooser(); // 创建文件选择器

int flag = fileChooser.showOpenDialog(null); // 显示打开对话框

if (flag == JFileChooser.APPROVE\_OPTION) {

// 获取选中图片的File对象

imgFile = fileChooser.getSelectedFile();

}

if (imgFile != null) {

tf\_path.setText(imgFile.getAbsolutePath()); // 图片完整路径

try {

sendImg = ImageIO.read(imgFile);// 构造BufferedImage对象

} catch (IOException ex) {

ex.printStackTrace();

}

}

sendImagePanel.repaint(); // 调用paint()方法

}

});

panel.add(btnSelect);

final JButton btnSend = new JButton("发 送");

// 为发送按钮注册事件侦听器

btnSend.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

try {

DataInputStream inStream = null; // 定义数据输入流对象

if (imgFile != null) {

lengths = imgFile.length(); // 获得选择图片的大小

// 创建输入流对象

inStream=new DataInputStream(new FileInputStream(imgFile));

} else {

JOptionPane.showMessageDialog(null, "还没有选择图片文件");

return;

}

out.writeLong(lengths); // 将文件的大小写入输出流

byte[] bt = new byte[(int) lengths];// 创建字节数组

int len = -1;

// 将图片文件读取到字节数组

while ((len = inStream.read(bt)) != -1) {

out.write(bt);// 将字节数组写入输出流

}

} catch (IOException e1) {

e1.printStackTrace();

}

}

});

panel.add(btnSend);

// 创建显示图片面板

final JPanel showPanel= new JPanel();

showPanel.setLayout(new GridLayout(1, 0));

getContentPane().add(showPanel, BorderLayout.CENTER);

final JPanel sendPanel = new JPanel();

sendPanel.setLayout(new BorderLayout());

final JLabel sendLabel = new JLabel("服务器选择的要发送的图片");

sendPanel.add(sendLabel, BorderLayout.NORTH);

sendPanel.add(sendImagePanel, BorderLayout.CENTER);

final JPanel recvPanel = new JPanel();

recvPanel.setLayout(new BorderLayout());

final JLabel recvLabel = new JLabel("接收到客户端发送的图片");

recvPanel.add(recvLabel, BorderLayout.NORTH);

recvPanel.add(receiveImagePanel, BorderLayout.CENTER);

showPanel.add(sendPanel);

showPanel.add(recvPanel);

setVisible(true);

}

// 创建面板类，绘制发送的图片

class SendImagePanel extends JPanel {

public void paint(Graphics g) {

if (sendImg != null) {

// 清除绘图上下文的内容

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

// 绘制指定大小的图片

g.drawImage(sendImg, 0, 0, this.getWidth(), this.getHeight(), this);

}

}

}

// 创建面板类，绘制接收的图片

class ReceiveImagePanel extends JPanel {

public void paint(Graphics g) {

if (receiveImg != null) {

// 清除绘图上下文的内容

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

// 绘制指定大小的图片

g.drawImage(receiveImg, 0, 0, this.getWidth(), this.getHeight(), this);

}

}

}

}

/\* TransferPhotoClient.java \*/

package chapter12.practice2;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

import javax.imageio.ImageIO;

import javax.swing.\*;

public class TransferPhotoClient extends JFrame {

private Image sendImg = null; // 声明发送的图像对象sendImg

private Image receiveImg = null; // 声明接收的图像对象receiveImg

// 声明发送的图像面板对象sendImagePanel

private SendImagePanel sendImagePanel = null;

// 声明接收图像面板对象receiveImagePanel

private ReceiveImagePanel receiveImagePanel = null;

private File imgFile = null; // 声明选择图片的File对象

private JTextField tf\_path; // 声明路径文本框tf\_path

private DataOutputStream out = null; // 声明数据字节输出流对象out

private DataInputStream in = null; // 创建数据字节输入流对象in

private Socket socket; // 声明Socket对象socket

private long lengths = -1; // 声明图片文件的大小lengths

private void connect() { // 连接套接字方法

try { // 捕捉异常

socket = new Socket("localhost", 1978); // 实例化Socket对象

out = new DataOutputStream(socket.getOutputStream());// 获得输出流对象

in = new DataInputStream(socket.getInputStream()); // 获得输入流对象

getServerInfo(); // 调用getServerInfo()方法

} catch (Exception e) {

e.printStackTrace(); // 输出异常信息

}

}

public static void main(String[] args) { // 主方法

TransferPhotoClient client = new TransferPhotoClient(); // 创建本例对象

client.connect(); // 调用连接方法

}

private void getServerInfo() {

while (true) {

try {

long lengths = in.readLong(); // 读取图片文件的长度

byte[] bt = new byte[(int) lengths]; // 创建字节数组

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

bt[i] = in.readByte(); // 读取字节信息并存储到字节数组

}

receiveImg = new ImageIcon(bt).getImage();// 创建图像对象

receiveImagePanel.repaint(); // 重新绘制图像

} catch (IOException e) {

}

}

}

public TransferPhotoClient() {

super("客户端");

setBounds(1000, 400, 400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// 创建面板panel，添加路径标签和文本框、选择图片按钮、发送按钮

final JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

final JLabel label = new JLabel("路径：");

panel.add(label);

tf\_path = new JTextField(10);

panel.add(tf\_path);

sendImagePanel = new SendImagePanel();

receiveImagePanel = new ReceiveImagePanel();

final JButton btnSelect = new JButton("选择图片");

// 为选择图片按钮注册事件侦听器

btnSelect.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

JFileChooser fileChooser = new JFileChooser(); // 创建文件选择器

int flag = fileChooser.showOpenDialog(null); // 显示打开对话框

if (flag == JFileChooser.APPROVE\_OPTION) {

// 获取选中图片的File对象

imgFile = fileChooser.getSelectedFile();

}

if (imgFile != null) {

tf\_path.setText(imgFile.getAbsolutePath()); // 图片完整路径

try {

sendImg = ImageIO.read(imgFile);// 构造BufferedImage对象

} catch (IOException ex) {

ex.printStackTrace();

}

}

sendImagePanel.repaint(); // 调用paint()方法

}

});

panel.add(btnSelect);

final JButton btnSend = new JButton("发 送");

// 为发送按钮注册事件侦听器

btnSend.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

try {

DataInputStream inStream = null; // 定义数据输入流对象

if (imgFile != null) {

lengths = imgFile.length(); // 获得选择图片的大小

// 创建输入流对象

inStream = new DataInputStream(new FileInputStream(imgFile));

} else {

JOptionPane.showMessageDialog(null, "还没有选择图片文件");

return;

}

out.writeLong(lengths); // 将文件的大小写入输出流

byte[] bt = new byte[(int) lengths];// 创建字节数组

int len = -1;

// 将图片文件读取到字节数组

while ((len = inStream.read(bt)) != -1) {

out.write(bt); // 将字节数组写入输出流

}

} catch (IOException e1) {

e1.printStackTrace();

}

}

});

panel.add(btnSend);

// 创建显示图片面板

final JPanel showPanel= new JPanel();

showPanel.setLayout(new GridLayout(1, 0));

getContentPane().add(showPanel, BorderLayout.CENTER);

final JPanel sendPanel = new JPanel();

sendPanel.setLayout(new BorderLayout());

final JLabel sendLabel = new JLabel("客户端选择的要发送的图片");

sendPanel.add(sendLabel, BorderLayout.NORTH);

sendPanel.add(sendImagePanel, BorderLayout.CENTER);

final JPanel recvPanel = new JPanel();

recvPanel.setLayout(new BorderLayout());

final JLabel recvLabel = new JLabel("接收到服务器发送的图片");

recvPanel.add(recvLabel, BorderLayout.NORTH);

recvPanel.add(receiveImagePanel, BorderLayout.CENTER);

showPanel.add(sendPanel);

showPanel.add(recvPanel);

setVisible(true);

}

// 创建面板类，绘制发送的图片

class SendImagePanel extends JPanel {

public void paint(Graphics g) {

if (sendImg != null) {

// 清除绘图上下文的内容

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

// 绘制指定大小的图片

g.drawImage(sendImg, 0, 0, this.getWidth(), this.getHeight(), this);

}

}

}

// 创建面板类，绘制接收的图片

class ReceiveImagePanel extends JPanel {

public void paint(Graphics g) {

if (receiveImg != null) {

// 清除绘图上下文的内容

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

// 绘制指定大小的图片

g.drawImage(receiveImg, 0, 0, this.getWidth(), this.getHeight(), this);

}

}

}

}

（3）

/\* SendBroadcasts.java \*/

package chapter12.practice3;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

public class SendBroadcasts extends JFrame {

DatagramSocket socket = null; // 声明DatagramSocket对象socket

JTextField tf; // 声明文本框tf

JTextArea taSend; // 声明文本区taSend

int[] ports = { 2556, 2560, 2564, 2568 }; // 声明端口数组ports

public SendBroadcasts() throws SocketException {

super("发送广播"); // 设置标题

setBounds(500, 400, 400, 300); // 设置窗体位置和大小

// 创建面板panel，设置为网格布局，并添加文本框和发送按钮

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(1, 0));

tf = new JTextField(20);

panel.add(tf);

JButton btnSend = new JButton("发送");

panel.add(btnSend);

// 将文本区添加到滚动面板scrollPane中

taSend = new JTextArea();

taSend.setEditable(false);

JScrollPane scrollPane = new JScrollPane(taSend);

// 设置窗体为边界布局，将panel和scrollPane添加到窗体中

setLayout(new BorderLayout());

getContentPane().add(panel, BorderLayout.NORTH);

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

// 为发送按钮注册事件侦听器

btnSend.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

sendPacket(); // 调用sendPacket()方法

} catch (SocketException e1) {

e1.printStackTrace();

}

}

});

socket = new DatagramSocket(); // 创建DatagramSocket对象

setVisible(true); // 设置窗体可见

}

public void sendPacket() throws SocketException {

try {

// 获取本地地址，并赋值给address

InetAddress address=InetAddress.getByName("localhost");

String broadcast = tf.getText(); // 从文本框获取发送的广播

byte[] buf = broadcast.getBytes(); // 把字符串转换成字节数组

// 创建DatagramPacket类对象packet

// 同时指定发送的目标地址（address）和端口号（2556）

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

DatagramPacket packet = new DatagramPacket(buf, buf.length,

address, ports[i]);

socket.send(packet); // 服务器发送数据报

}

taSend.append(broadcast + "\n"); // 在文本区显示广播

} catch (IOException e) {

e.printStackTrace();

socket.close(); // 关闭socket

}

}

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

SendBroadcasts frame = new SendBroadcasts();

}

}

/\* RecvBroadcasts.java \*/

package chapter12.practice3;

import java.io.IOException;

import java.net.\*;

import javax.swing.\*;

public class RecvBroadcasts extends JFrame{

JTextArea taRecv; // 声明文本区taRecv

public RecvBroadcasts() {

super("接收广播"); // 设置窗体标题

setBounds(1000, 400, 400, 300); // 设置窗体位置和大小

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 设置默认关闭方法

// 将taRecv添加到滚动面板scrollPane中

taRecv = new JTextArea();

taRecv.setEditable(false);

JScrollPane scrollPane = new JScrollPane(taRecv);

getContentPane().add(scrollPane); // 将scrollPane添加到窗体中

setVisible(true); // 设置窗体可见

}

public void recvPacket() throws SocketException {

// 创建DatagramSocket对象，同时指定发送的端口号

DatagramSocket socket = new DatagramSocket(2556);

byte[] buf = new byte[256];

// 创建接收数据报的DatagramPacket对象

while (true) {

DatagramPacket packet = new DatagramPacket(buf, buf.length);

try {

socket.receive(packet); // 客户端接收数据报

// 将接收的数据报转换为字符串

String received = new String(packet.getData());

taRecv.append(received + "\n"); // 将received显示在taRecv中

} catch (IOException e) {

e.printStackTrace();

socket.close(); // 关闭socket

break; // 退出循环

}

}

}

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

RecvBroadcasts frame = new RecvBroadcasts();

frame.recvPacket(); // 调用recvPacket()方法

}

}

# 实训13 Java数据库编程

编程训练

/\* Login.java \*/

package chapter13.practice;

import javax.swing.\*;

import java.awt.Container;

import java.awt.event.\*;

import java.sql.\*;

public class Login extends JFrame {

JTextField text\_user;

JPasswordField text\_pass;

public Login() {

setTitle("登录");

setLayout(null);

setSize(260, 180);

// 添加标签与文本框

JLabel lbUser = new JLabel("用户名：");

JLabel lbPass = new JLabel("密 码：");

JButton btn\_ok = new JButton("确定");

JButton btn\_cancel = new JButton("取消");

text\_pass = new JPasswordField();

text\_user = new JTextField();

lbUser.setBounds(35, 20, 60, 20);

lbPass.setBounds(35, 60, 60, 20);

text\_user.setBounds(85, 20, 120, 20);

text\_pass.setBounds(85, 60, 120, 20);

btn\_ok.setBounds(35, 100, 80, 25); // 确定按钮

btn\_ok.addActionListener((new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_okActionPerformed(e);

}

}));

btn\_cancel.setBounds(125, 100, 80, 25); // 取消按钮

btn\_cancel.addActionListener((new ActionListener() {

public void actionPerformed(ActionEvent e) {

//DbOp.Close(); // 关闭数据库

System.exit(0);

}

}));

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

//DbOp.Close(); // 关闭数据库

System.exit(0);

}

});

Container c = getContentPane();

c.add(lbUser);

c.add(lbPass);

c.add(text\_user);

c.add(text\_pass);

c.add(btn\_ok);

c.add(btn\_cancel);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

}

public void btn\_okActionPerformed(ActionEvent e) {

String user = text\_user.getText();

String pass = String.valueOf(text\_pass.getPassword());

String is\_admin;

// 如果用户名或密码任一为空，则终止后续操作

if (user.equals("")||pass.equals("")) {

JOptionPane.showMessageDialog(null, "用户名或密码不能为空！");

return;

}

try {

// 核对用户名和密码

String sql = "select \* from tb\_user where username=" + "'" + user

+ "'and password=" + "'" + pass + "'";

ResultSet rs = DbOp.executeQuery(sql);

// 如果此用户存在，则记录其状态（否：不是管理员，是：是管理员）

if (rs.next()) {

is\_admin = rs.getString("is\_admin");

} else {

JOptionPane.showMessageDialog(null, "用户名或密码不正确！");

return;

}

GlobalVar.login\_user = user; // 记录登录的用户名

ShowMain show = new ShowMain(); // 调用主程序

// 只有管理员才能使用"基础管理"菜单

show.setRights(is\_admin);

// 释放窗体及其全部组件的屏幕资源，即使释放登录窗体

dispose(); // 释放当前窗体

} catch (SQLException e1) {

JOptionPane.showMessageDialog(null, "用户数据库有误！");

}

}

public static void main(String[] args) {

new Login();

}

}

/\* ShowMain.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class ShowMain extends JFrame {

JMenuBar menuBar1;

JMenu menu1, menu2, menu3, menu4, menu5, menu6, menu7;

JMenuItem mi\_book\_add, mi\_book\_update, mi\_book\_delete, mi\_reader\_add,

mi\_reader\_update, mi\_reader\_delete, mi\_borrow, mi\_back,

mi\_query\_book, mi\_query\_reader, mi\_update\_pass, mi\_exit;

public void setRights(String rights) {

// 如果不是管理员，则禁止用户维护图书信息和读者信息

// 以及禁止进行借阅管理，即只能查询

if (rights.equals("否")) {

menu5.setEnabled(false);

}

}

public ShowMain() {

setTitle("图书管理系统");

setLayout(new BorderLayout());

setSize(400, 300);

menuBar1 = new JMenuBar();

menu5 = new JMenu("数据管理"); // 基础维护菜单

menu6 = new JMenu("图书管理"); // 图书维护菜单

mi\_book\_add = new JMenuItem("添加图书"); // 添加图书菜单

mi\_book\_update = new JMenuItem("更新图书"); // 修改图书菜单

mi\_book\_delete = new JMenuItem("删除图书"); // 删除图书菜单

menu7 = new JMenu("读者管理");// 读者维护菜单

mi\_reader\_add = new JMenuItem("添加读者"); // 添加读者菜单

mi\_reader\_update = new JMenuItem("更新读者");

mi\_reader\_delete = new JMenuItem("删除读者");

menu1 = new JMenu("借阅管理");

mi\_borrow = new JMenuItem("借书管理");

mi\_back = new JMenuItem("还书管理");

menu2 = new JMenu("查询管理");

mi\_query\_book = new JMenuItem("图书查询");

mi\_query\_reader = new JMenuItem("读者查询");

menu3 = new JMenu("系统管理");

mi\_update\_pass = new JMenuItem("修改密码");

mi\_exit = new JMenuItem("退出系统");

// 添加图书菜单

mi\_book\_add.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new BookOperate("添加图书");

}

});

mi\_book\_update.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new BookOperate("更新图书");

}

});

mi\_book\_delete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new BookOperate("删除图书");

}

});

mi\_reader\_add.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ReaderOperate("添加读者");

}

});

mi\_reader\_update.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ReaderOperate("更新读者");

}

});

mi\_reader\_delete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ReaderOperate("删除读者");

}

});

mi\_borrow.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new Borrow();

}

});

mi\_back.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new Back();

}

});

mi\_query\_book.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new BookQuery();

}

});

mi\_query\_reader.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ReaderQuery();

}

});

mi\_update\_pass.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new UpdatePassword();

}

});

mi\_exit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

DbOp.Close(); // 关闭数据库

System.exit(0);

}

});

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

DbOp.Close(); // 关闭数据库

System.exit(0);

}

});

menu5.add(menu6); menu6.add(mi\_book\_add);

menu6.add(mi\_book\_update); menu6.add(mi\_book\_delete);

menu5.add(menu7); menu7.add(mi\_reader\_add);

menu7.add(mi\_reader\_update); menu7.add(mi\_reader\_delete);

menu5.add(menu1); menu1.add(mi\_borrow);

menu1.add(mi\_back); menu5.add(menu2);

menu2.add(mi\_query\_book); menu2.add(mi\_query\_reader);

menu5.add(menu3); menu3.add(mi\_update\_pass);

menu3.add(mi\_exit); menuBar1.add(menu5);

menuBar1.add(menu1); menuBar1.add(menu2);

menuBar1.add(menu3);

setJMenuBar(menuBar1);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new ShowMain();

}

}

/\* BookOperate.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import java.text.\*;

import javax.swing.\*;

public class BookOperate extends JFrame {

JLabel lbbookid\_1 = new JLabel("图书编号");

JLabel lbbookid = new JLabel("图书编号");

JLabel lbbookname = new JLabel("图书名称");

JLabel lbbooktype = new JLabel("图书类别");

JLabel lbauthor = new JLabel("作者");

JLabel lbtranslator = new JLabel("译者");

JLabel lbpublisher = new JLabel("出版社");

JLabel lbpublish\_time = new JLabel("出版时间");

JLabel lbprice = new JLabel("定价");

JLabel lbstock = new JLabel("库存数量");

JButton btn = new JButton();

JButton closeBtn = new JButton("关闭");

JButton queryBtn = new JButton("查询");

JTextField tf\_bookid = new JTextField();

JTextField tf\_bookname = new JTextField();

JTextField tf\_author = new JTextField();

JTextField tf\_translator = new JTextField();

JTextField tf\_publisher = new JTextField();

JTextField tf\_publish\_time = new JTextField();

JTextField tf\_price = new JTextField();

JTextField tf\_stock = new JTextField();

JTextField tf\_bookid1 = new JTextField();

JComboBox<String> tf\_booktype = new JComboBox<String>();

String title;

public BookOperate(String title) {

setLayout(null);

setTitle(title);

setSize(440, 310);

this.title = title;

if (title == "删除图书") {

tf\_bookid.setEditable(false);

btn.setText("删除");

}

else if ( title == "更新图书") {

tf\_bookid.setEditable(false);

btn.setText("保存");

}

else if (title == "添加图书") {

lbbookid\_1.setVisible(false);

tf\_bookid1.setVisible(false);

queryBtn.setVisible(false);

btn.setText("保存");

}

lbbookid\_1.setBounds(80, 20, 50, 20);// 图书编号

tf\_bookid1.setBounds(140, 20, 100, 20);

queryBtn.setBounds(260, 20, 80, 20); // 查询按钮

queryBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_queryActionPerformed(e);

}

});

lbbookid.setBounds(30, 60, 50, 20); // 图书编号

tf\_bookid.setBounds(90, 60, 100, 20);

lbbookname.setBounds(220, 60, 50, 20); // 图书名称

tf\_bookname.setBounds(280, 60, 100, 20);

lbbooktype.setBounds(30, 90, 50, 20); // 图书类别

tf\_booktype.setBounds(90, 90, 100, 20);

tf\_booktype.addItem("科技");

tf\_booktype.addItem("文学");

tf\_booktype.addItem("社科");

tf\_booktype.addItem("其他");

lbauthor.setBounds(220, 90, 50, 20); // 作者

tf\_author.setBounds(280, 90, 100, 20);

lbtranslator.setBounds(30, 120, 50, 20); // 译者

tf\_translator.setBounds(90, 120, 100, 20);

lbpublisher.setBounds(220, 120, 50, 20); // 出版社

tf\_publisher.setBounds(280, 120, 100, 20);

lbpublish\_time.setBounds(30, 150, 50, 20); // 出版时间

tf\_publish\_time.setBounds(90, 150, 100, 20);

lbprice.setBounds(220, 150, 50, 20); // 价格

tf\_price.setBounds(280, 150, 100, 20);

lbstock.setBounds(30, 180, 50, 20); // 库存数量

tf\_stock.setBounds(90, 180, 100, 20);

btn.setBounds(100,220, 80, 25); // 删除按钮

btn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (title == "删除图书")

btn\_delActionPerformed(e);

else if (title == "添加图书" || title == "更新图书") {

btn\_saveActionPerformed(e);

}

}

});

closeBtn.setBounds(250, 220, 80, 25); // 关闭按钮

closeBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose(); // 释放当前窗体

}

});

/\* 关闭窗口 \*/

addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

dispose(); // 释放当前窗体

}

});

Container c = getContentPane();

c.add(lbbookid); // 将各组件增加到窗体中

c.add(lbbookid\_1); c.add(lbbookname); c.add(lbbooktype);

c.add(lbauthor); c.add(lbtranslator); c.add(lbpublisher);

c.add(lbpublish\_time); c.add(lbprice); c.add(lbstock);

c.add(btn); c.add(closeBtn); c.add(queryBtn);

c.add(tf\_bookid); c.add(tf\_bookname); c.add(tf\_author);

c.add(tf\_translator); c.add(tf\_publisher); c.add(tf\_publish\_time);

c.add(tf\_price); c.add(tf\_stock); c.add(tf\_bookid1);

c.add(tf\_booktype);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

}

private void btn\_delActionPerformed(ActionEvent e) {

String id = tf\_bookid.getText();

// 如果图书编号为空，则删除操作终止

if (id.equals("")) {

JOptionPane.showMessageDialog(null, "图书编号不能为空！");

return;

}

String sql = "delete from tb\_book where id='" + id + "'";

int i = DbOp.executeUpdate(sql);

if (i == 1) {

JOptionPane.showMessageDialog(null, "图书信息删除成功!");

// 清空全部文本框

clearAllTextfield();

} else

JOptionPane.showMessageDialog(null, "图书信息删除失败！");

}

private void btn\_queryActionPerformed(ActionEvent e) {

String id = tf\_bookid1.getText();

// 如果图书编号为空，则查询操作终止

if (id.equals("")) {

JOptionPane.showMessageDialog(null, "图书编号不能为空！");

return;

}

// 按编号查询图书，结果存入book对象中

Book book = BookSelect.SelectBookById(id);

// 如果查询到结果，将其显示在各文本框中

if (book != null) {

tf\_bookid.setText(book.getId());

tf\_bookid.setEditable(false);

tf\_bookname.setText(book.getBookname());

// 将Choice的选定项设置为其名称等于指定字符串的项

tf\_booktype.setSelectedItem(book.getBooktype());

tf\_author.setText(book.getAuthor());

tf\_translator.setText(book.getTranslator());

tf\_publisher.setText(book.getPublisher());

tf\_publish\_time.setText(book.getPublish\_time().toString());

tf\_price.setText(String.valueOf((book.getPrice())));

tf\_stock.setText(String.valueOf(book.getStock()));

} else

JOptionPane.showMessageDialog(null, "图书编号有误，查无此书！");

}

// 保存添加或更新的记录

private void btn\_saveActionPerformed(ActionEvent e) {

String id = tf\_bookid.getText();

String bookname = tf\_bookname.getText();

String booktype = tf\_booktype.getSelectedItem().toString();

String author = tf\_author.getText();

String translator = tf\_translator.getText();

String publisher = tf\_publisher.getText();

String publish\_time = tf\_publish\_time.getText();

// 如果图书编号为空，则终止保存记录操作

if (id.equals("")) {

JOptionPane.showMessageDialog(null, "图书编号不能为空！");

return;

}

if (title == "添加图书" & IfBookIdExit(id)) {

JOptionPane.showMessageDialog(null, "图书编号重复！");

return;

}

try {

// -------------------------------------------------

// 以下程序用于检查日期是否有效。如果日期无效，则会

// 产生ParseException异常

// 创建一个简单日期格式对象，注意：MM一定要用大写

// 这是用户输入日期的格式：年-月，如2010-7、2009-10等

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM");

// 将字符串转换为日期

sdf.parse(tf\_publish\_time.getText());

// -------------------------------------------------

float price = Float.parseFloat(tf\_price.getText());

int stock = Integer.parseInt(tf\_stock.getText());

// 将记录保存到tb\_book表中

String sql = "";

if (title == "添加图书") {

sql = "insert into tb\_book values('" + id

+ "','" + bookname + "','" + booktype + "','" + author

+ "','" + translator + "','" + publisher + "','"

+ publish\_time + "','" + price + "','" + stock + "')";

} if (title == "更新图书") {

sql = "update tb\_book set bookname='" + bookname

+ "',booktype='" + booktype + "',author='" + author

+ "',translator='" + translator + "',publisher='"

+ publisher + "',publish\_time='" + publish\_time

+ "',price='" + price + "',stock='" + stock

+ "' where id='" + id + "'";

}

int i = DbOp.executeUpdate(sql);

if (i == 1) {

JOptionPane.showMessageDialog(null, "图书信息保存成功！");

// 清空全部文本框

clearAllTextfield();

} else

JOptionPane.showMessageDialog(null, "数据有误，图书信息保存失败！");

} catch (ParseException e2) {

JOptionPane.showMessageDialog(null, "出版时间格式错误（年-月-日）！");

} catch (NumberFormatException e1) {

JOptionPane.showMessageDialog(null, "价格或库存数量错误，应为数字！");

}

}

// 判断Book表中是否存在指定编号的图书，如果存在，返回true，否则，返回false

private boolean IfBookIdExit(String id) {

String sql = "select \* from tb\_book where id='" + id + "'";

ResultSet rs = DbOp.executeQuery(sql);

try {

if (rs.next())

return true;

else

return false;

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "无法正常读取数据库！");

}

return false;

}

// 清空全部文本框

private void clearAllTextfield() {

tf\_bookid1.setText(""); tf\_bookid.setText(""); tf\_bookname.setText("");

tf\_author.setText(""); tf\_translator.setText(""); tf\_publisher.setText("");

tf\_publish\_time.setText(""); tf\_price.setText(""); tf\_stock.setText("");

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new BookOperate("添加图书");

}

}

/\* ReaderOperate.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

public class ReaderOperate extends JFrame {

JLabel lbreaderid = new JLabel("读者编号");

JLabel lbreaderid\_1 = new JLabel("读者编号");

JLabel lbreadername = new JLabel("读者姓名");

JLabel lbreadertype = new JLabel("读者类别");

JLabel lbsex = new JLabel("性别");

JLabel lbmax\_num = new JLabel("可借数量");

JLabel lbdays\_num = new JLabel("可借天数");

JTextField tf\_readerid = new JTextField();

JTextField tf\_readername = new JTextField();

JTextField tf\_max\_num = new JTextField();

JTextField tf\_days\_num = new JTextField();

JTextField tf\_readerid1 = new JTextField();

JComboBox<String> tf\_readertype = new JComboBox<String>();

JComboBox<String> tf\_sex = new JComboBox<String>();

JButton queryBtn = new JButton("查询");

JButton btn = new JButton();

JButton closeBtn = new JButton("关闭");

String title;

public ReaderOperate(String title) {

setLayout(null);

setTitle(title);

setSize(430, 240);

this.title = title;

if (title == "删除读者") {

tf\_readerid.setEditable(false);

btn.setText("删除");

}

else if ( title == "更新读者") {

tf\_readerid.setEditable(false);

btn.setText("保存");

}

else if (title == "添加读者") {

lbreaderid\_1.setVisible(false);

tf\_readerid1.setVisible(false);

queryBtn.setVisible(false);

btn.setText("保存");

}

lbreaderid\_1.setBounds(80, 20, 50, 20); // 读者编号

tf\_readerid1.setBounds(140, 20, 100, 20);

queryBtn.setBounds(270, 20, 80, 20); // 查询按钮

queryBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_queryActionPerformed(e);

}

});

lbreaderid.setBounds(30, 60, 50, 20); // 读者编号

tf\_readerid.setBounds(90, 60, 100, 20);

lbreadername.setBounds(220, 60, 50, 20); // 读者姓名

tf\_readername.setBounds(280, 60, 100, 20);

lbreadertype.setBounds(30, 90, 50, 20); // 读者类别

tf\_readertype.setBounds(90, 90, 100, 20);

tf\_readertype.addItem("教师");

tf\_readertype.addItem("学生");

tf\_readertype.addItem("职工");

lbsex.setBounds(220, 90, 50, 20); // 性别

tf\_sex.setBounds(280, 90, 100, 20);

tf\_sex.addItem("男");

tf\_sex.addItem("女");

lbmax\_num.setBounds(30, 120, 50, 20); // 最大可借数

tf\_max\_num.setBounds(90, 120, 100, 20);

lbdays\_num.setBounds(220, 120, 50, 20); // 最大可借天数

tf\_days\_num.setBounds(280, 120, 100, 20);

btn.setBounds(100, 160, 80, 25); // 删除按钮

btn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (title == "删除读者")

btn\_delActionPerformed(e);

else if (title == "添加读者" || title == "更新读者")

btn\_saveActionPerformed(e);

}

});

closeBtn.setBounds(240, 160, 80, 25); // 关闭按钮

closeBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose(); // 释放当前窗体

}

});

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

dispose(); // 释放当前窗体

}

});

Container c = getContentPane();

c.add(lbreaderid); // 将各组件添加到窗体中

c.add(lbreaderid\_1); c.add(lbreadername); c.add(lbreadertype);

c.add(lbsex); c.add(lbmax\_num); c.add(lbdays\_num);

c.add(tf\_readerid); c.add(tf\_readername); c.add(tf\_max\_num);

c.add(tf\_days\_num); c.add(tf\_readerid1); c.add(tf\_readertype);

c.add(tf\_sex); c.add(btn); c.add(closeBtn);

c.add(queryBtn);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

}

private void btn\_delActionPerformed(ActionEvent e) {

String id = tf\_readerid.getText();

// 如果读者编号为空，则终止删除操作

if (id.equals("")) {

JOptionPane.showMessageDialog(null, "读者编号不能为空！");

return;

}

String sql = "delete from tb\_reader where id='" + id + "'";

int i = DbOp.executeUpdate(sql);

if (i == 1) {

JOptionPane.showMessageDialog(null, "读者信息删除成功！");

// 清空全部文本框

clearAllTextfield();

} else

JOptionPane.showMessageDialog(null, "读者编号有误，查无此人！");

}

private void btn\_queryActionPerformed(ActionEvent e) {

String id = tf\_readerid1.getText();

if (id.equals("")) {

JOptionPane.showMessageDialog(null, "读者编号不能为空！");

return;

}

// 按编号查询读者，结果存入reader对象中

Reader reader = ReaderSelect.selectReaderById(id);

// 如果查询到结果，将其显示在各文本框中

if (reader != null) {

tf\_readerid.setText(reader.getId());

tf\_readerid.setEditable(false);

tf\_readername.setText(reader.getReadername());

// 将Choice的选定项设置为其名称等于指定字符串的项

tf\_readertype.setSelectedItem(reader.getReadertype());

tf\_sex.setSelectedItem(reader.getSex());

tf\_max\_num.setText(String.valueOf(reader.getMax\_num()));

tf\_days\_num.setText(String.valueOf(reader.getDays\_num()));

} else

JOptionPane.showMessageDialog(null, "读者编号有误，查无此人！");

}

private void btn\_saveActionPerformed(ActionEvent e) {

String id = tf\_readerid.getText();

String name = tf\_readername.getText();

String type = tf\_readertype.getSelectedItem().toString();

String sex = tf\_sex.getSelectedItem().toString();

// 如果读者编号为空，则终止保存记录操作

if (id.equals("")) {

JOptionPane.showMessageDialog(null, "读者编号不能为空！");

return;

}

// 如果读者编号重复，则记录无效，需修改读者编号

if (title == "添加读者" & IfReaderIdExit(id)) {

JOptionPane.showMessageDialog(null, "读者编号重复！");

return;

}

try {

int max\_num = Integer.parseInt(tf\_max\_num.getText());

int days\_num = Integer.parseInt(tf\_days\_num.getText());

String sql = "";

if (title == "添加读者") {

sql = "insert into tb\_reader values('" + id + "','" + name + "','"

+ type + "','" + sex + "','" + max\_num + "','" + days\_num

+ "')";

} else if (title == "更新读者") {

sql = "update tb\_reader set readername='" + name

+ "',readertype='" + type + "',days\_num='" + days\_num

+ "',sex='" + sex + "',max\_num='" + max\_num

+ "' where id='" + id + "'";

}

int i = DbOp.executeUpdate(sql);

if (i == 1) {

JOptionPane.showMessageDialog(null, "读者添加成功！");

// 清空全部文本框

clearAllTextfield();

}

} catch (NumberFormatException e1) {

JOptionPane.showMessageDialog(null, "最大可借数或"

+ "最大可借天数错误，应为整数！");

}

}

// 判断Reader表中是否存在指定编号的读者，如果存在，返回true，否则，返回false

private boolean IfReaderIdExit(String id) {

String sql = "select \* from tb\_reader where id='" + id + "'";

ResultSet rs = DbOp.executeQuery(sql);

try {

if (rs.next())

return true;

else

return false;

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "无法正常读取数据库！");

}

return false;

}

// 清空全部文本框

private void clearAllTextfield() {

tf\_readerid1.setText(""); tf\_readerid.setText("");

tf\_readername.setText(""); tf\_max\_num.setText("");

tf\_days\_num.setText("");

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new ReaderOperate("更新读者");

}

}

/\* Borrow.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.\*;

public class Borrow extends JFrame {

String SepLine = "-------------------------------------------------";

JLabel lbbookid = new JLabel("图书编号");

JLabel lbreaderid = new JLabel("读者编号");

JTextField tf\_bookid = new JTextField();

JTextField tf\_readerid = new JTextField();

JButton queryBtn = new JButton("查询");

JLabel lbbookinfo = new JLabel(SepLine + "图书信息" + SepLine);

JLabel lbbookname = new JLabel("图书名称：");

JLabel tf\_bookname = new JLabel("xx");

JLabel lbauthor = new JLabel("作者：");

JLabel tf\_author = new JLabel("xx");

JLabel lbpublisher = new JLabel("出版社：");

JLabel tf\_publisher = new JLabel("xx");

JLabel lbpublish\_time = new JLabel("出版时间：");

JLabel tf\_publish\_time = new JLabel("xx");

JLabel lbprice = new JLabel("定价：");

JLabel tf\_price = new JLabel("xx");

JLabel lbstock = new JLabel("库存数量：");

JLabel tf\_stock = new JLabel("xx");

JLabel lbreaderinfo = new JLabel(SepLine + "读者信息" + SepLine);

JLabel lbreadername = new JLabel("读者姓名：");

JLabel tf\_readername = new JLabel("xx");

JLabel lbreadertype = new JLabel("读者类型：");

JLabel tf\_readertype = new JLabel("xx");

JLabel lbmax\_num = new JLabel("最大可借数：");

JLabel tf\_max\_num = new JLabel("xx");

JLabel lbdays\_num = new JLabel("最大可借天数：");

JLabel tf\_days\_num = new JLabel("xx");

JLabel lbborrowinfo = new JLabel(SepLine + "借阅信息" + SepLine);

JLabel lbborrowednum = new JLabel("该读者已借图书数量：");

JLabel tf\_borrowednum = new JLabel("xx");

JLabel lbif\_borrow = new JLabel("该读者是否可借所选图书：");

JLabel tf\_if\_borrow = new JLabel("xx");

JLabel lbborrow\_date = new JLabel("借阅日期：");

JLabel tf\_borrow\_date = new JLabel("xx");

JButton borrowBtn = new JButton("借书");

JButton closeBtn = new JButton("关闭");

public Borrow() {

setLayout(null);

setTitle("借书管理");

setSize(500, 420);

lbbookid.setBounds(30, 20, 50, 25); // 图书编号

tf\_bookid.setBounds(90, 20, 90, 20);

lbreaderid.setBounds(200, 20, 50, 25); // 读者编号

tf\_readerid.setBounds(260, 20, 90, 20);

queryBtn.setBounds(370, 20, 80, 25); // 查询按钮

lbbookinfo.setBounds(30, 50, 440, 25); // 图书信息提示条

lbbookname.setBounds(30, 80, 60, 25); // 图书名称

tf\_bookname.setBounds(90, 80, 200, 25);

lbauthor.setBounds(310, 80, 60, 25); // 作者

tf\_author.setBounds(370, 80, 90, 25);

lbpublisher.setBounds(30, 105, 60, 25); // 出版社

tf\_publisher.setBounds(90, 105, 200, 25);

lbpublish\_time.setBounds(310, 105, 60, 25); // 出版时间

tf\_publish\_time.setBounds(370, 105, 90, 25);

lbprice.setBounds(30, 130, 60, 25); // 定价

tf\_price.setBounds(90, 130, 200, 25);

lbstock.setBounds(310, 130, 60, 25); // 库存数量

tf\_stock.setBounds(370, 130, 90, 25);

lbreaderinfo.setBounds(30, 160, 440, 25); // 读者信息提示条

lbreadername.setBounds(30, 185, 60, 25); // 读者姓名

tf\_readername.setBounds(90, 185, 90, 25);

lbreadertype.setBounds(310, 185, 60, 25); // 读者类型

tf\_readertype.setBounds(370, 185, 90, 25);

lbmax\_num.setBounds(30, 210, 75, 25); // 最大可借数

tf\_max\_num.setBounds(105, 210, 90, 25);

lbdays\_num.setBounds(310, 210, 85, 25); // 最大可借天数

tf\_days\_num.setBounds(395, 210, 70, 25);

lbborrowinfo.setBounds(30, 240, 440, 25); // 借阅信息提示条

lbborrowednum.setBounds(30, 265, 120, 25); // 已借图书数量

tf\_borrowednum.setBounds(150, 265, 50, 25);

lbif\_borrow.setBounds(30, 290, 145, 25); // 是否可借

tf\_if\_borrow.setBounds(175, 290, 50, 25);

lbborrow\_date.setBounds(30, 315, 60, 25); // 借书日期

tf\_borrow\_date.setBounds(90, 315, 100, 25);

borrowBtn.setBounds(160, 345, 80, 25); // 借出按钮

borrowBtn.setEnabled(false); // 开始时禁用借出按钮

closeBtn.setBounds(260, 345, 80, 25); // 关闭按钮

queryBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_querywActionPerformed(e);

}

});

borrowBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_borrowActionPerformed(e);

}

});

closeBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose(); // 关闭窗体

}

});

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

dispose(); // 关闭窗体

}

});

Container c = getContentPane();

c.add(lbbookid); c.add(lbreaderid); c.add(queryBtn);

c.add(lbbookinfo); c.add(lbbookname); c.add(lbauthor);

c.add(lbpublisher); c.add(lbpublish\_time); c.add(lbprice);

c.add(lbstock); c.add(lbreaderinfo); c.add(lbreadername);

c.add(lbreadertype); c.add(lbmax\_num); c.add(lbdays\_num);

c.add(lbborrowinfo); c.add(lbborrowednum); c.add(lbif\_borrow);

c.add(lbborrow\_date); c.add(borrowBtn); c.add(closeBtn);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

c.setBackground(Color.white);

c.add(tf\_bookid); c.add(tf\_readerid); c.add(tf\_bookname);

c.add(tf\_author); c.add(tf\_publisher); c.add(tf\_publish\_time);

c.add(tf\_price); c.add(tf\_stock); c.add(tf\_readername);

c.add(tf\_readertype); c.add(tf\_max\_num); c.add(tf\_days\_num);

c.add(tf\_borrowednum); c.add(tf\_if\_borrow); c.add(tf\_borrow\_date);

}

// 图书和读者查询

private void btn\_querywActionPerformed(ActionEvent e) {

String bookid = tf\_bookid.getText();

String readerid = tf\_readerid.getText();

// 如果图书编号或读者编号两者均为空，或者有一个为空，则返回

if (bookid.equals("") || readerid.equals("")) {

JOptionPane.showMessageDialog(null, "图书编号和读者编号均不能为空！");

init(); // 重新初始化各参数并禁止借出按钮

return;

}

// 按编号查询图书，结果存入book对象中

Book book = BookSelect.SelectBookById(bookid);

// 如果查询到结果，将其显示在各文本框中

if (book != null) {

tf\_bookname.setText(book.getBookname());

tf\_author.setText(book.getAuthor());

tf\_publisher.setText(book.getPublisher());

tf\_publish\_time.setText(book.getPublish\_time().toString());

tf\_price.setText(String.valueOf((book.getPrice())));

tf\_stock.setText(String.valueOf(book.getStock()));

} else {

JOptionPane.showMessageDialog(null, "图书编号有误，查无此书！");

init(); // 重新初始化各参数并禁止借出按钮

return;

}

if (book.getStock() == 0) {

JOptionPane.showMessageDialog(null, "图书已无库存，无法借阅！");

init(); // 重新初始化各参数并禁止借出按钮

return;

}

// 按编号查询读者，结果存入reader对象中

Reader reader = ReaderSelect.selectReaderById(readerid);

// 如果查询到结果，将其显示在各文本框中

if (reader != null) {

tf\_readername.setText(reader.getReadername());

tf\_readertype.setText(reader.getReadertype());

tf\_max\_num.setText(String.valueOf(reader.getMax\_num()));

tf\_days\_num.setText(String.valueOf(reader.getDays\_num()));

} else {

JOptionPane.showMessageDialog(null, "读者编号有误，查无此人！");

init(); // 重新初始化各参数并禁止借出按钮

return;

}

// 查询指定读者是否已借过指定图书且未归还

if (IfBorrowBack.findbook(bookid, readerid)) {

JOptionPane.showMessageDialog(null, "该读者已借阅所选图书，且未归还！");

init(); // 重新初始化各参数并禁止借出按钮

return;

}

// 统计读者所借图书数量

int borrowednum = statborrowednum(readerid);

tf\_borrowednum.setText(String.valueOf(borrowednum));

// 如果读者已借图书尚未超出其允许最大借书量，则允许其继续借阅所选图书

if (borrowednum < reader.getMax\_num()) {

tf\_if\_borrow.setText("是");

// 创建一个简单日期格式对象，注意：MM一定要用大写

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

// 创建日期变量，其内容为当前日期

Date currentdate = new Date();

// 将日期按指定格式输出

String borrowdate = sdf.format(currentdate);

tf\_borrow\_date.setText(borrowdate);

borrowBtn.setEnabled(true); // 使借出按钮有效

} else {

JOptionPane.showMessageDialog(null, "该读者借书过多，无法继续借阅！");

init(); // 重新初始化各参数并禁止借出按钮

return;

}

}

// 填写借出图书记录

private void btn\_borrowActionPerformed(ActionEvent e) {

String sql;

String bookid = tf\_bookid.getText();

String readerid = tf\_readerid.getText();

String borrowdate = tf\_borrow\_date.getText();

// 为borrow表增加借书记录

sql = "insert into tb\_borrow (book\_id,reader\_id,"

+ "borrow\_date,if\_back) values('" + bookid + "','" + readerid

+ "','" + borrowdate + "','否')";

DbOp.executeUpdate(sql);

// 将该读者所借图书数量加1

int iborrowednum = Integer.parseInt(tf\_borrowednum.getText()) + 1;

String cborrowednum = String.valueOf(iborrowednum);

tf\_borrowednum.setText(cborrowednum);

// 将图书库存数量减1

int istock = Integer.parseInt(tf\_stock.getText()) - 1;

String cstock = String.valueOf(istock);

// 更新画面中的图书库存数量

tf\_stock.setText(cstock);

// 更新数据库中的图书库存数量

sql = "update tb\_book set stock='" + cstock;

sql = sql + "' where id='" + bookid + "'";

DbOp.executeUpdate(sql);

JOptionPane.showMessageDialog(null, "借书成功！");

init(); // 重新初始化各参数并禁止借出按钮

}

// 统计某个读者当前已借图书且未归还的数量

private int statborrowednum(String readerid) {

int borrowednum = 0;

String reader\_id, if\_back;

// 读取数据库中记录

String sql = "select \* from tb\_borrow";

ResultSet rs = DbOp.executeQuery(sql);

// 执行查询统计操作

try {

while (rs.next()) {

reader\_id = rs.getString("reader\_id");

if\_back = rs.getString("if\_back");

if (reader\_id.equals(readerid) && if\_back.equals("否")) {

borrowednum++;

}

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "数据库统计失败！");

}

return borrowednum;

}

// 初始化各参数项并禁止借出按钮

private void init() {

tf\_bookname.setText("xx"); tf\_author.setText("xx");

tf\_publisher.setText("xx"); tf\_publish\_time.setText("xx");

tf\_price.setText("xx"); tf\_stock.setText("xx");

tf\_readername.setText("xx"); tf\_readertype.setText("xx");

tf\_max\_num.setText("xx"); tf\_days\_num.setText("xx");

tf\_borrowednum.setText("xx"); tf\_if\_borrow.setText("xx");

tf\_borrow\_date.setText("xx");

borrowBtn.setEnabled(false); // 禁止借出按钮

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new Borrow();

}

}

/\* Back.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.\*;

public class Back extends JFrame {

String SepLine = "-------------------------------------------------";

JLabel lbbookid = new JLabel("图书编号");

JLabel lbreaderid = new JLabel("读者编号");

JTextField tf\_bookid = new JTextField();

JTextField tf\_readerid = new JTextField();

JButton queryBtn = new JButton("查询");

JLabel lbbookinfo = new JLabel(SepLine + "图书信息" + SepLine);

JLabel lbbookname = new JLabel("图书名称：");

JLabel tf\_bookname = new JLabel("xx");

JLabel lbauthor = new JLabel("作者：");

JLabel tf\_author = new JLabel("xx");

JLabel lbpublisher = new JLabel("出版社：");

JLabel tf\_publisher = new JLabel("xx");

JLabel lbpublish\_time = new JLabel("出版时间：");

JLabel tf\_publish\_time = new JLabel("xx");

JLabel lbprice = new JLabel("定价："); JLabel tf\_price = new JLabel("xx");

JLabel lbstock = new JLabel("库存数量："); JLabel tf\_stock = new JLabel("xx");

JLabel lbreaderinfo = new JLabel(SepLine + "读者信息" + SepLine);

JLabel lbreadername = new JLabel("读者姓名：");

JLabel tf\_readername = new JLabel("xx");

JLabel lbreadertype = new JLabel("读者类型：");

JLabel tf\_readertype = new JLabel("xx");

JLabel lbmax\_num = new JLabel("最大可借数：");

JLabel tf\_max\_num = new JLabel("xx");

JLabel lbdays\_num = new JLabel("最大可借天数：");

JLabel tf\_days\_num = new JLabel("xx");

JLabel lbbackinfo = new JLabel(SepLine + "还书信息" + SepLine);

JLabel lbback\_date = new JLabel("还书日期：");

JLabel tf\_back\_date = new JLabel("xx");

JButton backBtn = new JButton("还书");

JButton closeBtn = new JButton("关闭");

public Back() {

setLayout(null);

setTitle("还书管理");

setSize(500, 350);

this.setForeground(Color.BLACK); // 设置前景色为黑色

lbbookid.setBounds(30, 40, 50, 20); // 图书编号

tf\_bookid.setBounds(90, 40, 90, 20);

lbreaderid.setBounds(200, 40, 50, 20); // 读者编号

tf\_readerid.setBounds(260, 40, 90, 20);

queryBtn.setBounds(370, 40, 80, 20); // 查询按钮

lbbookinfo.setBounds(30, 70, 440, 20); // 图书信息提示条

lbbookname.setBounds(30, 90, 60, 20); // 图书名称

tf\_bookname.setBounds(90, 90, 200, 20);

lbauthor.setBounds(310, 90, 60, 20); // 作者

tf\_author.setBounds(370, 90, 90, 20);

lbpublisher.setBounds(30, 110, 60, 20); // 出版社

tf\_publisher.setBounds(90, 110, 200, 20);

lbpublish\_time.setBounds(310, 110, 60, 20); // 出版时间

tf\_publish\_time.setBounds(370, 110, 90, 20);

lbprice.setBounds(30, 130, 60, 20); // 定价

tf\_price.setBounds(90, 130, 200, 20);

lbstock.setBounds(310, 130, 60, 20); // 库存数量

tf\_stock.setBounds(370, 130, 90, 20);

lbreaderinfo.setBounds(30, 150, 440, 20); // 读者信息提示条

lbreadername.setBounds(30, 170, 60, 20); // 读者姓名

tf\_readername.setBounds(90, 170, 90, 20);

lbreadertype.setBounds(310, 170, 60, 20); // 读者类型

tf\_readertype.setBounds(370, 170, 90, 20);

lbmax\_num.setBounds(30, 190, 75, 20); // 最大可借数

tf\_max\_num.setBounds(105, 190, 90, 20);

lbdays\_num.setBounds(310, 190, 85, 20); // 最大可借天数

tf\_days\_num.setBounds(395, 190, 70, 20);

lbbackinfo.setBounds(30, 210, 440, 20); // 还书信息提示条

lbback\_date.setBounds(30, 230, 60, 20); // 还书日期

tf\_back\_date.setBounds(90, 230, 100, 20);

backBtn.setBounds(160, 260, 80, 25); // 还书按钮

backBtn.setEnabled(false); // 开始时禁用还书按钮

closeBtn.setBounds(260, 260, 80, 25); // 关闭按钮

queryBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_querywActionPerformed(e);

}

});

backBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_backActionPerformed(e);

}

});

closeBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

setForeground(Color.BLACK); // 设置前景色为黑色

dispose(); // 关闭窗体

}

});

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

setForeground(Color.BLACK); // 设置前景色为黑色

dispose(); // 关闭窗体

}

});

Container c = getContentPane();

c.add(lbbookid); c.add(lbreaderid); c.add(queryBtn);

c.add(lbbookinfo); c.add(lbbookname); c.add(lbauthor);

c.add(lbpublisher); c.add(lbpublish\_time); c.add(lbprice);

c.add(lbstock); c.add(lbreaderinfo); c.add(lbreadername);

c.add(lbreadertype); c.add(lbmax\_num); c.add(lbdays\_num);

c.add(lbbackinfo); c.add(lbback\_date); c.add(backBtn);

c.add(closeBtn);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

c.setBackground(Color.white); // 设置前景色为红色

c.add(tf\_bookid); c.add(tf\_readerid); c.add(tf\_bookname);

c.add(tf\_author); c.add(tf\_publisher); c.add(tf\_publish\_time);

c.add(tf\_price); c.add(tf\_stock); c.add(tf\_readername);

c.add(tf\_readertype); c.add(tf\_max\_num); c.add(tf\_days\_num);

c.add(tf\_back\_date);

}

// 图书和读者查询

private void btn\_querywActionPerformed(ActionEvent e) {

String bookid = tf\_bookid.getText();

String readerid = tf\_readerid.getText();

// 如果图书编号或读者编号两者均为空，或者有一个为空，则返回

if (bookid.equals("") || readerid.equals("")) {

JOptionPane.showMessageDialog(null, "图书编号和读者编号均不能为空！");

init(); // 重新初始化各参数并禁止还书按钮

return;

}

// 按编号查询图书，结果存入book对象中

Book book = BookSelect.SelectBookById(bookid);

// 如果查询到结果，将其显示在各文本框中

if (book != null) {

tf\_bookname.setText(book.getBookname());

tf\_author.setText(book.getAuthor());

tf\_publisher.setText(book.getPublisher());

tf\_publish\_time.setText(book.getPublish\_time().toString());

tf\_price.setText(String.valueOf((book.getPrice())));

tf\_stock.setText(String.valueOf(book.getStock()));

} else {

JOptionPane.showMessageDialog(null, "图书编号有误，查无此书！");

init(); // 重新初始化各参数并禁止还书按钮

return;

}

// 按编号查询读者，结果存入reader对象中

Reader reader = ReaderSelect.selectReaderById(readerid);

// 如果查询到结果，将其显示在各文本框中

if (reader != null) {

tf\_readername.setText(reader.getReadername());

tf\_readertype.setText(reader.getReadertype());

tf\_max\_num.setText(String.valueOf(reader.getMax\_num()));

tf\_days\_num.setText(String.valueOf(reader.getDays\_num()));

} else {

JOptionPane.showMessageDialog(null, "读者编号有误，查无此人！");

init(); // 重新初始化各参数并禁止还书按钮

return;

}

// 查询指定读者是否借阅过指定图书，且未归还

if (!IfBorrowBack.findbook(bookid,readerid)){

JOptionPane.showMessageDialog(null, "该读者没有借过此种图书！");

init(); // 重新初始化各参数并禁止还书按钮

return;

}

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

// 创建日期变量，其内容为当前日期

Date currentdate = new Date();

// 将日期按指定格式输出

String borrowdate = sdf.format(currentdate);

tf\_back\_date.setText(borrowdate);

backBtn.setEnabled(true); // 使还书按钮有效

}

// 填写还书记录

private void btn\_backActionPerformed(ActionEvent e) {

String sql;

String bookid = tf\_bookid.getText();

String readerid = tf\_readerid.getText();

String backdate = tf\_back\_date.getText();

// 更新borrow表记录

sql = "update tb\_borrow set if\_back='是',back\_date='";

sql = sql + backdate + "' where ";

sql = sql + " book\_id='" + bookid + "' and ";

sql = sql + "reader\_id='" + readerid + "' and ";

sql = sql + "if\_back='否'";

DbOp.executeUpdate(sql);

// 将图书库存数量加1

int istock = Integer.parseInt(tf\_stock.getText()) + 1;

String cstock = String.valueOf(istock);

// 更新画面中的图书库存数量

tf\_stock.setText(cstock);

// 更新数据库中的图书库存数量

sql = "update tb\_book set stock='" + cstock;

sql = sql + "' where id='" + bookid + "'";

DbOp.executeUpdate(sql);

JOptionPane.showMessageDialog(null, "还书成功！");

init(); // 重新初始化各参数并禁止还书按钮

}

// 初始化各参数项并禁止还书按钮

private void init() {

tf\_bookname.setText("xx"); tf\_author.setText("xx");

tf\_publisher.setText("xx"); tf\_publish\_time.setText("xx");

tf\_price.setText("xx"); tf\_stock.setText("xx");

tf\_readername.setText("xx"); tf\_readertype.setText("xx");

tf\_max\_num.setText("xx"); tf\_days\_num.setText("xx");

tf\_back\_date.setText("xx");

backBtn.setEnabled(false); // 禁止还书按钮

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new Back();

}

}

/\* BookQuery.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import java.text.\*;

import java.util.\*;

import java.util.Date;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class BookQuery extends JFrame {

JScrollPane scrollPane;

JLabel lbbookname = new JLabel("图书名称");

JLabel lbauthor = new JLabel("作 者");

JLabel lbpublisher = new JLabel("出 版 社");

JLabel lbpublish\_time = new JLabel("出版时间");

JLabel lbnotes = new JLabel("（年-月-日）");

JTextField tf\_bookname = new JTextField("");

JTextField tf\_author = new JTextField("");

JTextField tf\_publisher = new JTextField("");

JTextField tf\_publish\_time = new JTextField("");

JButton queryBtn = new JButton("查询");

JButton closeBtn = new JButton("关闭");

JTable table;// 定义表格table

DefaultTableModel model;// 定义DefaultTableModel类对象model

// 构造方法

public BookQuery() {

setTitle("图书查询"); // 设置窗体标题

setSize(800, 500); // 设置窗体尺寸

setLayout(null); // 取消窗体布局

lbbookname.setBounds(170, 20, 50, 20); // 书名

tf\_bookname.setBounds(230, 20, 160, 20);

lbauthor.setBounds(410, 20, 50, 20); // 作者

tf\_author.setBounds(470, 20, 160, 20);

lbpublisher.setBounds(170, 50, 50, 20); // 出版社

tf\_publisher.setBounds(230, 50, 160, 20);

lbpublish\_time.setBounds(410, 40, 50, 20); // 出版时间

lbnotes.setBounds(400, 60, 70, 20);

tf\_publish\_time.setBounds(470, 50, 160, 20);

queryBtn.setBounds(300, 90, 80, 25); // 查询按钮

queryBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_queryActionPerformed(e);

}

});

closeBtn.setBounds(420, 90, 80, 25);// 关闭按钮

closeBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose(); // 释放当前窗体

}

});

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

dispose(); // 释放当前窗体

}

});

String[] heads = { "图书编号", "图书名称", "图书类别", "作者", "译者",

"出版社", "出版日期", "定价", "库存数量" };

// 创建DefaultTableModel类对象model

model = new DefaultTableModel(heads, 0);

// 创建JTable类对象table，并使用model保存table数据

table = new JTable(model);

// 创建JScrollPane类对象scrollPane，

// 将table添加到scrollPane中

JScrollPane scrollPane = new JScrollPane(table);

scrollPane.setBounds(10, 140, 760, 300);

Container c = getContentPane();

c.add(lbbookname); // 将各组件添加到窗体中

c.add(tf\_bookname); c.add(lbauthor); c.add(tf\_author);

c.add(lbpublisher); c.add(tf\_publisher); c.add(lbpublish\_time);

c.add(lbnotes); c.add(tf\_publish\_time); c.add(queryBtn);

c.add(closeBtn); c.add(scrollPane);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

}

private void btn\_queryActionPerformed(ActionEvent e) {

// 清空表格和查询文本框

int rowCount = model.getRowCount();

for (int i = 1; i <= rowCount; i++) {

model.removeRow(0);

}

try {

String bookname, author, publisher, publishtime;

String sql, sql1, sql2, sql3, sql4, sql5;

String pubyear, pubmonth;

bookname = tf\_bookname.getText();

author = tf\_author.getText();

publisher = tf\_publisher.getText();

publishtime = tf\_publish\_time.getText();

// 创建一条基本的SQL语句，表示选出表中全部记录

sql = "select \* from tb\_book ";

// 如果书名不空，生成sql1字句

if (bookname.equals(""))

sql1 = "";

else

sql1 = " bookname like '" + bookname + "%' ";

// 如果作者不空，生成sql2字句

if (author.equals(""))

sql2 = "";

else {

sql2 = " author like '" + author + "%' ";

if (!bookname.equals("")) // 如果书名不为空

sql2 = " and " + sql2;

}

// 如果出版社不空，生成sql3字句

if (publisher.equals(""))

sql3 = "";

else {

sql3 = "publisher like '" + publisher + "%' ";

// 如果书名和作者有一项不为空

if (!(bookname.equals("") && author.equals("")))

sql3 = " and " + sql3;

}

// 如果出版日期不空，生成sql4字句

if (publishtime.equals("")) {

sql4 = "";

} else {

// 创建一个简单日期格式对象，注意：MM一定要用大写

// 这是用户输入日期的格式：年-月，如2010-7、2009-10等

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM");

// 创建一个Calendar对象

Calendar cal = new GregorianCalendar();

// 将字符串转换为日期

Date pubtime = sdf.parse(tf\_publish\_time.getText());

// 使用给定日期设置cal的时间

cal.setTime(pubtime);

// 获取年

pubyear = String.valueOf(cal.get(Calendar.YEAR));

// 获取月

pubmonth = String.valueOf(cal.get(Calendar.MONTH) + 1);

sql4 = " year(publish\_time)=" + pubyear + " and ";

sql4 = sql4 + "month(publish\_time)=" + pubmonth;

// 如果书名、作者或出版社有一项不为空

if (!(bookname.equals("") && author.equals("") && publisher

.equals("")))

sql4 = " and " + sql4;

}

sql5 = sql1 + sql2 + sql3 + sql4;

// 如果已设置任意一项条件，则修改SQL语句

if (!sql5.equals("")) {

sql = sql + " where " + sql5;

}

// 执行查询

ResultSet rs = DbOp.executeQuery(sql);

while (rs.next()) {

String[] bookInfo = { rs.getString("id"), rs.getString("bookname"),

rs.getString("booktype"), rs.getString("author"),

rs.getString("translator"), rs.getString("publisher"),

rs.getString("publish\_time"), rs.getString("price"),

rs.getString("stock")};

model.addRow(bookInfo);// 将信息添加到model中

}

} catch (ParseException e2) {

JOptionPane.showMessageDialog(null, "出版时间格式错误（年-月-日）！");

} catch (SQLException e1) {

JOptionPane.showMessageDialog(null, "数据库不存在，或存在错误！");

}

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new BookQuery();

}

}

/\* ReaderQuery.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class ReaderQuery extends JFrame {

JLabel lbreadername = new JLabel("读者姓名");

JLabel lbreadertype = new JLabel("读者类别");

JTextField tf\_readername = new JTextField("");

JTextField tf\_readertype = new JTextField("");

JButton queryBtn = new JButton("查询");

JButton closeBtn = new JButton("关闭");

JTable table;// 定义表格table

DefaultTableModel model;// 定义DefaultTableModel类对象model

// 构造方法

public ReaderQuery() {

setTitle("读者查询"); // 设置窗体标题

setSize(600, 500); // 设置窗体尺寸

setLayout(null); // 取消窗体布局

lbreadername.setBounds(70, 20, 50, 20); // 读者姓名

tf\_readername.setBounds(130, 20, 160, 20);

lbreadertype.setBounds(310, 20, 50, 20); // 读者类别

tf\_readertype.setBounds(370, 20, 160, 20);

queryBtn.setBounds(200, 60, 80, 25); // 查询按钮

queryBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btn\_queryActionPerformed(e);

}

});

closeBtn.setBounds(320, 60, 80, 25); // 关闭按钮

closeBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose(); // 释放当前窗体

}

});

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

dispose(); // 释放当前窗体

}

});

String[] heads = { "读者编号", "读者姓名", "读者类型", "读者性别",

"最大可借数", "可借天数" };

// 创建DefaultTableModel类对象model

model = new DefaultTableModel(heads, 0);

// 创建JTable类对象table，并使用model保存table数据

table = new JTable(model);

// 创建JScrollPane类对象scrollPane，将table添加到scrollPane中

JScrollPane scrollPane = new JScrollPane(table);

// 设置JScrollPane的位置和尺寸

scrollPane.setBounds(10, 120, 560, 300);

Container c = getContentPane();

c.add(scrollPane);

c.add(lbreadername); // 将各组件添加到窗体中

c.add(tf\_readername); c.add(lbreadertype);

c.add(tf\_readertype); c.add(queryBtn);

c.add(closeBtn);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

}

private void btn\_queryActionPerformed(ActionEvent e) {

// 清空表格和查询文本框

int rowCount = model.getRowCount();

for (int i = 1; i <= rowCount; i++) {

model.removeRow(0);

}

try {

String readername, readertype;

String sql, sql1, sql2, sql3;

readername = tf\_readername.getText();

readertype = tf\_readertype.getText();

// 创建一条基本的SQL语句，表示选出表中全部记录

sql = "select \* from tb\_reader ";

// 如果读者姓名不空，生成sql1字句

if (readername.equals(""))

sql1 = "";

else

sql1 = " readername like '" + readername + "%' ";

// 如果作者不空，生成sql2字句

if (readertype.equals(""))

sql2 = "";

else {

sql2 = " readertype like '" + readertype + "%' ";

if (!readername.equals("")) // 如果书名不为空

sql2 = " and " + sql2;

}

sql3 = sql1 + sql2;

// 如果已设置任意一项条件，则修改SQL语句

if (!sql3.equals("")) {

sql = sql + " where " + sql3;

}

// 执行查询

ResultSet rs = DbOp.executeQuery(sql);

while (rs.next()) {

String[] readerInfo = { rs.getString("id"), rs.getString("readername"),

rs.getString("readertype"), rs.getString("sex"),

rs.getString("max\_num"), rs.getString("days\_num")};

model.addRow(readerInfo);// 将信息添加到model中

}

} catch (SQLException e1) {

JOptionPane.showMessageDialog(null, "数据库不存在，或存在错误！");

}

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new ReaderQuery();

}

}

/\* UpdatePassword.java \*/

package chapter13.practice;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class UpdatePassword extends JFrame {

JLabel newPassword = new JLabel("请输入新密码");

JLabel confirmPass = new JLabel("再次确认新密码");

JPasswordField text\_1 = new JPasswordField();

JPasswordField text\_2 = new JPasswordField();

JButton confirmBtn = new JButton("确定");

JButton cancelBtn = new JButton("取消");

public UpdatePassword() {

setLayout(null);

setTitle("修改密码");

setSize(300, 170);

newPassword.setBounds(30, 20, 90, 30);

text\_1.setBounds(125, 23, 120, 20);

confirmPass.setBounds(30, 50, 90, 30);

text\_2.setBounds(125, 53, 120, 20);

confirmBtn.setBounds(70, 90, 70, 25); // 确定按钮

confirmBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

confirmButtonPerformed(e);

}

});

cancelBtn.setBounds(160, 90, 70, 25); // 关闭按钮

cancelBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose(); // 释放窗体资源

}

});

/\* 关闭窗口 \*/

this.addWindowListener(new WindowAdapter() {

// 重写windowClosing()方法

public void windowClosing(WindowEvent e) {

dispose(); // 释放窗体资源

}

});

Container c = getContentPane();

c.add(newPassword); c.add(confirmPass); c.add(text\_1);

c.add(text\_2); c.add(confirmBtn); c.add(cancelBtn);

setLocationRelativeTo(null); // 使窗体在屏幕上居中放置

setVisible(true); // 使窗体可见

}

private void confirmButtonPerformed(ActionEvent e) {

String pass1 = String.valueOf(text\_1.getPassword());

String pass2 = String.valueOf(text\_2.getPassword());

// 如果两个密码输入框中有一个为空，则显示错误提示信息并返回

if (pass1.equals("") || pass1.equals("")) {

JOptionPane.showMessageDialog(null, "密码不能为空，请重新输入！");

return;

}

// 如果两个密码输入框中输入的内容不一致，则显示错误提示信息并返回

if (!pass1.equals(pass2)) {

JOptionPane.showMessageDialog(null, "两次输入的密码不一致，请重新输入！");

text\_1.setText("");

text\_2.setText("");

return;

}

String sql = "update tb\_user set password='" + pass1

+ "' where username='" + GlobalVar.login\_user + "'";

int i = DbOp.executeUpdate(sql);

if (i == 1) {

JOptionPane.showMessageDialog(null, "修改密码成功！");

} else

JOptionPane.showMessageDialog(null, "用户数据库有误或不存在，修改密码失败！");

}

// 增加main()方法，主要为了调试程序界面

public static void main(String[] args) {

new UpdatePassword();

}

}

/\* PublicModule.java \*/

package chapter13.practice;

import java.sql.\*;

import javax.swing.JOptionPane;

class DbOp {

// 数据库url路径

private static String url = "jdbc:mysql://127.0.0.1:3306/db\_book";

private static Connection con = null;

// 构造方法。如果数据库未打开，则通过创建连接打开数据库

private DbOp () {

try {

// 如果当前未创建连接，则加载JDBC驱动程序，然后创建连接

if (con == null) {

con = DriverManager.getConnection(url, "root", "123456");

}

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "数据库未能打开！");

}

}

// 执行数据库查询工作。如果出现异常，返回null

public static ResultSet executeQuery(String sql) {

try {

// 如果未创建数据库连接，则创建连接

if (con == null)

new DbOp();

// 返回查询结果

return con.createStatement().executeQuery(sql);

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "数据库不存在，或存在错误！");

return null;

}

}

// 执行数据库更新操作。如果有问题，则返回-1

public static int executeUpdate(String sql) {

try {

// 如果未创建数据库连接，则创建连接

if (con == null)

new DbOp();

// 返回操作结果

return con.createStatement().executeUpdate(sql);

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "数据有误，记录无法正常保存或更新！");

return -1;

}

}

// 关闭数据库

public static void Close() {

try {

// 如果数据库已打开，则关闭之

if (con != null)

con.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "数据库未打开！");

}

}

}

class Book {

private String id; private String bookname; private String booktype;

private String author; private String translator; private String publisher;

private Date publish\_time; private int stock; private float price;

public String getId() { return id; }

public void setId(String id) { this.id = id; }

public String getBookname() {return bookname;}

public void setBookname(String name) { this.bookname = name; }

public String getBooktype() {return booktype; }

public void setBooktype(String type) {this.booktype = type;}

public String getAuthor() {return author; }

public void setAuthor(String author) {this.author = author; }

public String getTranslator() {return translator; }

public void setTranslator(String translator) {this.translator = translator;}

public String getPublisher() {return publisher;}

public void setPublisher(String publisher) {this.publisher = publisher; }

public Date getPublish\_time() {return publish\_time; }

public void setPublish\_time(Date publish\_time) {this.publish\_time = publish\_time;}

public int getStock() {return stock;}

public void setStock(int stock) {this.stock = stock;}

public float getPrice() {return price;}

public void setPrice(float price) {this.price = price;}

}

class Reader {

private String id; private String readername; private String readertype;

private String sex; private int max\_num; private int days\_num;

public String getId() { return id; }

public void setId(String id) { this.id = id; }

public String getReadername() { return readername;}

public void setReadername(String name) {this.readername = name;}

public String getReadertype() {return readertype;}

public void setReadertype(String type) { this.readertype = type;}

public String getSex() {return sex; }

public void setSex(String sex) {this.sex = sex; }

public int getMax\_num() {return max\_num;}

public void setMax\_num(int max\_num) {this.max\_num = max\_num;}

public int getDays\_num() {return days\_num;}

public void setDays\_num(int days\_num) {this.days\_num = days\_num;}

}

class BookSelect {

// 按图书编号查询，查询结果保存在Book类的对象中

public static Book SelectBookById(String id) {

String sql = "select \* from tb\_book where id='" + id + "'";

ResultSet rs = DbOp.executeQuery(sql);

Book book = null;

try {

if (rs.next()) {

book = new Book();

book.setId(rs.getString("id"));

book.setBooktype(rs.getString("booktype"));

book.setBookname(rs.getString("bookname"));

book.setAuthor(rs.getString("author"));

book.setTranslator(rs.getString("translator"));

book.setPublisher(rs.getString("publisher"));

book.setPublish\_time(rs.getDate("publish\_time"));

book.setPrice(rs.getFloat("price"));

book.setStock(rs.getInt("stock"));

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "无法正常读取数据库！");

}

return book;

}

}

class ReaderSelect {

// 按读者编号查询，结果保存在Reader类的对象中

public static Reader selectReaderById(String id) {

String sql = "select \* from tb\_reader where id='" + id + "'";

ResultSet rs = DbOp.executeQuery(sql);

Reader reader = null;

try {

if (rs.next()) {

reader = new Reader();

reader.setId(rs.getString("id"));

reader.setReadername(rs.getString("readername"));

reader.setReadertype(rs.getString("readertype"));

reader.setSex(rs.getString("sex"));

reader.setMax\_num(rs.getInt("max\_num"));

reader.setDays\_num(rs.getInt("days\_num"));

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "无法正常读取数据库！");

}

return reader;

}

}

class IfBorrowBack {

// 查指定读者是否借过指定图书且未归还

public static boolean findbook(String bookid, String readerid) {

String sql;

sql = "select \* from tb\_borrow where book\_id='";

sql = sql + bookid + "' and reader\_id='" + readerid + "' and ";

sql = sql + "if\_back='否'";

ResultSet rs = DbOp.executeQuery(sql);

try {

// 如果指定读者借阅了指定图书，且未归还，返回true，否则返回false

if (rs.next())

return true;

else

return false;

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "数据库查询失败！");

}

return true;

}

}

class GlobalVar {

// 该变量用来保存登录用户名

public static String login\_user;

}