软件技术基础综合课程设计实验指导

以下的示例只提供一种思路，仅作为参考、学习使用。

示例（Java）

OS:win10

IDE: myeclipse2017

Java: jdk-8u241-windows-x64

Jar 包: mysql-connector-java-5.1.39-bin.jar

数据库：MySQL，版本：mysql-5.7.28-winx64

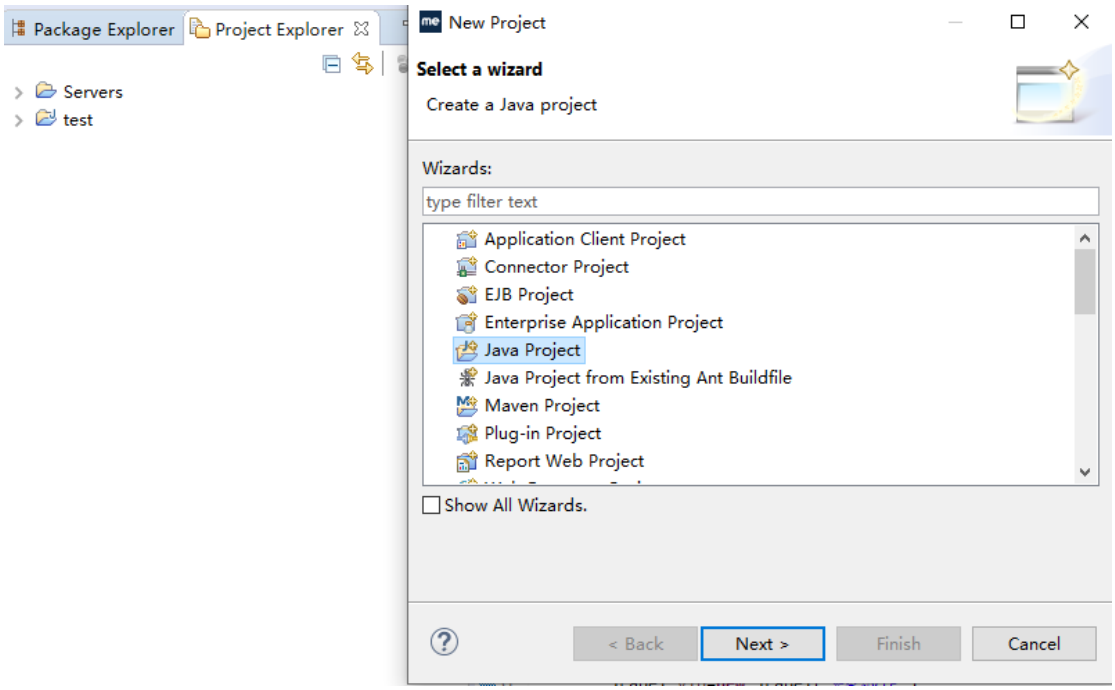
1．创建 Java 工程

（1）安装及配置 java:

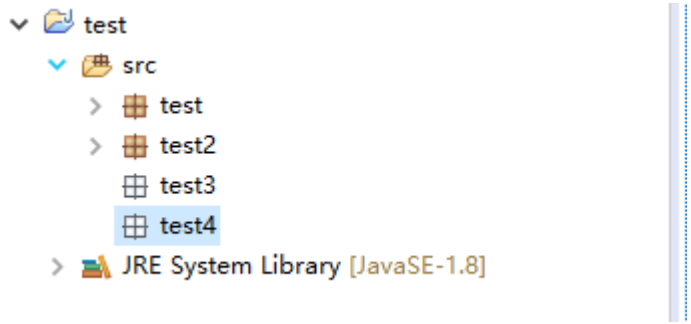
https://blog.csdn.net/tang_chuanlin/article/details/79586667

（2）安装 IDE，包括但不限于 eclipse、myeclipse、vscode 等。

默认环境已经搭建好、myeclipse 安装成功。在 project explorer 选项下，右键 new，新建 java project，此处命名为 test。



展开 test，在 src 处右键新建 package，在 package 下新建 class。Class 即为.java 文件，所有的代码在.java 文件中实现，也可以建立多个 class。



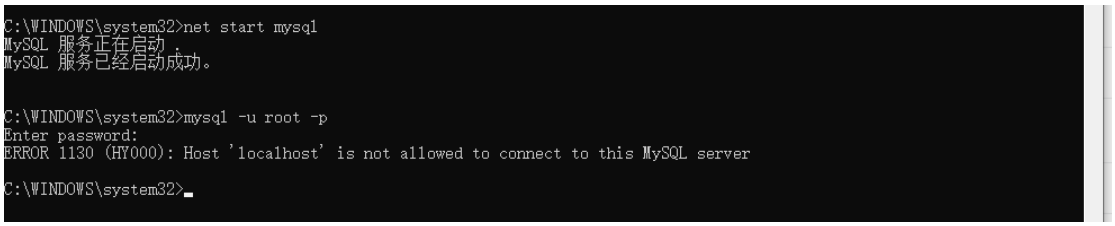
此处建立 4 个 package，分别表示 4 个实验。

2．Java 与数据库连接

（1）安装与配置 mysql

https://blog.csdn.net/weixin_42869365/article/details/83472466

DOS 下出现 1130-host 'localhost' is not allowed to connect to this MySQL server 错误



解决办法：

https://blog.csdn.net/Kitsuha/article/details/84555585

（2）navicat 下载与破解（也可以直接使用 MySQL 客户端，navicat 是轻量级的前端，占的空间很小）

https://www.cnblogs.com/yinfei/p/11427259.html

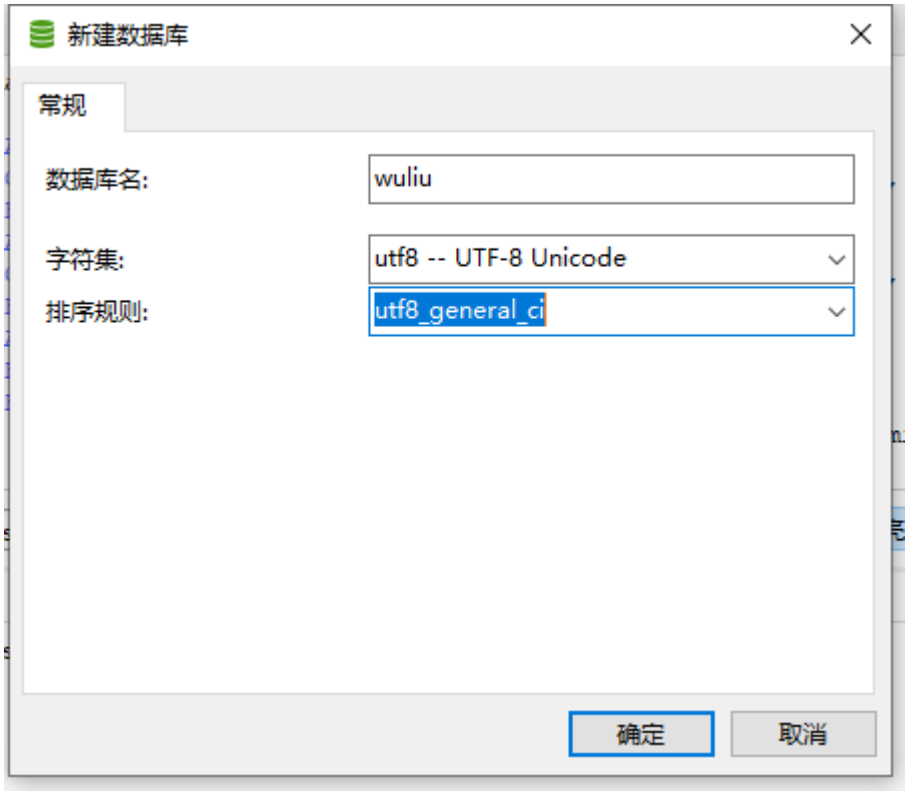
navicat 出现

解决办法：

https://blog.csdn.net/Kitsuha/article/details/84555585

https://blog.csdn.net/u013310119/article/details/88909031

（3）新建数据库，命名为 wuliu



在 wuliu 数据库下新建查询，建表、插入数据

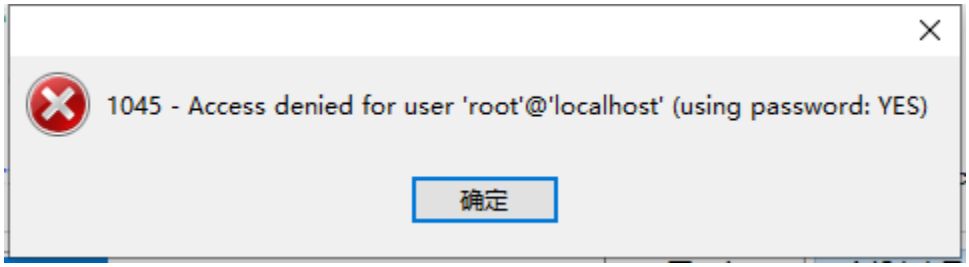
错误[Err] 1146 - Table 'performance_schema.session_status' doesn't exist 的解决办法

https://blog.csdn.net/he_yuhao/article/details/81983294

错误[Err] 1682 - Native table 'performance_schema'.'session_status' has the wrong structure 的解决办法

https://blog.csdn.net/landylxy/article/details/78477332

错误



解决办法：

https://blog.csdn.net/a33130317/article/details/81240197

注意：在配置好 navicat 的连接时，如果还出现 not allowed to connect 的错误，需要使用管理员打开 cmd，输入 net start mysql，用于启动 mysql 服务。关闭服务命令：net stop mysql

（4） java 与 mysql 连接

新建类 MysqlManager，运行如下代码

```
import java.sql.*;

public class MysqlManager {

    public static void main(String[] args) throws Exception {
        // 1.加载数据访问驱动
        Class.forName("com.mysql.jdbc.Driver");
        //2.连接到数据"库"上去
        Connection conn=
DriverManager.getConnection("jdbc:mysql://localhost:3306/wuliu?useSSL=f
alse", "root", "123456");
        //3.构建SQL命令
        Statement state=conn.createStatement();
        String s="insert into goodslist
values('1','1','Shanghai','false','ture')";
        state.executeUpdate(s);
        conn.close();//关闭连接
    }

}
```

数据库中多了一条 values('1','1','Shanghai','false','ture')";数据

1	1	Shanghai	false	ture

表示数据库连接成功

其中，wuliu 表示提前建立的数据库，root 表示数据库用户，123456 表示 root 用户的密码，goodslist 表示 wuliu 这个数据库中的一张表，这张表有五个字段（需要同学们自己设定）。

3．Java 写 GUI 界面 Demo

具体关于 Java GUI 组件的理论知识学习，可参考：

https://www.cnblogs.com/PengLee/p/3917479.html

tips：

单行注释：//

多行注释：/** 注释内容 **/

批量注释：Ctrl+/键

3.1 简单示例 1

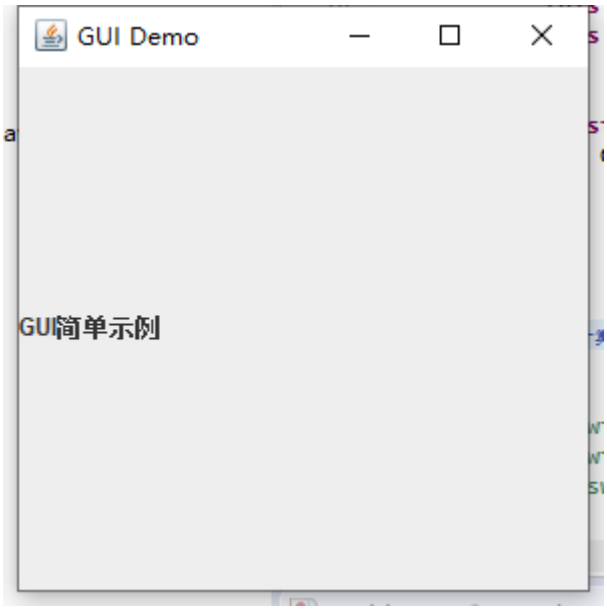
建立类，类名为 GUI，运行时选择 java application。

创建标签、设置窗口大小、名字等

```
import javax.swing.JFrame;//导入需要的窗口包JFrame
import javax.swing.JLabel;//导入需要的标签包JLabel
import javax.swing.*;//导入swing下所有组件
public class GUI extends JFrame{
    public GUI(){
        JLabel jl = new JLabel();//创建一个标签
        jl.setText("GUI简单示例");//标签上显示的文字
        //下面的this都指的是本窗口.都可以省略
        this.add(jl);//窗口添加刚刚创建的标签
        this.setTitle("GUI Demo");//窗口的标题名字
        this.setLocation(200, 200);//窗口的左顶点在屏幕上的位置
        this.setSize(300, 300);//窗口是 宽500像素，长500像素
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);//设置窗口被关闭时候
就退出窗口
        this.setVisible(true);//设置这个窗口能否被看见
    }

    public static void main(String[] args) {
        new GUI();//调用构造方法,创建一个窗口
    }
    }
```

运行结果



3.2 简单示例 2

读取文本框中的内容、设置按钮、实现按钮的响应

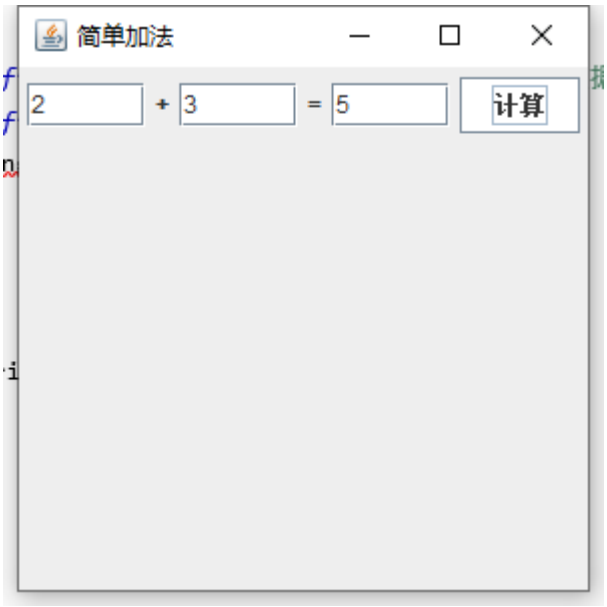
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class GUI
{
static JTextField field1=new JTextField(5);//设置文本框1
static JTextField field2=new JTextField(5);
static JTextField field3=new JTextField(5);
//static JTextField field3=new JTextField("0",5);
public GUI(){
     JFrame f=new JFrame("简单加法");
     f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     f.setSize(300,300);//设置窗口大小
     f.setLocation(300,300);//设置窗口位置
     JLabel jiahao=new JLabel("+");//显示+
     JLabel denghao=new JLabel("=");//显示=
     f.setVisible(true);
     JPanel p1=new JPanel();
     f.setContentPane(p1);
     p1.setLayout(new FlowLayout());

     p1.add(field1);//添加文本框1
     p1.add(jiahao);//添加加号
     p1.add(field2);
     p1.add(denghao);//添加等号
```

```
        p1.add(field3);
        JButton b1=new JButton("计算");
        Color bg=new Color(255,255,255);
        b1.setBackground(bg);
        p1.add(b1);

        b1.addActionListener(new ActionListener()//实现按钮1的响应
        {
        public void actionPerformed(ActionEvent e)    //重写actionPerformed方法
        {
            int a,b;
            a=Integer.parseInt(field1.getText()); //获取TextField1中的数据
            b=Integer.parseInt(field2.getText());
            field3.setText(String.valueOf(a+b));
        }
        }
        );
    }
    public static void main(String args[])
    {
        GUI fd=new GUI();//
    }
}
```

运行效果：



4. 实验一：最优物流路线计算实验

4.1 主要内容

给定物流节点信息，物品信息，节点间代价计算方法要求学生完成以下实验内容。

(1) 根据物品信息综合计算物流物品的优先级别，根据物流优先级别排序物流物品，根据排序结果对物流物品进行逐个发货。

(2) 根据物流物品的物流条件信息，归类物流物品到物流方案类型，物流方案类型可包括：价格最小物流方案，时间最短物流方案、综合最优方案、航空物流方案等。并运用树型结构存储所有的物流物品到划分的物流方案中。

(3) 根据给定的物流节点信息，计算各类物流方案下的物流最短路径

(4) 根据物流最短路径，物流方案和物流优先级发送货物

4.2 程序设计思路（按照自己的理解自行设计，下同）

输入：任意自定义物流节点信息、任意自定义物流物品信息 。

输出：物流物品信息表、根据优先级的物流物品排序结果、查询获得某类物流条件下所有的物品、每个物品的发送路径序列。(归类方案存储物流方案类型可包括： 价格最小物流方案、时间最短物流方案、综合最优方案、航空物流方案等。)(物流最短路径：最小价格、最短时间等。)

(1) 物品为一个类。实例变量由订单号、到达起点的时间、寄件人是否为高级用户、目的地以及物流路径组成，方法有设置物品信息、路径(setter)和获取订单号、时间、目的地是否为高级用户、以及获取路径(getter)。

(2) 物流节点图为另一个类。实例变量由节点数、节点与节点之间关系的邻接矩阵以及节点名组成。方法有从文件中获取邻接矩阵、获取节点名以及使用 dijkstra 算法计算最短路径。

(3) 主函数获取文件中物流物品、物流节点的信息，根据物流物品到达起点的时间以及寄件人是否为高级用户对物流物品进行排序并分类到物流方案(时间最短以及花费最小方案),调用节点类中的 dijkstra 算法计算最短路径，设置图形用户接口，将物流物品信息以表格形式显示在界面上。发货则是将物品从物品列表中删除。

4.3 算法分析与设计

(1) 计算物流优先级：

先到起点的物品先发货，高级用户的物品到达起点的时间提前一天，确保高级用户的优先级。以此计算结果作为衡量标准：

```
//排序依据（先到的先发货
class timeCompare implements Comparator<good>{
    public int compare(good o1, good o2) {
        String time1=o1.getTime();
        String time2=o2.getTime();
        long t1=Long.parseLong(time1);
        long t2=Long.parseLong(time2);
        if(o1.isVip)
            t1-=10000;
        if(o2.isVip)
            t2-=10000;
        if(t1>t2)  return 1;
        else if(t1==t2) return 0;
        else return -1;
    }
}
```

(2) 计算最短路径：

默认起点为第 1 个节点。使用 dijskra 算法可以获得起点到其余各节点的最短路径：

```
public void dijkstra(int[] prev, int[] dist) {
    int vs=0;
    boolean[] flag = new boolean[cityNum];

    for (int i = 0; i < cityNum; i++) {
        flag[i] = false;
        prev[i] = 0;
        dist[i] = cost[vs][i];
    }
    flag[vs] = true;
    dist[vs] = 0;
    int k=0;
    for (int i = 1; i < cityNum; i++) {

        int min = INF;
        for (int j = 0; j <cityNum; j++) {
            if (flag[j]==false && dist[j]<min) {
                min = dist[j];
                k = j;
            }
        }
        flag[k] = true;
        for (int j = 0; j < cityNum; j++) {
            int tmp = (cost[k][j]==INF ? INF : (min + cost[k][j]));
            if (flag[j]==false && (tmp<dist[j]) ) {
                dist[j] = tmp;
                prev[j] = k;
            }
        }
    }
}
```

其中，prev 数组存储了每个节点的前驱节点，dist 数组存储了起点到每个节点的花费。

4.4 数据结构设计

(1) 物品信息

物品信息从文件中获取后存储在名为 goods 的 ArrayList<good>中：

```
ArrayList<good> goods=new ArrayList<good>();
```

物品类 good 的实例变量为：

```
String number; //订单号
String time; //到达起点的时间
String destination; //目的地
String road; //路径
boolean isVip; //是否为高级用户
```

(2) 物流节点信息

物流节点信息分为物流节点名称以及节点之间的关系图。

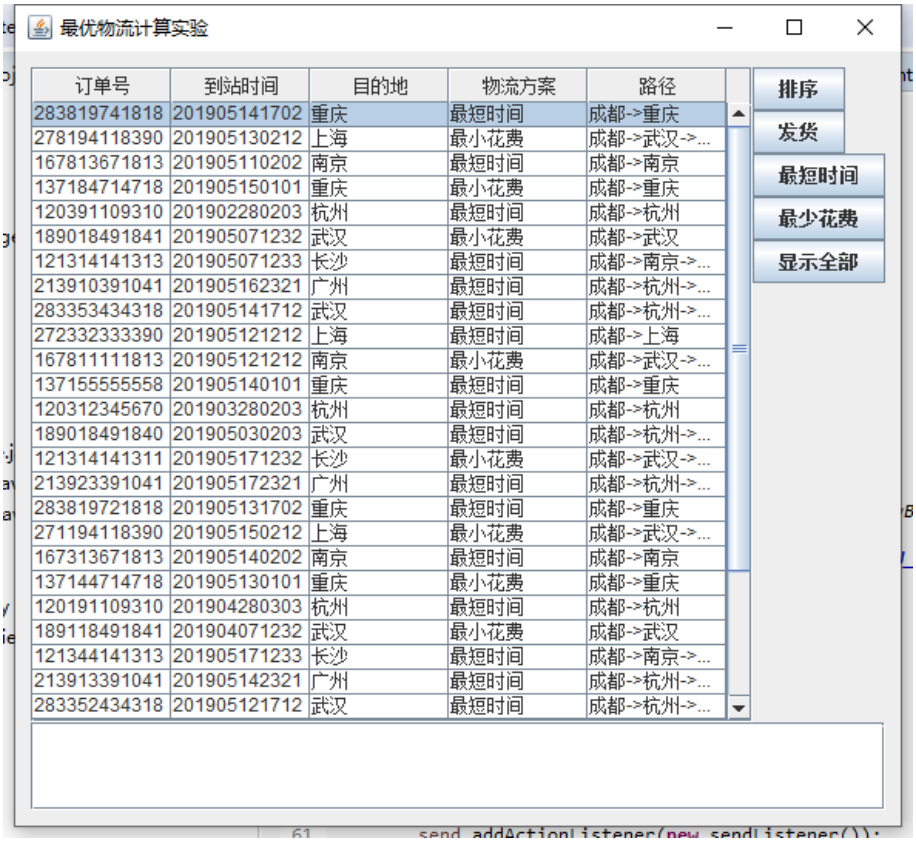
物流节点名称(String)从文件中读取后，存储在 ArrayList<String>中：

```
ArrayList<String> names=new ArrayList<String>();
```

节点之间的关系图以邻接矩阵的形式表示，从文件中取读后存储在二维数组 int[][]cost 中：

```
int[][] cost=new int[cityNum][cityNum];
```

4.5 系统实现



根据时间或者花费进行排序，将最优的物品发货。

5. 实验二：多进程多用户文件一致性读写访问设计实现

5.1 主要内容

实现多用户多进程环境下文件中数据记录的读写、查询操作。主要实验内容如下。

(1)设计实现数据表的文件存储方式，能在文件中存储多张数据表，每张数据表可存储多条记录。实现指定表中记录的存储、读写和记录的简单查询与索引查询函数。能够实现单用户和进程对文件数据中记录的写入与查询。

(2)实现多进程对单个文件中某表中的记录的互斥写入与查询访问操作，保证表中记录数据的一致性。

(3)实现多用户对文件中记录数据的同时写入与查询一致性操作。

5.2 程序设计思路

(1) 数据表文件利用 MySQL 存储在数据库中。使用 MySQL 创建数据库（wuliu），创建数据表（goodslist1、goodlist2）。导入实验一中物品信息的数据。

(2) 通过 MySQL 和 java 之间的 connector 将数据表导入 java 程序。通过 SQL 语句实现对指定表中记录的存储、读写和记录的简单查询与索引查询，编写对应的函数，进而实现单用户和进程对文件数据中记录的写入与查询。

(3) 设置读、写信号量。若有进程读取，则申请一个读信号量；若有进程写入，则申请一个写信号量。只有当读写信号量同时可申请时才可以写入，只有当写信号量可申请时才可以读取。进程结束时释放信号量。

5.3 算法分析与设计

(1) 连接到数据库以及执行 SQL 语句

具体内容见示例（Java）-Java 与数据库连接

(2) 存储、读写和记录的简单查询与索引查询函数

a) 读取

读取指定表中的记录，使用 SELECT FROM 语句，根据 columnNames 分别读取各行的数据。

读取后将物品信息存储在 ArrayList<good> goods 中。

```
    void read() {  //读取数据表，并将数据表中的数据存入ArrayList<good> goods中
        try {
              Connection connect=
DriverManager.getConnection("jdbc:mysql://localhost:3306/wuliu?useSSL=false
", "root", "123456");
              Statement stmt = connect.createStatement();
              ResultSet rs = stmt.executeQuery("select * from
"+client.tablename);
             while (rs.next()) {
               good a=new
good(rs.getString("OrderNumber"),rs.getString("AriveTime"),
                        rs.getString("Destination"),rs.getString("isVip"));
              goods.add(a);
             }
             connect.close();
             }
           catch (Exception e) {e.printStackTrace();}
    }
```

b) 写入

向指定表中写入记录，使用 INSERT INTO tablename VALUES(value1，value2，……)语句。

```
    void write(String ordernumber,String arivetime,String
destination,String isVip) {    //写入数据库的函数
        try {
              Connection connect=
DriverManager.getConnection("jdbc:mysql://localhost:3306/wuliu?useSSL=false
", "root", "123456");
              Statement stmt = connect.createStatement();
              String sql="insert into "+client.tablename+"
values('"+ordernumber+"','"+arivetime+"','"+destination+"','"+isVip+"','fal
se')";
              System.out.println(sql);
              stmt.executeUpdate(sql);
              JOptionPane.showMessageDialog(null, "写入数据库成功");
              connect.close();
            }
           catch (Exception e) {
            JOptionPane.showMessageDialog(null, "写入数据库失败", "ERROR",
JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();
           }
```

```
    }
```

c) 查询

查询指定表中所有列中的关键字。需要关键字（String key）。通过 SELECT FROM WHERELIKE 语句实现。

```
void search(String key) {  //简单查询，查询表中所有列含有key的行，存入goods中
        try {
             Connection connect=
DriverManager.getConnection("jdbc:mysql://localhost:3306/wuliu?useSSL=false
", "root", "123456");
             String sql="select * from "+client.tablename+" where
OrderNumber like '"+"%"+key+
                  "%"+"' or AriveTime like '%"+key+"%' or Destination like
'%"+key+"%' or isVip like '%"+key+"%' ";
             PreparedStatement pst = null;
             pst = (PreparedStatement) connect.prepareStatement(sql);
             ResultSet rs = pst.executeQuery();
               while (rs.next()) {
                good a=new
good(rs.getString("OrderNumber"),rs.getString("AriveTime"),
                        rs.getString("Destination"),rs.getString("isVip"));
               goods.add(a);
               }
               connect.close();
           }
           catch (Exception e) {e.printStackTrace();}
    }
```

d) 索引查询

查询指定表中某一列下的关键字。需要列的名称（String columnname）以及关键字（Stringkey）。通过 SELECT FROM WHERE 语句实现。

```
    void index(String columnname,String key) {//索引查询，查询表中某一列含key
的行，存入goods中
        try {
             Connection connect=
DriverManager.getConnection("jdbc:mysql://localhost:3306/wuliu?useSSL=false
", "root", "123456");
             Statement stmt = connect.createStatement();
             ResultSet rs = stmt.executeQuery("select * from
"+client.tablename+" where "+columnname+" like '%"+key+"%'");
```

```
            while (rs.next()) {
                good a=new
good(rs.getString("OrderNumber"),rs.getString("AriveTime"),rs.getString("De
stination"),rs.getString("isVip"));
                goods.add(a);
            }
            connect.close();
        }
        catch (Exception e) {e.printStackTrace();}
    }
```

e) 多进程互斥读写一致性

设置读写信号量：

```
public static boolean isReading=false;//信号量，true表示有进程正在读取
public static boolean isWriting=false;//信号量，true 表示有进程正在写入
```

当创建一个写进程时，需满足没有进程正在读写才可创建，否则弹窗：

```
if((!isReading)&&(!isWriting)) {
            new writeTable();
            isWriting=true;
        }
        else {
            JOptionPane.showMessageDialog(null, "其他进程正在读/写",
"ERROR", JOptionPane.ERROR_MESSAGE);
        }
```

当创建一个读进程时，需满足没有进程正在写入才可创建，否则弹窗；

```
if(!isWriting) {
            new readTable();
            isReading=true;
        }
        else {
            JOptionPane.showMessageDialog(null, "其他进程正在写入",
"ERROR", JOptionPane.ERROR_MESSAGE);
        }
```

关闭读写进程时，释放信号量：

```
f.addWindowListener(new WindowAdapter() { //关闭窗口时，isWriting=false，写入
进程结束
        public void windowClosing(WindowEvent e) {
        super.windowClosing(e);
        client.isWriting=false;
         }
```

```
            }

frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
            super.windowClosing(e);
            client.isReading=false;
             }
            }
```

f) 创建数据表

```
        public void actionPerformed(ActionEvent e) {
            try {
                  Connection connect=
DriverManager.getConnection("jdbc:mysql://localhost:3306/wuliu?useSSL=false
", "root", "123456");
                Statement stmt = connect.createStatement();
                String sql="CREATE TABLE " +name.getText() +
                          " (" +
                          "OrderNumber varchar(255)," +
                          "AriveTime varchar(255)," +
                          "Destination varchar(255)," +
                          "isVip varchar(255)" +
                          ");";
                stmt.executeUpdate(sql);
```

5.4 数据结构设计

物品信息：

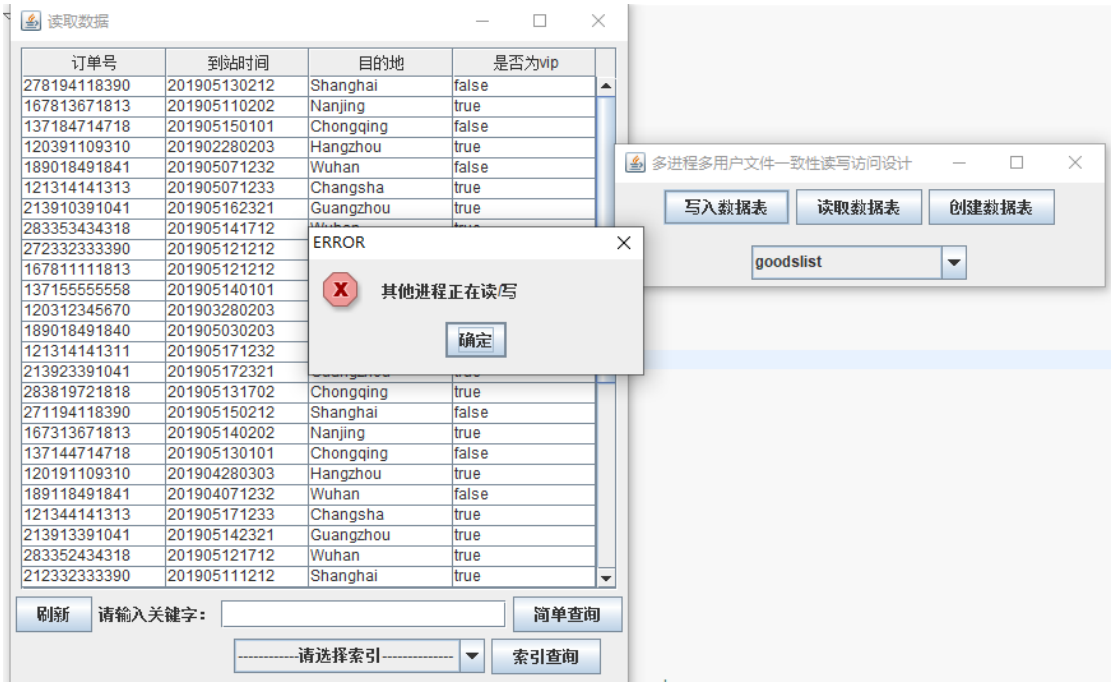
物品信息存储在 MySQL, wuliu 数据库中的 goodslist1 表中，四个 column 分别为：OrderNumber，AriveTime，Destination, isVip，isSend 均为 varchar 格式。使用 Navicat 查看 goodslist 表，如下图

对象 goodslist @wuliu (test) - 表

开始事务 备注 筛选 排序 导入 导出

OrderNumber	AriveTime	Destination	isVip	isSend
278194118390	201905130212	Shanghai	false	true
167813671813	201905110202	Nanjing	true	true
137184714718	201905150101	Chongqing	false	true
120391109310	201902280203	Hangzhou	true	false
189018491841	201905071232	Wuhan	false	false
121314141313	201905071233	Changsha	true	false
213910391041	201905162321	Guangzhou	true	false
283353434318	201905141712	Wuhan	true	false

5.5 系统实现



在主界面可以进行数据表的创建、写数据、读数据。在读/写 goodslist 表时，读/写数据会出现读写互斥。查询表时，可以根据索引进行数据的查询。

6. 实验三：SQL 解析器设计实现

6.1 主要内容

实现部分SQL 语句，包括Select 语句，Insert 语句和Update，创建

表语句的解析并对接实验2中对应的创建表、写入和查询函数实现数据表创建、写入和查询操作。

(1) 构建语法解析器实现部分SQL 语句，包括Select 语句，Insert 语句和Update，创建表语句的语法解析。

(2) 构建语义解析器对SQL 语句进行语义解析

(3) 将解析的语义对接底层实验2 中实现的各个数据操作函数

6.2 程序设计思路

(1) 将输入的字符串以“;”分割成短的字符串，存入队列Queue。

(2) 取队首字符串，对其进行词法分析，将字符串变为小写（SQL不区分大小写），除去多余的空格，将其中的关键字、标识符、常数、界符以及运算符等进行编码和分类，并按照顺序存储在ArrayList 中。

(3) 对ArrayList中的tokens进行语法分析，将token从左到右逐个匹配，如果能找到一条路线完匹配tokens，则完成语法分析，调用相对应的函数。

(4) 回到(2)，直到队列为空。

6.3 算法分析与设计

(1) 预处理

将输入的字符串全部转换为小写（SQL语句不区分大小写），并以“；”分割，存入sentences队列。

```
public void parse(String str) {
    str=str.toLowerCase();
    String[] sentence=str.split(";");
    for(int i=0;i<sentence.length;i++) {
        sentences.offer(sentence[i]);
    }
```

(2) 词法分析

逐个获取字符串中的字符，跳过空格，若为单个字符则判断其是否为“*”、“=”和分隔符号，若为多个字母或数字则将其连接起来，并判断其组合是标识符、常数还是关键字，并将这些tokens对应。直到读完整个字符串：

```
while(p<sentence.length()) {
    getChar();
    skipSpace();
    if(isLetter()) {
        while(isLetter()||isDigit()) {
            link();
            getChar();
        }
        back();
        if(isKeyWord()>0)
            insertKeyWords(token);
        else if(isDataType()>0)
            insertDataType(token);
        else
            insertId(token);
        token="";
    }
    else if(isDigit()) {
        while(isDigit()) {
            link();
            getChar();
        }
        back();
        insertConst(token);
        token="";
    }
    else if(isStar()>0) {
        insertStar(ch);
    }
    else if(isEqual()>0) {
        insertEqual(ch);
    }
    else if(isSeparator()>0) {
        insertSeparators(ch);
    }
```

1	select
2	from
3	insert
4	into
5	values
6	update
7	set
8	where
9	create
10	table
11	=
12	,
13	;
14	(
15	)
16	\'
17	*
18	int
19	varchar
20	id
21	num

Tokens按照读取顺序存入：

```
public static ArrayList<word> words =new ArrayList<word>();
```

其中，word的数据结构为：

```
public class word{
    public String name;
    public int type;
```

(3) 语法分析

Token流存储在words中后，sentenceAnalyze类中的checkSentence会对token流进行语法分析。这里使用的是递归下降分析法，这一过程基于match()函数实现，Match函数对每个token进行正则匹配：

```
boolean match(String regex) {//正则表达式
    String token=wordAnalyze.words.get(p).name;//token
    if(isMatch(regex,token)) {
        p++;
        return true;
    }
    else
        return false;
}
```

其中，regex为正则表达式，p为int类型的index，初始值为0，若token匹配正则表达式，则p自增，return true，否则return false。与match函数同理的还有用于匹配标识符的id()函数，用于匹配一个或多个以“,”分隔的标识符的idList()函数等。在上述匹配函数的基础上，checkSentence函数得以实现。以SELECT语句为例，使用一系列

的match函数对token流进行匹配。

```
public void checkSentence()  {
    if(match("select")) {
        if(match("\\*")&&match("from")&&id()) {
            connect();
            readAll(wordAnalyze.words.get(p-1).name);
        }
        else if(idList()&&match("from")&&id()) {
            connect();
            String[] column=new String[list.size()];
            for(int i=0;i<list.size();i++) {
                column[i]=list.get(i);
            }
            read(column,wordAnalyze.words.get(p-1).name);
        }
    }
```

其中，readAll会读取指定表中所有列的数据，而read则会根据从idList中传入的列读取这些列中的数据。两个函数都会连接到数据库，对数据库中的数据进行操作。UPDATE、DELETE、INSERT与SELECT同理，此处不再赘述。若无匹配，则会输出“grammar error”。

根据类型对标识符、常数、数据类型进行匹配：

```
boolean id() {
boolean num() {
boolean dataType() {
```

一个或者多个以逗号分隔的标识符：

```
boolean idList() {
    list=new ArrayList<String>();//
    if(id()) {
        list.add(wordAnalyze.words.get(p-1).name);
        while(match(",")&&id()) {
            list.add(wordAnalyze.words.get(p-1).name);
        }
        return true;

    }else
        return false;
}
```

（List类的boolean函数同理，此处不再赘述。）

对token流进行检查，如果:

```
if(match("select")) {
    if(match("\\*")&&match("from")&&id()) {
        connect();
        readAll(wordAnalyze.words.get(p-1).name);
    }
    else if(idList()&&match("from")&&id()) {
        connect();
        String[] column=new String[list.size()];
        for(int i=0;i<list.size();i++) {
            column[i]=list.get(i);
        }
        read(column,wordAnalyze.words.get(p-1).name);
```

执行对应的read函数。

```
else if(match("insert")&&match("into")&&id()&&match
        ("values")&&match("\\(")&&valueList()&&match("\\)"))
```

执行write函数。

```
else if(match("create")&&match("table")&&id()
        &&match("\\(")&&columnList()&&match("\\)")) {
```

执行创建表createTable函数。

```
else if(match("update")&&id()&&match("set")&&setList()&&
        match("where")&&id()&&match("=")&&match("'")&&id()&&match("'")) {//
```

执行对应的updateTable函数。

6.4 数据结构设计

(1) Token

每个token都有名字String name以及类型int type

```
public class word{
    public String name;
    public int type;
    word(){}
    word(String name,int type){
        this.name=name;
        this.type=type;
    }
}
```

(2) Sentences队列

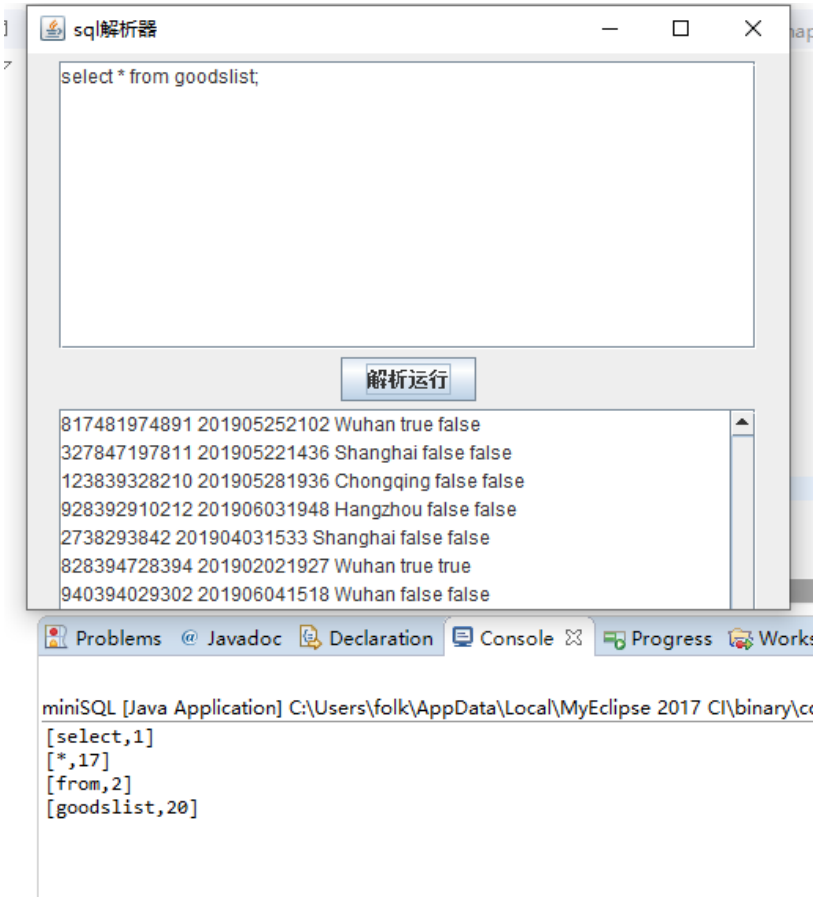
断句后，句子字符串按照顺序存储在sentences队列中

```
Queue<String> sentences = new LinkedList<String>();
```

每次取队首句子进行词法分析以及语法分析，直到队空：

```
while(!sentences.isEmpty()){
    String stc=sentences.poll();
    wordAnalyze wa=new wordAnalyze();
    wordAnalyze.p=0;
    wordAnalyze.sentence=stc;
    wa.analyze();
    sentenceAnalyze sa=new sentenceAnalyze();
    sentenceAnalyze.p=0;
    sa.checkSentence();
    result.append("----------------------------\n");
    }
```

6.5 系统实现



用户在界面上方的输入框内输入单个 SQL 语句（包括 SELECT、INSERT、UPDATE、DELETE 语句），或者输入多个 SQL 语句，以“;”间隔，点击“解析运行”按钮，sql 解析器将对输入的语句进行解析，若语法没有错误则会执行对应的增删查改操作。例，输入 select * from goodslist;查询语句，在控制台有词法分析结果，客户端有查询结果。插入、建表、更新语句同 select 语句。

7. 实验四：互联网+智慧物流质询系统设计实现

7.1 主要内容

实现智慧物流质询系统，系统具体要求如下。

(1)结合实验 1,2 构建物流节点信息表，实现物流节点信息的数据库存储与多进程多用户底层操作

(2) 结合实验 1,2，构建物品信息表，实现物品信息的存储与多进程多用户底层操作

(3) 结合实验 3 以 SQL 语句，对物流节点信息进行增删改查

(4) 结合实验 3 以 SQL 语句，对物品信息进行增删改查

(5) 结合实验 1，实现物品的优先级排序和物流方案分类（可采用多种分类方法不一定用树结构存储）

(6) 节点信息会动态变化，因此结合实验 1，每个物品需要动态计算物流最短路径的实现。

(7)物品的物流状态，用户可以对物件的物流状态进行查询。

7.2 主要功能

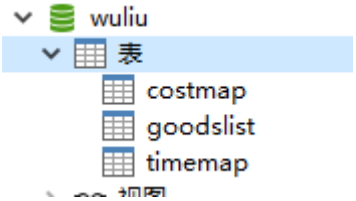
(1) 打开 SQL 解析器，通过 SQL 语句对物流节点信息和物品信息进行增删查改。

(2) 打开物品信息表和物流节点信息表，进行底层操作。

(3) 打开物流状态查询界面，查询物流状态。

7.3 数据库设计

根据系统设计和业务逻辑分析，建立以下数据表：



其中 goodslist 表用于保存物品信息。结构如下：

名	类型	长度	注释
OrderNumber	varchar	50	订单号
AriveTime	varchar	50	到站时间
Destination	varchar	50	目的地
isVip	varchar	50	是否为高级用户
isSend	varchar	50	是否已经发货

costmap 表用于保存物流节点之间运输花费的信息，结构如下：

名	类型	长度	注释
Chengdu	varchar	255	到成都的花费
Chongqing	varchar	255	到重庆的花费
Changsha	varchar	255	到长沙的花费
Wuhan	varchar	255	到武汉的花费
Guangzhou	varchar	255	到广州的花费
Hangzhou	varchar	255	到杭州的花费
Nanjing	varchar	255	到南京的花费
Shanghai	varchar	255	到上海的花费
cityName	varchar	255	物流节点名

timemap 表用于保存物流节点之间运输时间的信息，结构如下

名	类型	长度	注释
Chengdu	varchar	255	到成都的时间
Chongqing	varchar	255	到重庆的时间
Changsha	varchar	255	到长沙的时间
Wuhan	varchar	255	到武汉的时间
Guangzhou	varchar	255	到广州的时间
Hangzhou	varchar	255	到杭州的时间
Nanjing	varchar	255	到南京的时间
Shanghai	varchar	255	到上海的时间
cityName	varchar	255	物流节点名

7.4 系统分析与设计

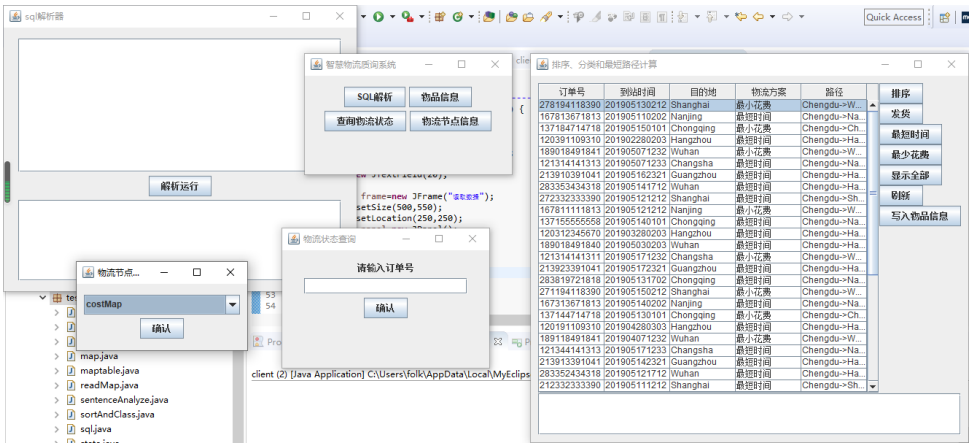
见实验一、实验二、实验三

7.5 系统实现

主界面：


智慧物流质询系统
SQL解析
物品信息
查询物流状态
物流节点信息

其余子模块


解析运行
SQL解析
物品信息
查询物流状态
物流节点信息
costMap
确认
请输入订单号
确认
排序
发货
最短时间
最少花费
显示全部
刷新
写入物品信息
map.java
maptable.java
readMap.java
sentenceAnalyze.java
sortAndClass.java
sql.java