



**《Java程序设计》实验及课程期末大作业**

学 院 计算机学院

专 业 网络工程

班 别 2019级四班

学 号 3119005369

姓 名 高子淇

指导教师 宋 玮

2021**年7月1日**

**评分标准**

|  |  |  |
| --- | --- | --- |
| 序号 | 评分内容 | 得分 |
| 1 | 文档的规范（40分）  遵照实验和大作业模板进行规范编写。  字体，字号，行距正确；图表格式规范；文字通顺，文本清晰美观。  报告结构完整，能正确使用类图，顺序图，状态图，时序图等图形化手段对系统进行功能设计描述。 |  |
| 2 | 实践期间的表现（10分）  学习态度端正，按时上交完整文档和源码。 |  |
| 3 | 实验以及大作业的完成度与正确性（30分）  完成全部实验；完成每个实验的各项要求。  完成选定的大作业，完成大作业的主要功能。  代码能正确运行。 |  |
| 5 | 其它(20分)：  1、根据项目需求合理选择及使用数据库或文件系统；并在报告中正确描述文件系统或数据库的设计与使用。  2、根据项目需求合理选择及使用其他相关技术，并在报告中正确描述相关技术的设计与使用。  3、界面设计完整。 |  |
|  |  | 总分： |

**实验一 Java语言基础**

1. 实验内容

题目1：Helloworld的编写,打印出字符串，字符串的内容包括学号，班级，姓名，Helloworld。使用记事本和集成开发环境两种方式。

题目2：模拟考试试卷的乱序处理，安排一个一维数组存放1-20之间不重复的数值，产生一个新的数组存放原来数组元素乱序处理后的数据。分别打印出新旧数组。

基本思路：数组作为方法参数；Array类的使用

题目3：利用随机函数产生25个随机整数给一个5行5列的二维数组赋值。按行列输出该数组；求其最外一圈元素之和；求主对角线中最大元素的值，指出其位置。

基本思路：求最外一圈元素之和的关键是找出最外一圈元素的特征。主对角线上元素的特征是行列值相等。

1. 实验题目分析与设计

题目一：

图示

描述已自动生成

题目二：

处理乱序用的方法是随机生成一个数组索引数字，用for循环对索引和当前数字进行调换。

图示

描述已自动生成

题目三：

按行列输出直接用for循环，最外一圈元素之和，对第一行和最后一行所有元素相加，对中间行第一列和最后一列加和。对角线最大元素值可以用一个循环对对角线上元素进行对比，就可取出最大元素。

图示

描述已自动生成

三、运行结果

题目一：



题目二：

输出第一行是顺序数组1-10，第二行是乱序处理后的数组



题目三：

二维数组的生成我们用的是随机数生成，随机生成1-10之内的整数放入数组中。如图可以计算出最外围之和为73，对角线最大元素是9.

文本

描述已自动生成

**四、附录：**

题目一代码：

public class main{

public static void main(String args[]){

System.out.print("3119005369 "+"网络工程四班 "+"高子淇 "+"Helloworld");

}

}

题目二代码：

import java.util.Collections;

import java.util.Random;

public class main{

public static void main(String args[]){

int[] a=new int[]{1,2,3,4,5,6,7,8,9,10};

int[] b=new int[10];

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

b[i]=a[i];

}

int ranIndex=0;

Random random=new Random();

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

ranIndex=random.nextInt(10);

swap(b,ranIndex,i);

}

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

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

}

System.out.println();

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

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

}

}

private static void swap(int[] b, int ranIndex, int i) {

int temp;

temp=b[ranIndex];

b[ranIndex]=b[i];

b[i]=temp;

}

}

题目三代码：

import java.lang.reflect.Array;

import java.util.Random;

public class main {

public static void main(String args[]){

int[][] arr=new int[5][5];

Random random=new Random();

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

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

arr[i][j]=random.nextInt(10);

}

}

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

System.out.println();

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

System.out.print(arr[i][j]);

}

}

int sum=0;

System.out.println();

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

// System.out.println();

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

//System.out.print(arr[i][j]);

if(i==0||i==4){

sum+=arr[i][j];

}else if(j==0||j==4){

sum+=arr[i][j];

}

}

}

System.out.println("最外圈元素之和："+sum);

int max=0;

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

if(arr[i][i]>=max){

max=arr[i][i];

}

}

System.out.println("主对角线最大元素"+max);

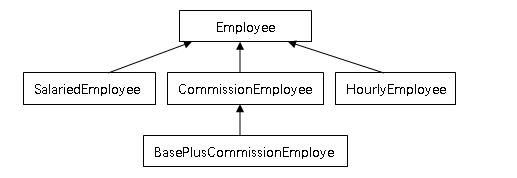
}

}

**实验二** **Java面向对象核心概念及应用**

1. 实验内容

题目一 工资支付系统：为某公司编写一个工资支付系统，用于计算某一类员工的月薪。该公司共有四类员工：领固定月薪的（SalariedEmployee）；计时取酬的（HourlyEmployee，如果一月工时超过160小时，则还需对额外的工时支付加班费）；按销售额提成（CommissionEmployee）的和带底薪并按销售额提成的（BasePlusCommissionEmployee），其继承层次结构如下所示。已知每类员工均有表示员工工号、姓名和出生年月的属性，和用于计算员工月薪的方法。创建一个Employee变量数组，保存Employee类层次结构中每个具体类对象的引用，对每个Employee显示其工号、姓名、出生年月和月收入，如果当月是Employee的生日所在的月份，则还另发给他100月作为红包。



题目二 交通工具通用程序：为某研究所编写一个通用程序，用来计算每一种交通工具运行1000公里所需的时间，已知每种交通工具的参数都是3个整数A、B、C的表达式。现有两种工具：Car007 和Plane，其中Car007 的速度运算公式为：A\*B/C，Plane 的速度运算公式为：A+B+C。需要编写三个类：ComputeTime.java,Plane.java,Car007.java和接口Common.java，要求在未来如果增加第3种或多种交通工具的时候，不必修改以前的任何程序，只需要编写新的交通工具的程序。其运行过程如下，从命令行输入ComputeTime的四个参数，第一个是交通工具的类型，第二、三、四个参数分别是整数A、B、C，举例如下：

计算Plane的时间："java ComputeTime Plane 20 30 40"

计算Car007的时间："java ComputeTime Car007 23 34 45"

如果第3种交通工具为Ship,则只需要编写Ship.java，运行时输入："java ComputeTime Ship 22 33 44"

提示：1、实例化一个对象的另外一种办法：Class.forName(str).newInstance（）；例如需要实例化一个Plane对象的话，则只要调用Class.forName("Plane").newInstance()便可。

2、注意分析程序中有可能产生的异常，根据需要进行异常捕获和处理。

题目三 猜数程序：

编写一个猜数程序。命令行显示菜单：1.开始；2.退出。

用户选择1，则程序生成一个0~99之间的随机整数，命令行显示“请输入你猜的数：”让用户猜。用户输入猜测的数据，猜对了命令行显示“你猜对了”，并且出现菜单：1.再来一次；2.退出。没有猜对程序给出提示（如：太大了，太小了），并要求在命令行继续输入猜测的值；三次没猜对则程序公布正确的数字，并且出现菜单：1.再来一次；2.退出。猜数时任何一个环节，命令行上总会显示用户的总得分情况。得分规则如下：一次猜中得3分，第二次猜中得2分，其三次得1分，三次没有猜中扣2分。主类的名称定为GuessNumber。

提示：1、题目只是大致描述了命令行的输入和输出的要求，可以自行设计命令行的输入输出的形式，使得用户在使用时觉得界面友好。同样，整个猜测的流程可以根据自己的理解进行优化和调整，使得用户在使用时更为流畅。

2、注意分析程序中有可能产生的异常，根据需要进行异常捕获和处理。

题目四 婚配系统：编写一个Person类，定义人的基本属性:name（姓名）、age(年龄)、gender（性别）、partner（配偶）。

要求至少为Person类写一个marry(Person p)方法，来判断两个人是否可以结婚。如果可以结婚，将其partner属性赋值为其配偶。

结婚必须满足以下3个条件。

 必须是异性，同性不允许结婚。

 有配偶者不能结婚。

 要达到年龄方可结婚：男方23岁以上，女性21岁以上。

1. 实验题目分析与设计

题目一：

我们创建一个抽象类员工，抽象方法工资，具体员工的工资结算就可以改变，后面有其他结算方法的工种就只需要写工资结算方法就行了。然后创建四个员工工资结算的类。在主类里写了让员工输入基本信息的方法。

图示

描述已自动生成

题目二：

创建一个接口，输入交通工具就可以通过接口获取相对应的类，然后计算。

图示

描述已自动生成

题目三：

随机生成一个数，for循环3次输入，最后计算得分。

图示

描述已自动生成

题目四：

Person类用来存储信息，先输入配偶人和被配偶人的信息，先查看双方是否已经有配偶，然后逻辑语句比较是否符合条件，最后输出结果。

图示

描述已自动生成

三、运行结果

题目一：

文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

题目二：

图形用户界面, 文本

描述已自动生成

题目三：

文本, 信件

描述已自动生成

题目四：

文本

低可信度描述已自动生成

四、附录：

题目一代码：

abstract class Employee{

public int number;

public String name;

public int birthYear;

public int birthMonth;

public static int salary;

public abstract int salary();

}

interface GetSalaryPerMonth{

int SalaryPerMonth=5000;

}

class SalariedEmployee extends Employee implements GetSalaryPerMonth{

static int RedEnvelope = 100; //生日红包

public int salary() {

int month = Calendar.getInstance().get(Calendar.MONTH) + 1; //获取当前月份

if (month == birthMonth)

salary = SalaryPerMonth + RedEnvelope;

else

salary = SalaryPerMonth;

return salary;

}

}

interface Getsale{

int salarypersale=10;

}

class CommissionEmployee extends Employee implements Getsale{

// public static int RedEnvelope=100;

static int RedEnvelope = 100;

public int salary(){

int month=Calendar.getInstance().get(Calendar.MONTH)+1;

int sale;

System.out.print("输入销售额：");

sale=new Scanner(System.in).nextInt();

if(month==birthMonth)

salary=RedEnvelope+salarypersale\*sale;

else

salary=salarypersale\*sale;

return salary;

}

}

interface GetBaseSalary{

int basesalary=1000;

}

class BasePlusCommissionEmployee extends CommissionEmployee implements GetBaseSalary{

public int salary(){

int sale=super.salary();

sale=sale+basesalary;

return sale;

}

}

interface Gethoursale{

int intimehoursale=30;

int outtimehoursale=40;

}

class HourlyEmployee extends Employee implements Gethoursale{

static int RedEnvelope = 100;

public int salary(){

int month=Calendar.getInstance().get(Calendar.MONTH)+1;

int hour;

hour=new Scanner(System.in).nextInt();

if(month==birthMonth){

if(hour>160){

salary=RedEnvelope+160\*intimehoursale+(hour-160)\*outtimehoursale;

}

else{

salary=RedEnvelope+hour\*intimehoursale;

}

}else{

if(hour>160){

salary=160\*intimehoursale+(hour-160)\*outtimehoursale;

}

else{

salary=hour\*intimehoursale;

}}

return salary;

}

}

public class system {

public static String get\_name() {

System.out.print("请输入名字：");

String name=new Scanner(System.in).next();

System.out.println();

return name;

}

public static int get\_birthMonth() {

System.out.print("请输入生日月份：");

int month=new Scanner(System.in).nextInt();

System.out.println();

return month;

}

public static int get\_birthyear() {

System.out.print("请输入生日年份：");

int year=new Scanner(System.in).nextInt();

System.out.println();

return year;

}

public static int get\_number() {

System.out.print("请输入工号:");

int number=new Scanner(System.in).nextInt();

System.out.println();

return number;

}

public static void main(String args[]) {

Employee[]data=new Employee[4];

data[0]=new SalariedEmployee();

data[1]=new CommissionEmployee();

data[2]=new BasePlusCommissionEmployee();

data[3]=new HourlyEmployee();

//print=new System.out.print();

while(true) {

System.out.println("输入员工类型：1.领固定月薪 2.按销售额提成 3.带底薪并按销售额提成 4.计时取酬");

int i=new Scanner(System.in).nextInt();

switch(i) {

case 1:

data[0].name=get\_name();

data[0].birthMonth=get\_birthMonth();

data[0].birthYear=get\_birthyear();

data[0].number=get\_number();

System.out.println("姓名："+data[0].name);

System.out.println("工号:"+data[0].number);

System.out.println("生日年份："+data[0].birthYear);

System.out.println( "生日月份："+data[0].birthMonth);

System.out.println("工资："+data[0].salary());

break;

此处省略。。。

}

}

}

}

题目二代码：

ComputeTime.java

public class ComputeTime {

public static void main(String args[]) {

String name=args[0];

int A=Integer.parseInt(args[1]);

int B=Integer.parseInt(args[2]);

int C=Integer.parseInt(args[3]);

try {

Common x=(Common)Class.forName("java\_exp2\_2."+name).getDeclaredConstructor().newInstance();

System.out.print(x.timecompute(A, B, C));

} catch (Exception e) {

System.out.println("class not found");

}

}

}

Common.java

public interface Common {

public int timecompute(int A,int B,int C);

}

Car007.java

public class Car007 implements Common{

public int timecompute(int A,int B,int C) {

return (A\*B/C)\*1000;

}

}

Plane.java

public class Plane implements Common {

public int timecompute(int A,int B,int C) {

return (A+C+B)\*1000;

}

}

题目三代码：

import java.util.Random;

import java.util.Scanner;

public class guessnumber {

public static void main(String args[]) {

System.out.println("1.开始；2.结束");

int main=new Scanner(System.in).nextInt();

if(main==1) {

int score=0;

while(true) {

Random random=new Random();

System.out.print("请输入你猜的数：");

int guessnumber=random.nextInt(100);

int number=new Scanner(System.in).nextInt();

int i=0;

for(i=0;i<2;i++) {

if(number==guessnumber) {

System.out.println("猜对了");

switch(i) {

case 0:

score+=3;

case 1:

score+=2;

case 2:

score+=1;

}

System.out.print("得分:"+score);

// System.out.println("1.再来一次；2.退出");

main=new Scanner(System.in).nextInt();

}

else if(number!=guessnumber) {

if(number<guessnumber) {

System.out.println("太小了");

}

else {

System.out.println("太大了");

}

System.out.print("再次输入:");

number=new Scanner(System.in).nextInt();

}

}

if(i==2) {

score-=2;

System.out.print("得分:"+score);

System.out.println("正确数字:"+guessnumber);

}

System.out.println("1.再来一次；2.退出");

main=new Scanner(System.in).nextInt();

if(main==2) {

break;

}

}

}

}

}

题目四代码：

Person.java

public class Person {

String name=new String();

int age;

String gender;

Person partner=null;

Person(String name,int age,String gender){

this.age=age;

this.gender=gender;

this.name=name;

}

void marry(Person P) {

if(this.partner!=null) {

System.out.println("已有配偶");

return;

}

if(this.gender!=P.gender) {

if((this.age>23&&this.gender.equals("male"))||(this.age>21&&this.gender.equals("female"))) {

if((P.age>23&&P.gender.equals("male"))||(P.age>21&&P.gender.equals("female"))) {

this.partner=P;

P.partner=this;

System.out.println("可以结婚");

return;

}

}

}

System.out.println("不符合条件无法结婚");

return;

}

}

Main.java

import java.util.Scanner;

public class main {

public static void main(String args[]) {

while(true) {

System.out.println("输入本人信息：");

System.out.println("输入姓名：");

String name=new Scanner(System.in).next();

System.out.println("输入年龄：");

int age=new Scanner(System.in).nextInt();

System.out.println("输入性别：");

String gender=new Scanner(System.in).next();

System.out.println("输入欲配偶人信息：");

System.out.println("输入姓名：");

String pname=new Scanner(System.in).next();

System.out.println("输入年龄：");

int page=new Scanner(System.in).nextInt();

System.out.println("输入性别：");

String pgender=new Scanner(System.in).next();

//int age=new

Person P1=new Person(name,age,gender);

Person P2=new Person(pname,page,pgender);

P1.marry(P2);

}

}

}

**实验三** **Java语言的高级特性**

1. 实验内容

题目一 用多线程模拟多用户汇款。

假设某一银行账户，它可接受顾客的汇款，每做一次汇款，便可计算出汇款的总额。现有多个顾客，每人分多次将钱(每个顾客汇入的时间和钱数不一样)汇入到该银行账户。试编写一个程序，模拟实际作业。

[基本要求] 运行程序5 次，观察每次运行结果是否相同，并对现象做解释。

题目二 socket通信。

利用Socket类和ServerSocket类编写一个C/S程序，实现C/S通信。

客户端向服务器端发送请求给服务端，形式如“From 学号 姓名： what time is it ”；服务器端接受到该字符串后将服务器端当前时间返回给客户端，形式如“To 姓名： 当前时间”；客户端向服务器端发送“From 学号 姓名：Exit”命令，服务器端向客户端返回“To 姓名：Bye”后退出。

题目三 学生数据存储。

编写一个student类用来描述学生对象，创建若干学生，将其写入文件；再从文件读出学生信息，展示在屏幕上。

1. 实验题目分析与设计

题目一：

我们编写一个bank类来计算转账，再写一个线程类account用来实现不同用户的转账。我们创建三个客户，每个客户进行三次转账，金额已经设定好了，每个客户转账金额不同。

图片包含 图示

描述已自动生成

题目二：

我们创建一个客户端和服务端，服务端先打开监听，再打开客户端发送流，服务端接收到后输入流打开，准备接收客户端的信息。客户端先输入姓名学号建立本地信息，然后输入发送信息发送，服务端接收到信息后分析信息字符串，根据不同字符串进行处理，再返回相应应答。如果输入Exit，服务端接收到先发送一个bye回来再关闭输入输出流，客户端收到后再关闭。

图示

描述已自动生成

题目三：

创建student类存储信息，在主类里写一个WriteFile和ReadFile实现读和写

图示

描述已自动生成

1. 运行结果

题目一：

可以看到每次的结果都不同，因为线程之间存在竞争，每次进程时间都不一样，所以会造成过程不一样

文本

描述已自动生成

文本, 表格

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

文本, 表格

描述已自动生成

题目二：

先在本地输入学号和姓名，这个只保存在客户端，输入字符串发送给服务端，服务端返回相应回答。

文本

描述已自动生成

题目三：

可以看到输入的学生信息已经写入本地记事本。读取格式也和txt文件里面的一样。

文本

描述已自动生成

文本, 信件

描述已自动生成

1. 附录：

题目一代码：

class bank{

static int balance=0;

static void take(String name,int money) {

int temp=balance;

temp+=money;

//try {Thread.sleep((int)(1000\*Math.random()));}

// catch(InterruptedException e) {}

balance = temp;

System.out.println("用户："+name+" 银行总余额为：" + balance);

}

}

class account extends Thread{

int money;

String name;

account(int money,String name){

this.money=money;

this.name=name;

}

public void run() {

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

bank.take(this.name,money/3);

}

}

}

public class bankmoney {

public static void main(String args[]) {

account c1=new account(3000,"客户1");

account c2=new account(300,"客户2");

account c3=new account(6000,"客户3");

c1.start();

c2.start();

c3.start();

}

}

题目二代码:

客户端

package java\_exp3\_2;

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class TestClient{

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

Socket socket=new Socket(InetAddress.getLocalHost().getHostAddress(),9999);

OutputStream out=socket.getOutputStream();

System.out.println("请输入学号和姓名：");

String number=new Scanner(System.in).next();

String name=new Scanner(System.in).next();

while(true) {

System.out.print("From "+number+" "+name+":");

String text=new Scanner(System.in).nextLine();

byte[] bytes = text.getBytes();

out.write(bytes);

out.flush();

InputStream in=socket.getInputStream();

byte [] bytes1 = new byte[1024];

int count = in.read(bytes1);

String content = new String(bytes1,0,count);

System.out.println("To "+name+":"+content);

if(content.equals("Bye")) {

out.close();

in.close();

break;

}

}

}

}

服务端

package java\_exp3\_2;

import java.net.\*;

import java.text.SimpleDateFormat;

import java.io.\*;

import java.util.\*;

public class TestServer {

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

ServerSocket server=new ServerSocket(9999);

Socket socket=server.accept();

while(true) {

InputStream in=socket.getInputStream();

byte [] bytes = new byte[1024];

int count = in.read(bytes);

System.out.print(count);

String content = new String(bytes,0,count);

System.out.print(content);

if(content.equals("what time is it")) {

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String text=df.format(new Date());//设置日期格式

//System.out.println(df.format(new Date()));// new Date()为获取当前系统时间

OutputStream out=socket.getOutputStream();

byte[] bytes1 = text.getBytes();

out.write(bytes1);

out.flush();

}else if(content.equals("Exit")) {

String text="Bye";

OutputStream out=socket.getOutputStream();

byte[] bytes1 = text.getBytes();

out.write(bytes1);

out.flush();

out.close();

in.close();

return;

}

else {

String text="woc";

OutputStream out=socket.getOutputStream();

byte[] bytes1 = text.getBytes();

out.write(bytes1);

out.flush();

}

}

}

}

题目三代码：

Student.java

public class student {

String id;

String grade;

String name;

student(String id,String grade,String name){

this.name=name;

this.id=id;

this.grade=grade;

}

String return\_information() {

return "学号："+this.id+"\t年级："+this.grade+"\t姓名："+this.name;

}

}

Writeread.java

import java.io.\*;

import java.util.\*;

public class WriteRead {

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

System.out.print("请输入录入学生人数：");

int num=new Scanner(System.in).nextInt();

student[] stu=new student[num];

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

System.out.print("请输入学号：");

String id=new Scanner(System.in).next();

System.out.print("请输入年级：");

String grade=new Scanner(System.in).next();

System.out.print("请输入姓名：");

String name=new Scanner(System.in).next();

stu[i]=new student(id,grade,name);

}

File file=new File("Student.txt");

WriteFile(file,stu);

readfile(file);

}

public static void WriteFile(File file, student []stu) throws IOException //写入文件

{

FileWriter fw = new FileWriter(file,false); //要覆盖之前的内容，选择false

BufferedWriter bw = new BufferedWriter(fw);

for (student s : stu)

{

bw.write(s.return\_information());

bw.newLine(); //写入一个学生的信息后换行

}

bw.close();

fw.close();

}

public static void readfile(File file) throws IOException {

FileReader fr=new FileReader(file);

BufferedReader br=new BufferedReader(fr);

String s;

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

System.out.println(s);

}

br.close();

fr.close();

}

}

# 1 大作业题目

## 第2题：交通信号灯模拟

模拟实现十字路口的交通灯管理系统逻辑.

1.随机生成按照各个路线行驶的车辆

2. 信号灯忽略黄灯，只考虑红灯和绿灯。

3. 应考虑左转车辆受信号灯控制，右转车辆不受信号灯控制。

4. 具体信号灯控制逻辑与现实生活中普通交通灯控制逻辑相同，不考虑特殊情况下的控制逻辑：

南北车辆与东西车辆交替放行，同方向车辆等待应先放行直行车辆，后放行左转车辆。

图示

描述已自动生成

图1.1

5. 每辆车通过路口的时间为1秒（提示：可通过线程sleep的方式模拟）。

6. 随机生成车辆时间间隔以及红绿灯交换时间间隔自定，可以设置。

# 2 需求分析

根据题目要求可以看出，我们要创造一个多线程处理的程序，先分析可能出现的十字路口的车辆情况：

四个路口，每个路口只有左转、右转和直行三种选择。右转不受信号灯控制。我们知道南北和东西是交替同行的，也就是说顺序是南北直行->南北左转->东西直行->东西左转这个顺序进行，我们又知道朝东和朝西，朝南和朝北，这之间直行是不冲突的，两个朝向只需要设计一个信号灯就够了。再算上直行和左转分别一个，我们就需要四个信号灯就够了，但我们只用一个线程去实现，而不是四个线程，因为四个状态是循环的，没必要使用四个线程，线程之间的延迟问题反而麻烦，然而在同一个循环下绝对不会出现同时出现两个绿灯的情况，反而更加安全。

然后是小车，我们随即设计产生车辆的多少和方向，我们只需要知道车辆是南北方向还是东西方向，是左转右转还是直行就可以做出判断了。我们同样只用一个线程实现，一个列表存储当前路口的车辆就够了，只需循环这个列表就行，循环的时间相比信号灯的交换时间是非常短的。

**3模拟交通信号灯系统的设计**

**3.1 系统的总体设计**

系统的总体使用了四个类，分别是main.java，car.java，createcar.java，light.java。main类实现整个系统的运行。Car类用来存储车辆信息，createcar类用来一定时间间隔随机创建一批车辆到达路口，light类实现信号灯的交替。

**3.2 系统的设计**

灯线程流程图

图示

描述已自动生成

图1.2

生成车线程流程图

图示

描述已自动生成

图1.4

主线程流程图

图示

描述已自动生成

图1.4

**3.3 数据库的设计或文件结构的设计**

图形用户界面, 文本, 应用程序

描述已自动生成

图1.5

表格

中度可信度描述已自动生成**3.4 界面的设计**

图1.6

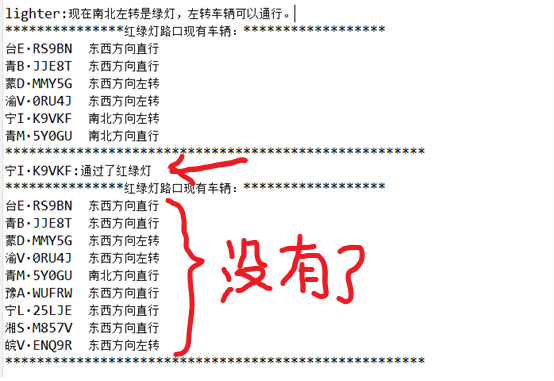
图1.6基本包含了界面的所有设计，lighter：表示现在是什么方向的信号灯是绿灯，红绿灯路口现有车辆：列表下面是当前路口的车辆，每次生成新车辆会打印一次。

图1.7

图1.7最下面是打印车辆的通过情况，哪些车通过了红绿灯会在下面打印，在下次打印列表的时候这辆车就会被删除。

**4 某某系统的实现**

**4.1项目结构及配置**

图形用户界面, 文本, 应用程序

描述已自动生成

主程序类在main.java，light.java实现了灯线程，createcar.java用来生成car类的小车。

**4.2 实现的关键代码**

主类：

public static void main(String args[]) {

System.out.println("网络工程4班高子淇3119005369");

System.out.println("模拟红绿灯信号开始");

System.out.println();

l.start();

cars.start();

while (true) {

try {

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

car c =(car)cars.list.get(i);

if(c.getFinish()){

continue;

}

switch (c.getDirection()) {

case 0:

if (l.forwardstate1.equals("red")) {

break;

} else {

sleep(1000);

System.out.println(c.getName() + ":通过了红绿灯");

cars.list.remove(i);

// i=i-1;

c.setFinish();

break;

}

case 1:

if (l.leftstate1.equals("red")) {

break;

} else {

sleep(1000);

System.out.println(c.getName() + ":通过了红绿灯");

cars.list.remove(i);

// i=i-1;

c.setFinish();

break;

}

case 2:

if (l.forwardstate2.equals("red")) {

break;

} else {

sleep(1000);

System.out.println(c.getName() + ":通过了红绿灯");

cars.list.remove(i);

// i=i-1;

c.setFinish();

break;

}

case 3:

if (l.leftstate2.equals("red")) {

break;

} else {

sleep(1000);

System.out.println(c.getName() + ":通过了红绿灯");

cars.list.remove(i);

c.setFinish();

break;

}

case 4:

sleep(1000);

System.out.println(c.getName()+"右转无需考虑信号灯");

cars.list.remove(i);

// i=i-1;

c.setFinish();

break;

}

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

Light类：

public void run(){

while(true){

try{

forwardstate1 = "green";

System.out.println();

System.out.println("lighter:现在南北直向是绿灯，前行车辆通行");

Thread.sleep(20\*1000);

//10s后灯变绿，设绿灯时间位5秒

forwardstate1 = "red";

leftstate1="green";

System.out.println();

System.out.println("lighter:现在南北左转是绿灯，左转车辆可以通行。");

Thread.sleep(20\*1000);

forwardstate1 = "red";

leftstate1="red";

forwardstate2="green";

System.out.println();

System.out.println("lighter:现在东西前行是绿灯，前行车辆可以通行。");

Thread.sleep(20\*1000);

forwardstate2="red";

leftstate2="green";

System.out.println();

System.out.println("lighter:现在东西左转是绿灯，左转车辆可以通行。");

Thread.sleep(20\*1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

Car类：

public class car{

private int direction;

private String name;

private boolean finish=false;

car(String name,int direction){

this.direction=direction;

this.name=name;

}

void setFinish(){

this.finish=true;

}

boolean getFinish(){

return this.finish;

}

String getName() {

return this.name;

}

int getDirection(){

return this.direction;

}

String getrealdirection(){

if(this.direction==0){

return "南北方向直行";

}

else if(this.direction==1){

return "南北方向左转";

}

else if(this.direction==2){

return "东西方向直行";

}

else if(this.direction==3){

return "东西方向左转";

}

else{

return "右转";

}

}

}

Createcar类：

public void run(){

Random random=new Random();

while(true){

try{

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*红绿灯路口现有车辆：\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

for(int i=0;i<random.nextInt(6);i++){

car c=new car(generateCarID(), random.nextInt(5));

list.add(c);

}

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

car a=(car)list.get(i);

System.out.println(a.getName()+" "+a.getrealdirection());

}

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

//wait(5000);

sleep(10000);

}catch (InterruptedException e) {

e.printStackTrace();

}

}

}

表格

中度可信度描述已自动生成**5 测试数据与运行结果**

图1.8

表格

描述已自动生成从图1.8中可以看出，第一个绿灯是南北直行，这时候下面打印的当前路口的车辆没有南北直行的车辆所以没有车辆通过，又过了十秒线程自动生成随即车辆，可以看到这次没有新的车辆生成，并且在生成之后会再打印一次当前路口的车辆。

图1.9

这里可以看到20秒后南北直行变成了南北左转，并且再次生成了一批车辆，这次生成的有南北左转的，可以看到宁IK9VKF通过了红绿灯，在看到下一次列表中，这辆车就已经不在这个路口列表里了。

表格

描述已自动生成

图1.10

表格

描述已自动生成

图1.11

可以从图1.11看到第二次生成车辆有右转车辆，根据右转无需考虑信号灯的原则，直接通过。

**总结**

在程序设计中，关键在于理清思路，到底要几个信号灯可以实现，一定越多线程越好吗？小车的生成和类需要哪些变量？经过思考和实际生活，我们最终只使用一个线程去实现整个路口的信号灯，为了实现互斥，不会出现多个线程延时问题。每一个绿灯一定是在之前绿灯变红之后才会做出改变。略有不足的地方可能是如何不断查询是否列表中又可以通过的小车。我是直接用一个while循环实现不断查表，毕竟循环整个表的时间相比红绿灯的等待时间可以说是忽略不计的，当然在解决红绿灯问题是没有问题，但在解决计算量更大的问题就会出问题了。

这次设计让我熟练地掌握了线程的使用，以及多个线程之间的交互，比较好的掌握了java的thread特性。