# 认识Java

1．选择题

（1）D （2）B （3）D （4）D （5）A （6）C （7）C

2．简答题

（1）Java语言有3个版本分别为Java SE、Java EE和Java ME。

（2）Java语言具有简单性、面向对象、分布性、解释性、可移植性、安全性、健壮性、多线程、高性能和动态等特点。

（3）Java程序的开发过程包括需求分析、算法设计、编写程序、编译程序、运行程序和编写程序文档。

（4）JVM是运行Java程序的软件环境，Java解释器是JVM的一部分。利用JVM可以把字节码程序和具体的硬件平台以及操作系统环境分隔开来，只要在不同的平台上安装针对特定具体平台的JVM，Java程序就可以运行，从而实现真正二进制代码级的跨平台移植。

3．编程题

（1）

/\* Program1.java \*/

public class Program1 {

public static void main(String[] args) {

System.out.println("姓名:\*\*\*");

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

System.out.println("电话:152\*\*\*\*\*\*\*\*");

}

}

（2）

/\* Program2.java \*/

public class Program2 {

public static void main(String[] args) {

System.out.println("《登鹳雀楼》");

System.out.println(" ——王之涣");

System.out.println("白日依山尽，");

System.out.println("黄河入海流。");

System.out.println("欲穷千里目，");

System.out.println("更上一层楼。");

}

}

# Java语言基础

1．选择题

（1）C （2）D （3）C （4）C （5）A （6）B （7）D

2．填空题

（1）字母、数字、下划线、美元符号，数字

（2）byte、short、int、long、float、double、char和boolean

（3）true和false

（4）1.5 6.7

（5）false

3．编程题

（1）

/\* Program1.java \*/

import java.util.Scanner;

public class Program1 {

public static void main(String[] args) {

System.out.println("请输入一个三位整数：");

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

int num = scan.nextInt();//扫描为int型整数

int a = num / 100; //计算百位数

int b = (num % 100) / 10; //计算十位数

int c = num % 10; //计算个位数

int num1 = c \* 100 + b \* 10 + a; //计算逆序后整数

System.out.println("逆序输出的三位整数为:" + num1);

scan.close();

}

}

（2）

/\* Program2.java \*/

public class Program2 {

public static void main(String[] args) {

//定义变量a、b、c、d、e、f

double a = 25.6;

int b = 3;

int e = 28;

int c = 8;

float d = 13.4f;

int f = 65;

double x = (e \* d - b \* f)/(a \* d - b \* c); //计算x

double y = (a \* f - e \* c)/(a \* d - b \* c); //计算y

System.out.println("该二元一次方程组中的x = " + x);

System.out.println("该二元一次方程组中的y = " + y);

}

}

# 第3章 流程控制

1．选择题

（1）A （2）A （3）D （4）B （5）C （6）B

2．填空题

（1）24

（2）20 10 10

（3）① (a > b) ② b = t;

（4）15

3．编程题

（1）

/\* Program1.java \*/

import java.util.Scanner;

public class Program1 {

public static void main(String[] args) {

//创建Scanner对象，接收从控制台输入的信息

Scanner scan = new Scanner(System.in);

System.out.print("请输入第一条边长："); //输出提示

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

System.out.print("请输入第二条边长："); //输出提示

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

System.out.print("请输入第三条边长："); //输出提示

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

//如果任意两条边的和大于第三条边

if ((a + b) > c & (a + c) > b & (b + c) > a) {

double S = (a + b + c) / 2;//定义S并计算该值

//定义Area并计算该值

double Area = Math.sqrt(S \* (S -a) \* (S - b) \* (S - c));

//输出Area

System.out.println("三角形的面积为" + Area);

}

else //如果任意两条边的和不大于第三条边，输出不能组成三角形提示

System.out.println("边长为" + a +"、" + b + "、" + c

+ "的三条边不能组成三角形");

scan.close(); //关闭扫描器

}

}

（2）

/\* Program2.java \*/

public class Program2 {

public static void main(String[] args) {

int k = 2, i, count = 0; //定义变量k，i，count

while (k <= 100) { //外循环，当k>100结束循环

i = 2; //i赋初值

while (i <= k-1) { //内循环，循环检测k能否被i整除

if (k % i == 0) //k能被i整除时

break; //退出内循环

i ++; //i自增1

}

if (i == k) { //如果i等于k，k为素数

System.out.print(k + " "); //输出该素数k

count ++; //count自增1

if (count % 5 ==0) //如果count除5取余为0

System.out.print("\n"); //输出换行符

}

k ++; //k自增1

}

}

}

# 第4章 Java面向对象编程基础

1．选择题

（1）D （2）C （3）A （4）B （5）A （6）A （7）D （8）D

2．填空题

（1）封装、继承和多态

（2）package

（3）private、friendly、protected和public

（4）Point a = new Point();

（5）static void move();

3．编程题

（1）

/\* Box.java \*/

package chapter4.program1;

public class Box {

private int length; // 定义长度

private int width; // 定义宽度

private int height; // 定义高度

Box(int length, int width, int height) { // 构造方法，初始化成员变量

this.length = length;

this.width = width;

this.height = height;

}

public void showBox() {

System.out.println("盒子的长是：" + length);

System.out.println("盒子的宽是：" + width);

System.out.println("盒子的高是：" + height);

}

public static void main(String[] args) {

Box box = new Box(1, 5, 7); // 创建对象

box.showBox(); // 调用showBox()方法

}

}

（2）

/\* Compute.java \*/

package chapter4.program2;

import java.util.Scanner;

public class Compute {

static int x, y; // 定义两个数

Compute(int x, int y) { // 构造方法，初始化成员变量

Compute.x = x;

Compute.y = y;

}

void add() { // 定义加法方法

System.out.println("两数的和为：" + (x + y));

}

void sub() { // 定义减法方法

System.out.println("两数的差为：" + (x - y));

}

void Multiplication() { // 定义乘法方法

System.out.println("两数的积为：" + (x \* y));

}

void division() { // 定义除法方法

System.out.println("两数的商为：" + (x / y));

}

public static void main(String[] args) {

System.out.print("请输入两个数：（注意：中间以空格分隔）");

Scanner scan = new Scanner(System.in);

Compute data = new Compute(scan.nextInt(), scan.nextInt()); // 创建对象

data.add();

data.sub();

data.Multiplication();

data.division();

scan.close();

}

}

（3）

/\* SalesClerk.java \*/

package chapter4.program3;

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

public class SalesClerk {

public static void main(String[] args) {

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

System.out.print("请选择图书类别（Java/Python/C++）：");// 输出提示

String choice = scan.nextLine(); // 扫描为int型变量并赋给choice

SalesClerk clerk = new SalesClerk(); // 创建SalesClerk类对象clerk

// 调用recommend()方法

System.out.println("店员推荐：" + clerk.recommend(choice));

scan.close();

}

private String recommend(String choice) { // 定义recommend()方法

String bookName = null; // 定义变量bookName

switch (choice) {

case "Java": // 如果choice等于“Java”

bookName = "《Java程序设计教程（第2版）》";

break;

case "Python": // 如果choice等于“Python”

bookName = "《Python编程案例教程》";

break;

case "C++": // 如果choice等于“C++”

bookName = "《C++程序设计实例与操作》";

break;

default:

break;

}

return bookName; // 返回bookName

}

}

# 第5章 Java面向对象高级特性

1．选择题

（1）D （2）D （3）D （4）C （5）D （6）B

2．填空题

（1）interface

（2）extends

（3）单 多

（4）DateFormat df = new SimpleDateFormat("hh 时 mm 分ss 秒");

（5）Math.pow(a, b);

3．编程题

（1）

/\* Person.java \*/

package chapter5.program1;

public class Person { // 定义Person类

String name; // 定义姓名name

char sex; // 定义性别sex

int age; // 定义年龄age

Person(String name, char sex, int age) { // 定义有参构造方法，初始化变量

this.name = name;

this.sex = sex;

this.age =age;

}

public String toSting() { // 返回姓名、性别和年龄

return "姓名：" + name + " 性别：" + sex + " 年龄：" + age;

}

}

/\* Student.java \*/

package chapter5.program1;

public class Student extends Person { // 定义Student类，继承Person类

String no; // 定义学号no

double scoreEn; // 定义英语分数scoreEn

double scoreMath; // 定义数学分数scoreMath

double scoreCh; // 定义语文分数scoreCh

Student(String name, char sex, int age, String no, double scoreEn, double scoreMath, double scoreCh) {

super(name, sex, age); // 调用父类构造方法

this.no = no;

this.scoreEn = scoreEn;

this.scoreMath = scoreMath;

this.scoreCh = scoreCh;

}

public double aver() { // 计算平均分并返回

return (scoreEn + scoreMath + scoreCh) / 3;

}

public double max() { // 计算最高分并返回

double max;

max = Math.max(scoreEn, scoreMath);

max = Math.max(max, scoreCh);

return max;

}

public double min() { // 计算最低分并返回

double min;

min = Math.min(scoreEn, scoreMath);

min = Math.min(min, scoreCh);

return min;

}

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

// 重写toSting()方法，调用父类toSting()方法

// 返回姓名、性别、年龄、学号、平均分、最高分和最低分

return super.toSting() + " 学号：" + no + " 平均分：" + aver() + " 最高分：" + max() + " 最低分：" + min();

}

}

/\* StudentTest.java \*/

package chapter5.program1;

public class StudentTest {

public static void main(String[] args) {

// 创建Person对象，并通过Student类实例化对象

Person p = new Student("凌一", '女', 20, "20190327", 98, 89, 92);

System.out.println(p.toSting()); // 调用toSting()方法，输出学生信息

}

}

（2）

/\* Animal.java \*/

package chapter5.program2;

public interface Animal { // 定义Animal接口

void eat(Food food); // 定义eat()方法

}

class Cat implements Animal { // 定义Cat类实现Animal接口

@Override

public void eat(Food food) { // 重写eat()方法

// 调用getFood()方法，并输出猫吃什么

System.out.println("猫吃" + food.getFood());

}

}

class Dog implements Animal { // 定义Dog类实现Animal接口

@Override

public void eat(Food food) { // 重写eat()方法

// 调用getFood()方法，并输出狗吃什么

System.out.println("狗吃" + food.getFood());

}

}

/\* Food.java \*/

package chapter5.program2;

public abstract class Food { // 定义Food抽象类

String name; // 定义变量name

abstract String getFood(); // 定义抽象方法getFood()

abstract void setFood(String name); // 定义抽象方法setFood()

}

class Fish extends Food { // 定义Fish类继承Food类

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

System.out.println("饲养员给猫喂" + name); // 输出饲养员给猫喂什么

return name; // 返回name

}

public void setFood(String name) { // 重写setFood()方法

this.name = name; // 给name赋值

}

}

class Bone extends Food { // 定义Bone类继承Food类

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

System.out.println("饲养员给狗喂" + name); // 输出饲养员给狗喂什么

return name; // 返回name

}

public void setFood(String name) { // 重写setFood()方法

this.name = name; // 给name赋值

}

}

/\* Feeder.java \*/

package chapter5.program2;

public class Feeder { // 定义Feeder类

public static void main(String[] args) {

Food fish = new Fish(); // 声明Food类对象fish，并使用Fish类对象初始化

fish.setFood("小鱼干");// 调用setFood()方法

Animal cat = new Cat();// 声明Animal类对象cat，并使用Cat类对象初始化

cat.eat(fish); // 调用eat()方法

Food bone = new Bone();// 声明Food类对象bone，并使用Bone类对象初始化

bone.setFood("大骨头");// 调用setFood()方法

Animal dog = new Dog();// 声明Animal类对象dog，并使用Dog类对象初始化

dog.eat(bone); // 调用eat()方法

}

}

# 第6章 数组和字符串

1．选择题

（1）B （2）B （3）B （4）D （5）B （6）D

2．填空题

（1）100 43 453

（2）25

（3）2 5 6 7 8 10 13

（4）JAVA STRING

java string

2

Java Strgni

3．编程题

（1）

/\* CheckString.java \*/

package chapter6.program1;

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

public class CheckString {

public static void main(String[] args) {

String checkStr = null;

System.out.print("请输入要检测的字符串：");

Scanner scan = new Scanner(System.in);

checkStr = scan.nextLine(); // 存储用户输入的字符串

if (isPaildrome(checkStr)) { // 判断输入的字符串是否为回文串

System.out.println(checkStr + "是回文串。");

}

else {

System.out.println(checkStr + "不是回文串。");

}

scan.close();

}

private static boolean isPaildrome(String check) {

int low = 0; // 定义首字符的索引

int high = check.length() - 1; // 定义尾子符的索引

while (low < high) {

// 检测首尾两个字符是否相等

if (check.charAt(low) != check.charAt(high))

return false;

low++;

high--;}

return true;

}

}

（2）

/\* Sudoku.java\*/

package chapter6.program2;

public class Sudoku {

public static void main(String[] args) {

// 声明int类型的数组并分配内存（三阶方阵）

int arr[][] = new int[3][3];

// 确定数字“1”的位置

int a = 2;// 第3行的下标

int b = 3 / 2;// 第2列的下标

for (int i = 1; i <= 9; i++) {// 给数组赋值

arr[a++][b++] = i;// 避免数组下标越界

if (i % 3 == 0) {// 如果i是3的倍数

a = a - 2;

b = b - 1;

} else {// 如果i不是3的倍数

a = a % 3;

b = b % 3;

}

}

System.out.println("九宫格：");

// 遍历数组

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

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

System.out.print(arr[i][j] + " ");// 输出数组中的数据

}

System.out.println();// 换行

}

}

}

# 第7章 泛型和集合

1．选择题

（1）D （2）C （3）D （4）C

2．填空题

（1）类型

（2）键-值对 键 值

（3）size()

（4）ArrayList类 LinkedList类

（5）keySet() Set

3．编程题

（1）

/\* StringProcess.java \*/

package chapter7.program1;

import java.util.\*;

public class StringProcess {

public static void main(String[] args) {

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

System.out.println("输入字符串："); // 输出提示

String s = scan.nextLine(); // 扫描数据为String型，并赋给s

Set<Character> setChar = new TreeSet<>(); // 创建字符型的Set对象setChar

for(int i = 0; i < s.length(); i++) { // 循环

setChar.add(s.charAt(i)); // 将s按字符添加到setChar中

}

Iterator<Character> it = setChar.iterator(); // 生成setChar的迭代器

while(it.hasNext()) { // 遍历迭代器

System.out.print(it.next()); // 输出setChar中的每个元素

}

scan.close(); // 关闭扫描器

}

}

（2）

/\* Course.java \*/

package chapter7.program2;

import java.util.\*;

public class Course {

public static void main(String[] args) {

// 创建String类型的HashMap对象mapCourse

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

// 调用put()方法将员工信息键值对添加到mapEmployee中

mapCourse.put("吕红", "数据库");

mapCourse.put("周婷", "线性代数");

mapCourse.put("肖扬", "Java");

mapCourse.put("李娟", "数据库");

mapCourse.put("丁锦", "英语");

mapCourse.put("周玲玲", "Java");

mapCourse.put("赵红", "Java");

mapCourse.put("周玲玲", "Python");

System.out.println("\*\*\*\*\*\*\*\*课程安排信息\*\*\*\*\*\*\*\*");

// 调用keySet()获取键的集合，并生成对应的迭代器

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

while (it.hasNext()) { // 遍历迭代器

String key = it.next(); // 获取键

// 调用get()方法获取键对应的值

String val = mapCourse.get(key);

// 输出课程信息

System.out.println("教师：" + key + "，课程:" + val);

}

System.out.println("\*\*\*\*\*\*所有讲授Java的教师\*\*\*\*\*\*");

// 调用keySet()获取键的集合，并生成对应的迭代器

it = mapCourse.keySet().iterator();

while (it.hasNext()) { // 遍历迭代器

String key = it.next(); // 获取键

// 调用get()方法获取键对应的值

String val = mapCourse.get(key);

if (val.equals("Java")) // 如果val为Java

System.out.print(key + " "); // 输出所有讲授Java的教师

}

}

}

# 第8章 异常和断言

1．选择题

（1）B （2）B （3）C （4）D （5）B

2．填空题

（1）ArrayIndexOutOfBoundsException

（2）assert

（3）catch

（4）throw

（5）throws

3．编程题

（1）

/\* Loan.java \*/

package chapter8.program1;

public class Loan {

double total; // 定义贷款总额total

double rate; // 定义贷款利率rate

int year; // 定义贷款年数year

Loan(double total, double rate, int year) { // 定义有参构造方法

if (total <= 0 || rate <= 0 || year <= 0) { // 如果total、rate和year有一个小于0

// 抛出RuntimeException异常

throw new RuntimeException ("贷款总额、利率和年限必须大于0");

}

else { // 如果total、rate和year都大于0，将参数赋给它们

this.total = total;

this.rate = rate;

this.year = year;

}

}

public static void main(String[] args) {

try {

Loan loan = new Loan(2025.6, 0, 10); // 创建Loan对象loan

// 输出total、rate和year

System.out.println("贷款总额：" + loan.total + "\n"

+ "贷款利率：" + loan.rate + "\n"

+ "贷款年限：" + loan.year);

} catch (RuntimeException e){// 捕获RuntimeException异常

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

}

}

}

（2）

/\* ElectricityBill.java \*/

package chapter8.program2;

import java.util.InputMismatchException;

import java.util.Scanner;

class ElectNotNegativeException extends Exception { // 定义用电度数小于等于零时异常

ElectNotNegativeException() {

}

}

public class ElectricityBill {

double bill; // 定义电费bill

double computeBill(int count) throws ElectNotNegativeException {

if (count <= 0) { // 如果count小于0，抛出自定义异常

throw new ElectNotNegativeException();

}

else if (count <= 240) { // 计算count小于等于240的bill

bill = count \* 0.50;

}

else if (count <= 400) { // 计算count小于等于400的bill

bill = 240 \* 0.50 + (count - 240) \* 0.53;

}

else { // 计算count大于400的bill

bill = 240 \* 0.50 + 160 \* 0.53 + (count - 400) \* 0.78;

}

return bill;

}

public static void main(String[] args) {

System.out.println("请输入用电度数：");

Scanner scan = new Scanner(System.in);

int count;

while (true) { // 循环

try {

count = scan.nextInt(); // 扫描数据为int型，并赋给count

ElectricityBill ef = new ElectricityBill();// 创建ElectricityBill对象ef

ef.computeBill(count); // 调用computeBill()方法

System.out.println("本月电费和用电量的比值为：" + (ef.bill / count));

break; // 退出循环

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

System.out.println("你输入的用电数不合法，请重新输入:");

scan.nextLine();

} catch (ElectNotNegativeException ee) { // 捕获自定义异常

System.out.println("你输入的用电数不能小于等于0，请重新输入:");

scan.nextLine();

}

}

scan.close(); // 关闭扫描器

}

}

# 第9章 I/O流和文件操作

1．选择题

（1）C （2）A （3）A （4）A （5）A （6）B

2．填空题

（1）输入流、输出流、字节流、字符流

（2）exists()、getAbsolutePath()、creatNewFile()、delete()

（3）BufferedOutputStream、BufferedReader

（4）RandomAccessFile

（5）java.io

3．编程题

（1）

/\* FileCopy.java \*/

package chapter9.program1;

import java.io.\*;

import java.util.Scanner;

public class FileCopy {

public static void copyFile(String sourceFileName, String targetFileName)

throws IOException {

File sourceFile = new File(sourceFileName); // 创建源文件

File targetFile = new File(targetFileName); // 创建目标文件

// 验证源文件是否存在

if (!sourceFile.exists()) {

System.out.println("文件复制失败！源文件" +

sourceFile.getName() + "不存在");

return;

}

// 验证目标文件是否为目录

if (targetFile.isDirectory()) {

// 如果为目录，则修改目标文件的路径和名称

targetFile = new File(targetFile, sourceFile.getName());

}

// 如果目标文件存在

if (targetFile.exists()) {

// 询问用户是否覆盖目标文件

System.out.println("文件进行复制，是否覆盖现有文件" +

targetFile.getName()+ "?(Y/N):");

// 以标准输入创建字符缓冲流

BufferedReader in = new BufferedReader(new InputStreamReader(

System.in));

String override = in.readLine(); // 存储用户的输入值

// 取消复制

if (!override.equalsIgnoreCase("Y")) {

System.out.println("操作已取消!");

return;

}

}

/\* 复制文件 \*/

FileInputStream in = null;

FileOutputStream out = null;

in = new FileInputStream(sourceFile); // 创建源文件输入流

out = new FileOutputStream(targetFile); // 创建目标文件输出流

byte[] buffer = new byte[8]; // 创建缓冲区

int num;

/\* 循环读取源文件数据并写入目标文件 \*/

while ((num = in.read(buffer)) != -1)

out.write(buffer, 0, num);

System.out.println("文件复制成功！");

System.out.println("源文件：" + sourceFile.getPath());

System.out.println("目标文件：" + targetFile.getPath());

// 关闭源文件和目标文件

in.close();

out.close();

}

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

String SrcFileName, DesFileName;

Scanner scan = new Scanner(System.in);

System.out.println("请输入源文件名： ");

SrcFileName = scan.nextLine();

System.out.println("请输入目标文件名： ");

DesFileName = scan.nextLine();

// 调用复制文件方法copyFile

FileCopy.copyFile(SrcFileName, DesFileName);

scan.close();

}

}

（2）

/\* ToUpper.java \*/

package chapter9.program2;

import java.io.\*;

import java.util.Scanner;

public class ToUpper{

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

System.out.println("请输入字符串：");

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

String line = scan.nextLine(); // 从键盘输入字符串

line = line.toUpperCase(); //将输入的字符串转换为大写字母

FileWriter out = new FileWriter("a.txt"); // 打开文件，创建字符输出流out

out.write(line); // 将line写入文件中

out.close(); // 关闭out

scan.close(); //关闭scan

}

}

（3）

/\* OddEven.java \*/

package chapter9.program3;

import java.io.\*;

public class OddEven {

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

File oddFile = new File("OddNnumber.txt"); // 创建保存奇数的文件oddFile

File evenfile = new File("EvenNnumber.txt"); // 创建保存偶数的文件evenfile

BufferedOutputStream odd\_bufout = null; // 声明缓冲字节输出流odd\_bufout

FileOutputStream odd\_dataout = null; // 声明文件输出流odd\_dataout

BufferedOutputStream even\_bufout = null; // 声明缓冲字节输出流even\_bufout

FileOutputStream even\_dataout = null; // 声明文件输出流even\_dataout

try {

// 如果oddFile不存在则创建

if (!oddFile.exists()) {

oddFile.createNewFile();

}

// 如果evenfile不存在则创建

if (!evenfile.exists()) {

evenfile.createNewFile();

}

// 打开oddFile，赋值给odd\_dataout，并基于odd\_dataout初始化odd\_bufout

odd\_dataout = new FileOutputStream(oddFile);

odd\_bufout = new BufferedOutputStream(odd\_dataout);

// 打开evenfile，赋值给even\_dataout，并基于even\_dataout初始化even\_bufout

even\_dataout = new FileOutputStream(evenfile);

even\_bufout = new BufferedOutputStream(even\_dataout);

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

if (i % 2 == 0) { // 如果i为偶数，将i写入even\_bufout

String even = Integer.valueOf(i).toString() + "\t";

even\_bufout.write(even.getBytes());

}

else { // 如果i为偶数，将i写入odd\_bufout

String odd = Integer.valueOf(i).toString() + "\t";

odd\_bufout.write(odd.getBytes());

}

}

} catch (IOException e) {

System.out.println(e);

} finally {

odd\_bufout.close();

odd\_dataout.close();

even\_bufout.close();

even\_dataout.close();

}

}

}

# 第10章 多线程

1．选择题

（1）D （2）D （3）D （4）D （5）B

2．填空题

（1）创建、就绪、运行、阻塞和死亡

（2）setPriority()

（3）继承Thread类和实现Runnable接口

（4）yield()

3．编程题

（1）

/\* NumberAlphabet.java \*/

package chapter10.program1;

public class NumberAlphabet {

public static void main(String[] args) {

NumberThread n = new NumberThread(); // 创建NumberThread类对象n

Thread t1 = new Thread(n); // 基于n创建Thread类对象t1

AlphabetThread a = new AlphabetThread(); // 创建AlphabetThread类对象a

Thread t2 = new Thread(a); // 基于a创建Thread类对象t2

t1.start(); // 启动线程t1

t2.start(); // 启动线程t2

}

}

// 定义NumberThread类实现 Runnable接口，用于数字的输出

class NumberThread implements Runnable{

public void run(){

for (int i = 0; i <= 10; i++){

for(int j = 0; j < 10; j++) // 每10个数换行处理

System.out.print((int)(Math.random()\*10) + " ");

System.out.println();

}

}

}

// 定义AlphabetThread类实现 Runnable接口，用于字母的输出

class AlphabetThread implements Runnable{

public void run(){

for (int i = 0;i <= 10; i++){

for (char c = 'a'; c <= 'z'; c++) // 每行26个字母

System.out.print(c);

System.out.println();

}

}

}

（2）

/\* WriteTest.java \*/

package chapter10.program2;

public class WriteTest {

public static void main(String[] args) {

LeftHand left = new LeftHand(); // 创建线程left

RightHand right = new RightHand(); // 创建线程right

left.start(); // 启动left

right.start(); // 启动right

}

}

// 左手线程类LeftHand

class LeftHand extends Thread {

public void run() {

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

System.out.print("A");

try {

sleep(500); // left线程休眠0.5 s

} catch (InterruptedException e) {

}

}

}

}

// 右手线程类RightHand

class RightHand extends Thread {

public void run() {

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

System.out.print("B");

try {

sleep(300); // right线程休眠0.3 s

} catch (InterruptedException e) {

}

}

}

}

（3）

/\* BreakfastThread.java \*/

package chapter10.program3;

public class BreakfastThread implements Runnable {

static Thread MrZhang, MrLi, MrWang;

// 定义5元钱、10元钱与20元钱的个数

int sumFive = 3, sumTwenty = 0, sumTen = 0;

public synchronized void sellBreakfast(int money) {

if (money == 5) {

sumFive = sumFive + 1;

System.out.println("李先生，您给的钱数正好。");

} else if (money == 10) {

while (sumFive < 1) {

try {

wait(); // 如果5元的个数少于2张，则线程等待

} catch (InterruptedException e) {

}

}

sumFive = sumFive - 1;

sumTen = sumTen + 1;

System.out.println("王先生，您给我10元，找您5元。");

} else if (money == 20) {

while (sumFive < 3) {

try {

wait(); // 如果5元的个数少于3张，则线程等待

} catch (InterruptedException e) {

}

}

sumFive = sumFive - 3;

sumTwenty = sumTwenty + 1;

System.out.println("张先生，您给我20元，找您15元。");

}

notifyAll(); // 通知等待的线程

}

public void run() {

if (Thread.currentThread() == MrZhang) { // 判断当前的线程

sellBreakfast(20); // 调用买票的方法

} else if (Thread.currentThread() == MrLi) {

sellBreakfast(5);

} else if (Thread.currentThread() == MrWang) {

sellBreakfast(10);

}

}

public static void main(String[] args) {

BreakfastThread t = new BreakfastThread();

MrZhang = new Thread(t);

MrWang = new Thread(t);

MrLi = new Thread(t);

MrZhang.start(); // 启动张先生线程

MrWang.start(); // 启动王先生线程

MrLi.start(); // 启动李先生线程

}

}

# 第11章 GUI编程

1．选择题

（1）B （2）D （3）D （4）D （5）B （6）B

2．填空题

（1）AWT Swing

（2）FlowLayout

（3）add() setLayout()

3．编程题

（1）

/\* Questionnaire.java \*/

package chapter11.program1;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import javax.swing.\*;

public class Questionnaire extends JFrame implements ActionListener {

JTextField name = new JTextField(30); // 企业名称，宽度为30

// 员工数量，初始值为50，文本域宽度为4

JTextField Count = new JTextField("50", 4);

JComboBox<String> industry = new JComboBox<String>(); // 从事行业

JTextField turnover = new JTextField(30); // 年营业额

JTextField funds = new JTextField(30); // 注册资金

JTextField margin = new JTextField(30); // 利润率

JButton btnSubmit = new JButton("提交"); // 提交按钮

JButton btnCancel = new JButton("取消"); // 取消按钮

JButton btnQuit = new JButton("退出"); // 退出按钮

// 创建标签

JLabel lName = new JLabel("企业名称");

JLabel lCount = new JLabel("员工数量");

JLabel lIndustry = new JLabel("从事行业");

JLabel lTurnover = new JLabel("年营业额");

JLabel lFunds = new JLabel("注册资金");

JLabel lMargin = new JLabel("利润率");

public Questionnaire(String title) {

super(title); // 调用父类构造方法

this.setSize(400, 400); // 设置窗体的尺寸

this.setLayout(null); // 取消窗体的布局管理器

lName.setBounds(30, 50, 60, 20); // 企业名称标签

name.setBounds(90, 50, 150, 20); // 企业名称文本域

lCount.setBounds(30, 100, 60, 20); // 员工数量标签

Count.setBounds(90, 100, 80, 20); // 员工数量文本域

lIndustry.setBounds(210, 100, 60, 20); // 从事行业标签

industry.setBounds(270, 100, 80, 20); // 从事行业

lTurnover.setBounds(30, 150, 60, 20); // 年营业额标签

turnover.setBounds(90, 150, 80, 20); // 年营业额

lFunds.setBounds(210, 150, 60, 20); // 注册资金标签

funds.setBounds(270, 150, 80, 20); // 注册资金

lMargin.setBounds(30, 200, 60, 20); // 利润率标签

margin.setBounds(90, 200, 80, 20); // 利润率

btnSubmit.setBounds(50, 250, 80, 20); // 确认

btnCancel.setBounds(160, 250, 80, 20); // 取消

btnQuit.setBounds(260, 250, 80, 20); // 退出

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

setVisible(true); // 设置窗体可见

btnSubmit.addActionListener(this); // 为三个按钮注册事件侦听器

btnCancel.addActionListener(this);

btnQuit.addActionListener(this);

// 设置选项框内容

industry.addItem("机构组织");

industry.addItem("信息产业");

industry.addItem("医药卫生");

industry.addItem("机械机电");

Container c = getContentPane();

// 将组件添加到容器中

c.add(lName);

c.add(name);

c.add(lCount);

c.add(Count);

c.add(lIndustry);

c.add(industry);

c.add(lTurnover);

c.add(turnover);

c.add(lFunds);

c.add(funds);

c.add(lMargin);

c.add(margin);

c.add(btnSubmit);

c.add(btnCancel);

c.add(btnQuit);

Count.addActionListener(this); // 为员工数量文本域注册焦点事件侦听器

funds.addActionListener(this); // 为注册基金文本域注册焦点事件侦听器

turnover.addActionListener(this); // 为年企业额文本域注册事件侦听器

margin.addActionListener(this); // 为利润率文本域注册焦点事件侦听器

}

// 重写ActionEvent事件处理方法

public void actionPerformed(ActionEvent e) {

Object ob = e.getSource(); // 获取事件对象

if (ob == btnQuit) { // 单击退出按钮

System.exit(0); // 退出系统

} else if (ob == btnSubmit) { // 单击确认按钮

try {

// 判断数据类型

Integer.parseInt(Count.getText());

Integer.parseInt(funds.getText());

Float.parseFloat(turnover.getText());

Float.parseFloat(margin.getText());

// 将企业信息保存到文本

FileWriter fw = null;

BufferedWriter bw = null;

String filename = "企业信息.txt";

fw = new FileWriter(filename, true);

bw = new BufferedWriter(fw);

bw.write("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

bw.newLine();

bw.write("企业名称：" + name.getText());

bw.newLine();

bw.write("员工数量：" + Count.getText());

bw.newLine();

bw.write("从事行业：" + industry.getSelectedItem());

bw.newLine();

bw.write("年营业额：" + turnover.getText());

bw.newLine();

bw.write("注册资金：" + funds.getText());

bw.newLine();

bw.write("利润率：" + margin.getText());

bw.newLine();

bw.close();

fw.close();

JOptionPane.showMessageDialog(null, "数据保存成功！",

"提示", JOptionPane.INFORMATION\_MESSAGE);

} catch(IOException e1) {

e1.printStackTrace();

} catch (Exception e2) {

JOptionPane.showMessageDialog(null, "员工数量和注册基金为整数,"

+ "年营业额和利润率为浮点数","错误提示",

JOptionPane.ERROR\_MESSAGE);

Count.setText("50");

funds.setText("");

turnover.setText("");

margin.setText("");

}

} else if (ob == btnCancel) { // 单击取消按钮

// 清空填写信息

name.setText("");

Count.setText("50");

turnover.setText("");

funds.setText("");

margin.setText("");

}

}

public static void main(String[] args) {

new Questionnaire("企业信息调查表");

}

}

（2）

/\* TextEditor.java \*/

package chapter11.program2;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import javax.swing.\*;

public class TextEditor extends Frame implements ActionListener {

MenuBar mainmenubar = new MenuBar(); // 声明菜单栏

Menu file; // 声明主菜单项

MenuItem nw; // 声明各子菜单项

MenuItem op;

MenuItem cl;

MenuItem sf;

MenuItem ex;

Menu edit;

MenuItem copy;

MenuItem cut;

MenuItem paste;

JTextArea tx; // 声明文本区对象

public TextEditor(String title) {

super(title); // 调用父类构造方法

setSize(400, 400); // 设置窗体大小

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

menuinit(); // 调用构建与处理菜单方法

tx = new JTextArea(); // 创建文本区对象

this.add(tx); // 将文本区对象放入窗体

setVisible(true); // 使窗体可见

}

// 定义菜单构建与处理方法

void menuinit() {

mainmenubar = new MenuBar(); // 创建主菜单栏

file = new Menu("文件"); // 创建主菜单项

nw = new MenuItem("新建文件"); // 创建各子菜单项

op = new MenuItem("打开文件");

cl = new MenuItem("关闭文件");

sf = new MenuItem("保存文件");

ex = new MenuItem("退 出");

file.add(nw); // 将各子菜单项加入到主菜单项中

file.add(op);

file.add(cl);

file.add(sf);

file.add(ex);

edit = new Menu("编辑");

copy = new MenuItem("复制");

cut = new MenuItem("剪切");

paste = new MenuItem("粘贴");

edit.add(copy);

edit.add(cut);

edit.add(paste);

edit.addSeparator();

mainmenubar.add(file); // 将主菜单项加入到主菜单栏

mainmenubar.add(edit);

setMenuBar(mainmenubar); // 为窗体设置主菜单

nw.addActionListener(this); // 为各菜单项注册事件侦听器

op.addActionListener(this);

cl.addActionListener(this);

sf.addActionListener(this);

ex.addActionListener(this);

copy.addActionListener(this);

cut.addActionListener(this);

paste.addActionListener(this);

}

// 重写ActionEvent事件处理方法

public void actionPerformed(ActionEvent e) {

Object ob = e.getSource(); // 获取事件对象

JFileChooser f = new JFileChooser(); // 创建文件选择器对象

if ((ob == nw) || (ob == cl)) { // 选择"新建文件"或"关闭文件"子菜单项

tx.setText(""); // 清空文本区

} else if (ob == op) { // 选择"打开文件"子菜单项

// 弹出具有自定义按钮的文件选择器对话框

f.showOpenDialog(this);

try {

// 创建StringBuffer对象s

StringBuffer s = new StringBuffer();

// 创建FileReadder对象in，参数为在文件选择器中选中的文件

FileReader in = new FileReader(f.getSelectedFile());

// 读取文件内容，将其追加到s中

while (true) {

int b = in.read();

if (b == -1)

break;

s.append((char) b);

}

tx.setText(s.toString()); // 将文件内容显示在文本区

in.close(); // 关闭文件

} catch (Exception ee) {

}

} else if (ob == sf) { // 选择"保存文件"子菜单项

f.showSaveDialog(this); // 显示文件选择对话框

try {

// 创建FileWriter对象，其参数为选择的文件

FileWriter out = new FileWriter(f.getSelectedFile());

out.write(tx.getText()); // 将文本区内容写入文件

out.close(); // 关闭文件

} catch (Exception ee) {

}

} else if (ob == ex) { // 选择"退 出"子菜单项

System.exit(0); // 退出系统

} else if (ob == copy) {

tx.copy();

} else if (ob == cut) {

tx.cut();

} else if (ob == paste) {

tx.paste();

}

}

public static void main(String[] args) {

new TextEditor("简易文本编辑器");

}

}

# 第12章 Java网络编程

1．选择题

（1）A （2）B （3）C（4）D （5）B

2．填空题

（1）receive()

（2）accept()

（3）getInputStream() getOutputStream()

3．编程题

/\* RemoteFileServer.java \*/

package chapter12.program;

import java.io.\*;

import java.net.\*;

public class RemoteFileServer {

protected DataOutputStream socketWriter;// 声明数据字节输出流对象socketWriter

protected DataInputStream socketReader;// 声明数据字节输入流对象socketReader

protected FileReader fileReader; // 声明字符文件输入流对象fileReader

// 声明缓存字符输入流对象bufferedFileReader

protected BufferedReader bufferedFileReader;

protected int listenPort; // 声明监听端口listenPort

public RemoteFileServer(int listenPort) {

this.listenPort = listenPort;

}

// 允许客户机连接到服务器,等待客户机请求

public void acceptConnections() {

try {

ServerSocket server = new ServerSocket(listenPort);

Socket connection = null;

while (true) {

connection = server.accept();

handleConnection(connection);

}

} catch (IOException e) {

e.printStackTrace();

}

}

// 与客户机Socket交互以将客户机所请求的文件的内容发送到客户机

public void handleConnection(Socket connection) {

try {

socketWriter = new DataOutputStream(connection.getOutputStream());

socketReader = new DataInputStream(connection.getInputStream());

// 从socketReader中读取文件名，并创建fileReader

fileReader = new FileReader(new File(socketReader.readUTF()));

// 创建bufferedFileReader

bufferedFileReader = new BufferedReader(fileReader);

String line = null;

// 从bufferedFileReader读取文件的内容，并发送给客户端

while ((line = bufferedFileReader.readLine()) != null) {

socketWriter.writeUTF(line);

}

bufferedFileReader.close();

fileReader.close();

socketWriter.close();

fileReader.close();

} catch (FileNotFoundException e) { // 捕获找不到文件异常

try {

socketWriter.writeUTF("文件不存在，或文件路径不明确！");

bufferedFileReader.close();

fileReader.close();

socketWriter.close();

fileReader.close();

} catch (IOException e1) {

e1.printStackTrace();

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String args[]) {

RemoteFileServer server = new RemoteFileServer(1001);

server.acceptConnections();

}

}

/\* RemoteFileClient.java \*/

package chapter12.program;

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class RemoteFileClient {

protected DataInputStream socketReader;// 声明数据字节输入流对象socketReader

protected DataOutputStream socketWriter;// 声明数据字节输出流对象socketWriter

protected String hostIp; // 声明hostIp

protected int hostPort; // 声明端口hostPort

protected Socket client; // 声明Socket对象client

// 构造方法

public RemoteFileClient(String hostIp, int hostPort) {

this.hostIp = hostIp;

this.hostPort = hostPort;

}

// 向服务器请求文件的内容

public String getFile(String fileNameToGet) {

StringBuffer fileLines = new StringBuffer();

try {

socketWriter.writeUTF(fileNameToGet); // 向服务器发送文件名

socketWriter.flush();

String line = null;

// 读取服务器发送的内容，并添加到fileLines中

while ((line = socketReader.readUTF()) != null)

fileLines.append(line + "\n");

} catch (EOFException e) {

System.out.println("接收数据完成！");

} catch (IOException e) {

}

return fileLines.toString();

}

// 连接到远程服务器

public void setUpConnection() {

try {

client = new Socket(hostIp, hostPort);

socketReader = new DataInputStream(client.getInputStream());

socketWriter = new DataOutputStream(client.getOutputStream());

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String args[]) {

RemoteFileClient remoteFileClient = new RemoteFileClient("localhost",1001);

remoteFileClient.setUpConnection();

// 创建StringBuffer对象fileContents，存放文件内容

StringBuffer fileContents = new StringBuffer();

// 创建Scanner对象scan

Scanner scan = new Scanner(System.in);

System.out.print("请输入文件名：");

// 输入文件名

String fileName = scan.nextLine();

fileContents.append(remoteFileClient.getFile(fileName));

System.out.println(fileContents); // 输出fileContents

scan.close();

}

}

# 第13章 Java数据库编程

1．选择题

（1）B （2）D （3）A

2．填空题

（1）select \* from students

（2）executeUpdate()

（3）executeQuery()

3．编程题

/\* UserLogin.java \*/

package chapter13.program;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.sql.\*;

public class UserLogin extends JFrame {

// 创建鼠标事件侦听器

MouseEventHandler handler = new MouseEventHandler();

JLabel lRegister; // 声明注册标签lRegister

JLabel lChange; // 声明修改密码标签lChange

JTextField tName; // 声明用户名文本框tName

JPasswordField tPassword; // 声明密码文btnLogin本框tPassword

JButton btnLogin; // 声明登录按钮

public UserLogin() {

super("登录"); // 设置窗体标题

setSize(400, 300); // 设置窗体大小

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

// 定义Font对象，表示字体，设置字体、样式和大小

Font font = new Font("宋体", Font.PLAIN, 16);

// 创建组件

JLabel lName = new JLabel("用户名");

lName.setFont(font);

tName = new JTextField();

JLabel lPassword = new JLabel("密码");

lPassword.setFont(font);

tPassword = new JPasswordField();

btnLogin = new JButton("登 录");

btnLogin.setFont(font);

lRegister = new JLabel("用户注册");

lChange = new JLabel("忘记密码");

// 取消窗体布局管理器，设置组件位置和大小

setLayout(null);

lName.setBounds(70, 20, 50, 30);

tName.setBounds(150, 20, 150, 30);

lPassword.setBounds(70, 80, 50, 30);

tPassword.setBounds(150, 80, 150, 30);

btnLogin.setBounds(70, 150, 230,30);

lRegister.setBounds(100, 190, 50,20);

lChange.setBounds(200, 190, 50,20);

// 将组件添加到窗体

Container c = getContentPane();

c.add(lName);

c.add(tName);

c.add(lPassword);

c.add(tPassword);

c.add(btnLogin);

c.add(lRegister);

c.add(lChange);

// 为登录按钮，注册和修改密码标签注册鼠标事件侦听器

btnLogin.addMouseListener(handler);

lRegister.addMouseListener(handler);

lChange.addMouseListener(handler);

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

setVisible(true); // 设置窗体可见

}

public static void main(String[] args) {

new UserLogin();

}

// 定义继承MouseAdapter适配器的事件侦听器类

class MouseEventHandler extends MouseAdapter {

public void mousePressed(MouseEvent e) {

try {

// 如果事件源是btnLogin

if (e.getSource() == btnLogin) {

login(); // 调用login()方法

}

else if (e.getSource() == lRegister) { // 如果事件源是lRegister

// 创建OperateDialog对象，并设置标题为注册

new OperateDialog("用户注册");

}

else if (e.getSource() == lChange) { // 如果事件源是lChange

// 创建OperateDialog对象，并设置标题为修改密码

new OperateDialog("重置密码");

}

} catch (SQLException e1) {

e1.printStackTrace();

}

}

}

public void login() throws SQLException {

String name = tName.getText(); // 获取tName文本

// 获取tPassword文本

String password = String.valueOf(tPassword.getPassword());

int flag = 0; // 1： 密码正确；2：密码错误；0：用户不存在

// 创建Connection类对象con，连接数据库

Connection con = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/account\_sql",

"root", "123456");

// 创建Statement对象tm

Statement st = con.createStatement();

String sql = "select \* from accounts"; // 声明SQL语句字符串sql

// 执行查询语句，返回查询的结果，并赋给res

ResultSet res = st.executeQuery(sql);

while (res.next()) { // 如果当前语句不是最后一条则进入循环

// 调用getString()方法获取各字段值

// 判断用户名和密码是否与数据库中一致

if (res.getString("用户名").equals(name)) {

if (res.getString("密码").equals(password)) {

flag = 1;

break;

} else {

flag = 2;

}

}

}

String information = "";

if (flag == 0)

information = "用户不存在，请注册！";

else if (flag == 1)

information = "用户名和密码正确！";

else if (flag == 2)

information = "密码错误，请重新输入！";

// 弹出一个提示对话框

JOptionPane.showMessageDialog(null, information,

"提示", JOptionPane.INFORMATION\_MESSAGE);

}

}

/\* OperateDialog.java \*/

package chapter13.program;

import java.awt.Container;

import java.awt.Font;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

public class OperateDialog extends JFrame {

String title; // 声明标题字符串title

JTextField tName; // 声明用户名文本框tName

JPasswordField tPassword; // 声明密码文本框tPassword

JPasswordField tPassword1; // 声明密码确认文本框tPassword1

JButton btnConfirm; // 声明确认按钮

public OperateDialog(String title) {

super(title); // 设置窗体标题

this.title = title;

setSize(400, 300); // 设置窗体大小

// 定义Font对象，表示字体，设置字体、样式和大小

Font font = new Font("宋体", Font.PLAIN, 16);

// 创建组件

JLabel lName = new JLabel("用户名");

lName.setFont(font);

tName = new JTextField();

// 根据title设置lPassword文本

JLabel lPassword = new JLabel("密码");

if (title == "密码重置")

lPassword.setText("新密码");

lPassword.setFont(font);

tPassword = new JPasswordField();

JLabel lPassword1 = new JLabel("确认密码");

lPassword1.setFont(font);

tPassword1 = new JPasswordField();

btnConfirm = new JButton("确定");

btnConfirm.setFont(font);

// 取消窗体布局管理器，设置组件位置和大小

setLayout(null);

lName.setBounds(70, 20, 50, 30);

tName.setBounds(150, 20, 150, 30);

lPassword.setBounds(70, 70, 50, 30);

tPassword.setBounds(150, 70, 150, 30);

lPassword1.setBounds(70, 120, 100, 30);

tPassword1.setBounds(150, 120, 150, 30);

btnConfirm.setBounds(140, 200, 100,30);

// 将组件添加到窗体

Container c = getContentPane();

c.add(lName);

c.add(tName);

c.add(lPassword);

c.add(tPassword);

c.add(lPassword1);

c.add(tPassword1);

c.add(btnConfirm);

// 处理单击确认按钮事件

btnConfirm.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

operate(); // 调用operate()方法

} catch (SQLException e1) {

e1.printStackTrace();

}

}

});

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

setVisible(true); // 设置窗体可见

}

public void operate() throws SQLException {

String name = tName.getText(); // 获取tName文本

// 获取tPassword文本

String password = String.valueOf(tPassword.getPassword());

// 获取tPassword1文本

String password1 = String.valueOf(tPassword1.getPassword());

// 如果用户名或密码为空，弹出一个提示对话框

if (name.isEmpty() || password.isEmpty() || password1.isEmpty())

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

"提示", JOptionPane.INFORMATION\_MESSAGE);

else {

// 如果两次输入密码不一致，弹出一个提示对话框

if (!password.equals(password1))

JOptionPane.showMessageDialog(null,"两次密码不一致，请重新输入！",

"提示", JOptionPane.INFORMATION\_MESSAGE);

else {

int flag = 0; // 1：用户已经存在；0：用户不存在

// 创建Connection类对象con，连接数据库

Connection con = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/account\_sql",

"root", "123456");

// 创建Statement对象tm

Statement st = con.createStatement();

String sql = "select \* from accounts"; // 声明SQL语句字符串sql

// 执行查询语句，返回查询的结果，并赋给res

ResultSet res = st.executeQuery(sql);

while (res.next()) { // 如果当前语句不是最后一条则进入循环

// 调用getString()方法获取各字段值

if (res.getString("用户名").equals(name)) {

flag = 1;

}

}

String information = "";

if (title == "用户注册") { // 如果为注册窗口

if (flag == 0) {

// 定义插入语句，将用户名和密码保存到数据库

sql = String.format("insert into accounts value('%s', '%s')",

name, password);

st.executeUpdate(sql);

information = "用户注册成功！";

}

else if (flag == 1)

information = "用户名已存在，请重新设置！";

} else if (title == "重置密码") { // 如果为修改密码窗口

if (flag == 0)

information = "用户不存在！";

else if (flag == 1) {

// 定义更新语句，修改用户名的密码

sql = String.format("insert into accounts value('%s', '%s')",

name, password);

st.executeUpdate(sql);

information = "修改密码成功！";

}

}

// 弹出一个提示对话框

JOptionPane.showMessageDialog(null, information,

"提示", JOptionPane.INFORMATION\_MESSAGE);

}

}

}

}