面向对象与多线程综合实验

结合华为云平台的基于JAVASE数据挖掘系统设计与实现

主要内容

一
- 实验目标和要求

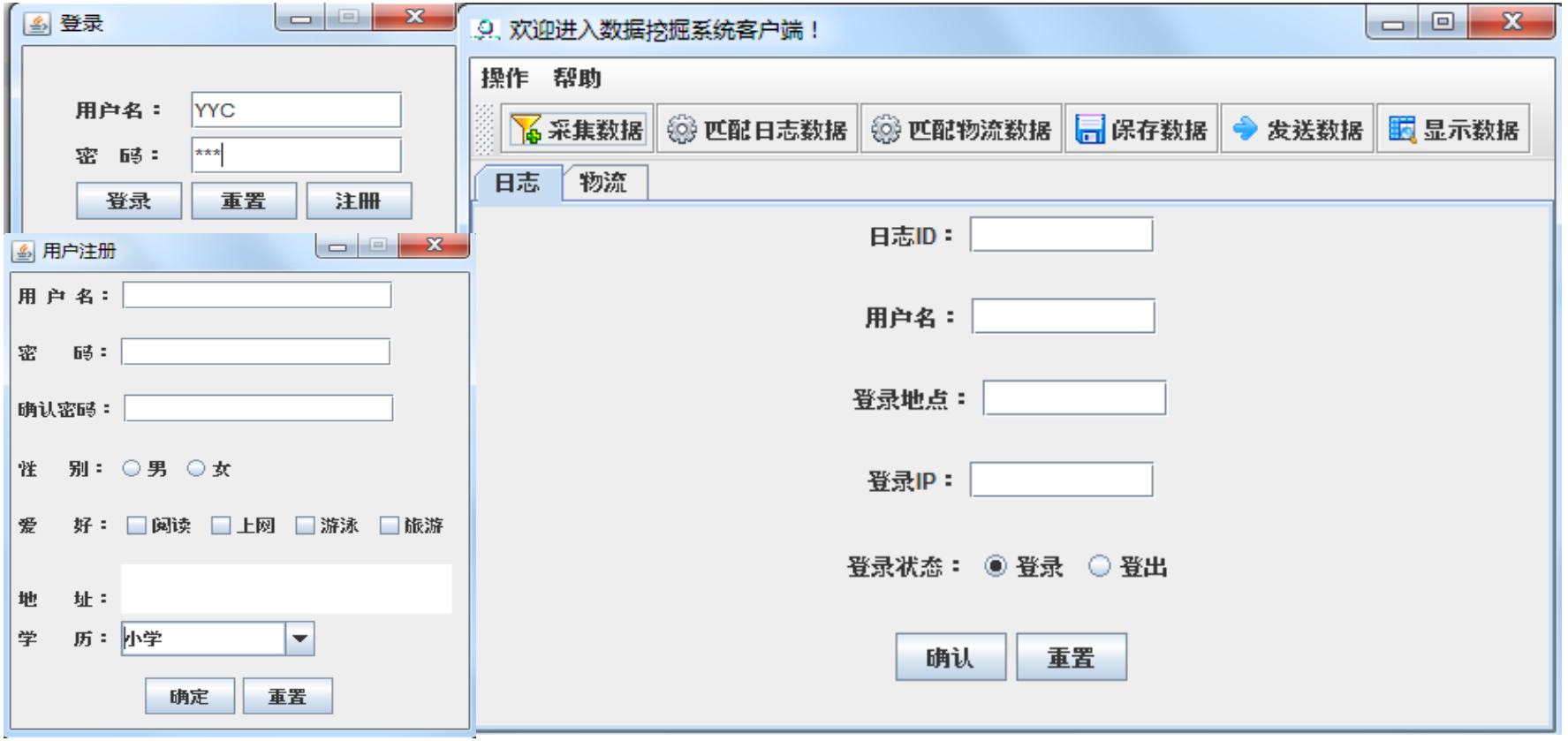
二
- 实验内容

三
- 知识要点

系统简介

- 基于Java SE 的数据挖掘系统

基于客户端服务器端（Client-Server，C-S）模式，实现日志与物流数据信息的采集、匹配、保存、显示等功能，为数据分析挖掘提供基础支撑。



需求分析

- 系统包括客户端应用程序、服务器端应用程序---C/S模式
- 用户和数据信息的保存---JDBC数据库保存和查询
- 用户能够进行注册和登录，授权后使用系统---GUI登录和注册界面设计与功能实现
- 能够实现日志和物流信息的数据采集（录入），登录登出对匹配、信息保存和数据显示等功能---GUI主界面设计与功能实现
- 系统能够进行数据自动刷新功能，与数据库保持同步---线程
- 客户端与服务器端交互，客户端能够将数据发送到服务器端，服务器端接收客户端发送的日志和物流信息，进行保存和处理---Socket通信
- 系统优化---JAVA高级应用与新特性

任务分配

课时分配	实验任务	任务列表	主要知识点分解
第1-2次课	基于控制台的系统数据**采集、匹配、显示**和**记录**功能实现	✓ 注册华为软开云账户 ✓ 搭建数据挖掘系统**框架** ✓ 实现日志和物流数据信息的**采集**、**匹配**和**显示**功能 ✓ 实现匹配的物流数据信息的文件保存和读取**记录**功能	继承与多态/异常处理/集合/文件存储及IO流/华为软开云
第3次课	基于**JDBC**的控制台系统基本功能实现	✓ 创建项目所需的数据库表，并搭建数据访问基础环境 ✓ 实现并测试匹配的日志、物流信息的数据库保存和查询功能	JAVAJDBC/MySQL
第4次课	基于SwingGUI的系统**注册**和**登录界面**设计实现	✓ 创建用户数据库表、用户实体类和用户业务逻辑类 ✓ 创建用户注册窗口，并将用户注册信息保存到数据库 ✓ 创建用户登录窗口，登录成功则进入系统主界面	SwingGUI/事件驱动/WinBuilder
第5次课	基于SwingGUI系统**主界面**设计实现和系统**优化**	✓ 实现主界面中的菜单和工具栏 ✓ 实现主界面中的日志和物流**数据采集**、**匹配**、**保存**和**显示**功能 ✓ GUI系统优化	高级UI组件
第6次课	系统信息**自动刷新**功能实现	✓ 使用线程实现每隔**1分钟(或自定义)**日志和物流显示数据表格**自动刷新**功能，以便与数据库保持同步	线程
第7次课	系统客户端/服务器端数据**发送(交互)**功能实现	✓ 客户端应用程序：修改主界面**发送数据**页面响应，即使用Socket实现数据由客户端发送到服务器 ✓ 服务器端应用程序：使用Server Socket实现接收客户端发送的日志和物流数据信息，并将信息保存到数据库	Socket网络编程
第8次课	系统验收	✓ 数据挖掘系统的基本功能实现与演示 ✓ **在华为软开云平台完成系统需求分析和设计反推**	PPT演示、系统运行
拓展	Java 高级应用以及Java8新特性	◆ 使用注解重新迭代升级系统代码 ◆ 使用格式化将输出的日期进行格式化输出 ◆ 使用Lambda表达式迭代升级主窗口中“帮助”菜单的事件处理 ◆ 使用Lambda表达式实现查找指定的匹配信息并显示	增加注解和格式化以及Lambda优化和查询

功能要求

系统基于C/S模式，包括客户端和服务器端应用程序

- 1. 用户登录和注册功能:用户验证口令通过后登录系统，新用户进行注册，并将注册信息保存数据库
- 2. 日志、物流数据的采集功能：对日志和物流数据进行采集，并保存到Mysql数据库；
- 3. 日志、物流数据的筛选匹配功能，以日志信息为例：
  - 根据日志的登录、登出状态，对日志进行分类，分别存放到**登录**日志集合（ArrayList<LogRec> logIns）和**登出**日志集合（ArrayList<LogRec> logOuts）中。
  - 在登录日志和登出日志中，根据**用户名**和IP地址进行匹配：如果存在相同的用户名和IP地址，则日志信息匹配成功，将匹配的日志数据封装到MatchedLogRec对象，并保存到匹配日志集合（ArrayList<MatchedLogRec> matchLogs）中。
- 4. 日志、物流数据的数据保存功能：在系统主界面中点击“保存数据”按钮时，将匹配的日志数据和物流数据保存到本地文件和数据库中。
- 5. 客户端服务器端交互功能：
  - ✓ 客户端的数据发送功能：在客户端通过Socket技术向服务器端发送匹配的日志数据和物流数据；当数据发送成功后，清空客户端暂时存放数据的集合，然后弹出信息提醒；
  - ✓ 服务器数据查询功能：当点击客户端显示数据功能时，服务器端从数据库中查找符合条件的数据，并发送到客户端；在客户端以表格的形式将数据显示。

实验目标和要求

实验目标

1. 理解与掌握知识点：
   - ◆ 掌握JAVA图形用户界面(GUI)的基本结构
   - ◆ 掌握JAVA Swing基础组件和部分高级组件的使用及布局管理
   - ◆ 掌握事件处理机制
2. 熟悉工具与平台：Windowbuilder插件完成GUI界面开发
3. 开发程序：SwingGUI开发，将数据挖掘系统控制台界面改为GUI图形用户界面，实现系统的用户注册、登录和主窗口界面及功能。

实验要求与交付

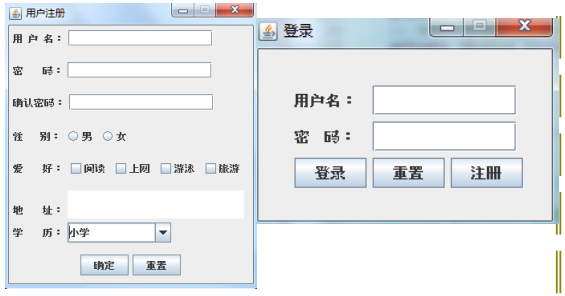
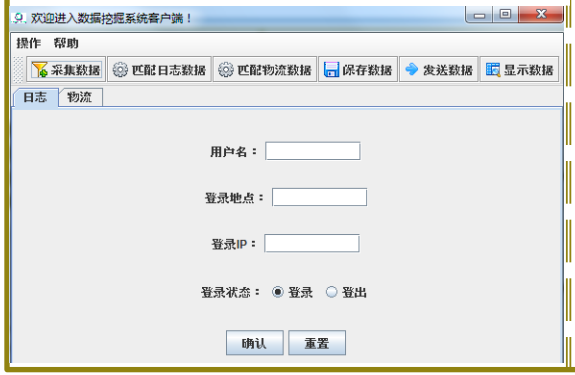
要求：

1. Windowbuilder插件实现用户注册、登录和主窗口界面开发及功能实现
2. 界面开发中选取合适的布局、综合使用按钮、文本框、对话框等组件，实现友好的人机交互界面和响应

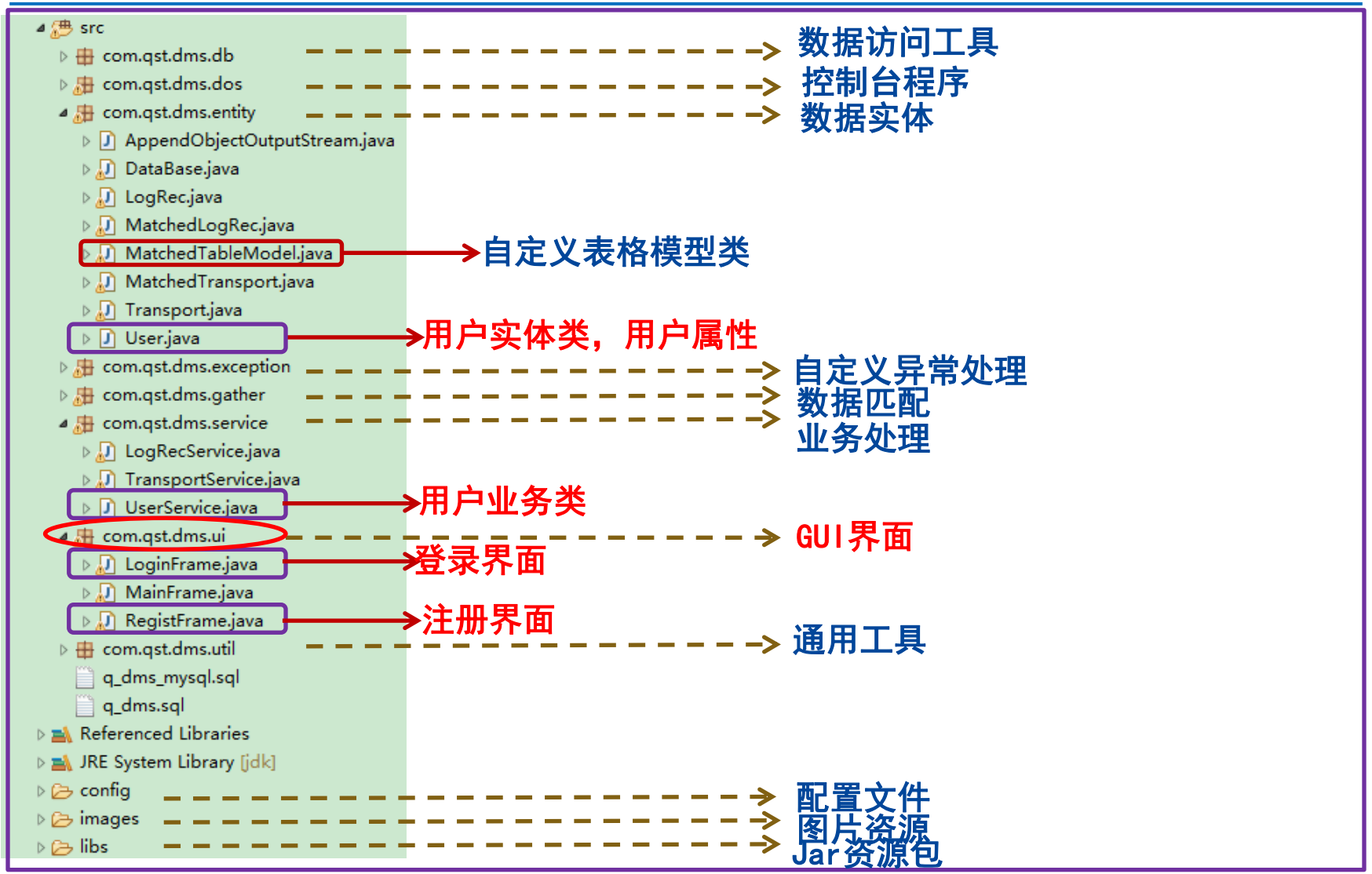
✱交付：

1. 代码与运行结果
2. 开发过程中遇到问题截图和解决办法

实验内容

1. 做什么	2. 怎么做？	3. 对应项
◆ 设计并实现用户登录和注册界面 	◆ 创建用户注册窗口，并将用户注册信息保存到数据库	◆ WindowBuilder插件 ◆ 创建用户数据实体类 entity-user.java ◆ 创建用户操作类 service-》UserService.java ◆ 创建用户注册界面 ui-》RegistFrame.java
	◆ 设计用户登录界面，登录成功则进入系统主界面	◆ 创建用户登录界面 ui-》LoginFrame.java
◆ 设计实现主界面及优化 	◆ 系统主界面及优化： ✓ 主界面中的菜单栏 ✓ 主界面中的工具栏 ■ 数据采集界面及功能 ■ 数据匹配界面及功能 ■ 数据保存界面及功能 ■ 数据显示界面及功能 ✓ 系统优化	◆ 创建自定义表格模型 Entity-MatchedTableModel ◆ 创建主界面 ui-》MainFrame.java

实验内容系统框架


src
com.qst.dms.db
com.qst.dms.dos
com.qst.dms.entity
AppendObjectOutputStream.java
DataBase.java
LogRec.java
MatchedLogRec.java
MatchedTableModel.java
MatchedTransport.java
Transport.java
User.java
com.qst.dms.exception
com.qst.dms.gather
com.qst.dms.service
LogRecService.java
TransportService.java
UserService.java
com.qst.dms.ui
LoginFrame.java
MainFrame.java
RegistFrame.java
com.qst.dms.util
q_dms_mysql.sql
q_dms.sql
Referenced Libraries
JRE System Library [jdk]
config
images
libs
数据访问工具
控制台程序
数据实体
自定义表格模型类
用户实体类，用户属性
自定义异常处理
数据匹配
业务处理
用户业务类
GUI界面
登录界面
注册界面
通用工具
配置文件
图片资源
Jar资源包

实验内容一
用户注册界面设计与实现

实验内容

用户注册界面设计

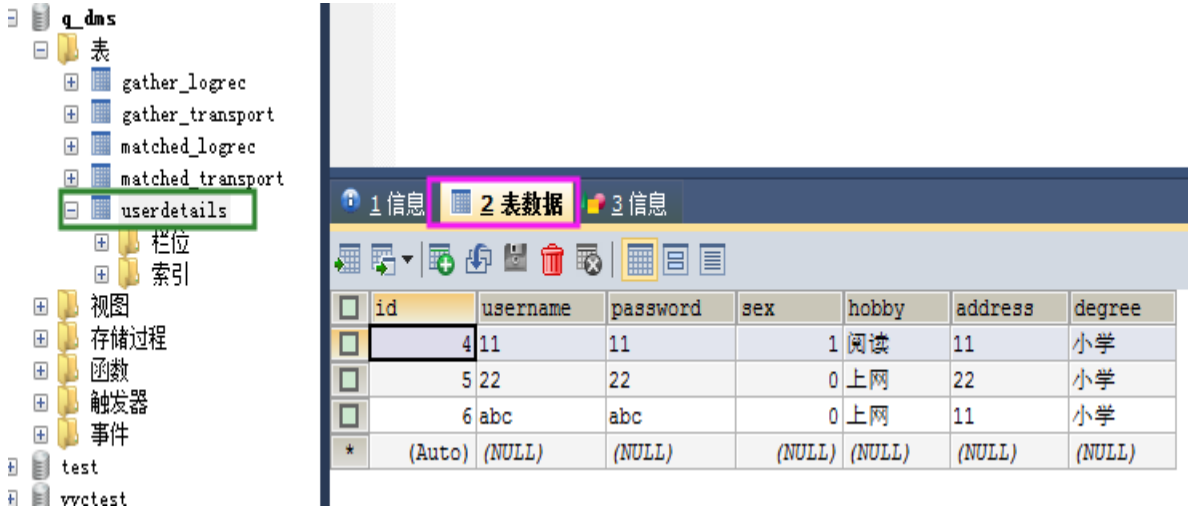
1. 用户实体类和数据表设计
2. WindowBuilder插件辅助开发
3. 用户注册界面组件设计
4. 事件响应

1. 创建用户实体类和用户数据库表

Entity-》User.java

用户数据表userdetails

```java
//用户实体
public class User {
    // 用户id
    private int id;
    // 用户名
    private String username;
    // 密码
    private String password;
    // 性别
    private int sex;
    // 爱好
    private String hobby;
    // 地址
    private String address;
    // 学历
    private String degree;

    public int getId() {
        return id;
    }
```



id	username	password	sex	hobby	address	degree
4	11	11	1	阅读	11	小学
5	22	22	0	上网	22	小学
6	abc	abc	0	上网	11	小学
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

2. WindowBuilder辅助界面开发

JAVA GUI界面开发方式

1. 纯手工编写代码
2. WindowBuilder基于Eclipse平台插件进行GUI双向设计：所见即所得

WindowBuilder安装方式：

1. Eclipse官网在线安装
2. 离线安装

WindowBuilder插件使用过程见课件视频

2. WindowBuilder辅助界面开发

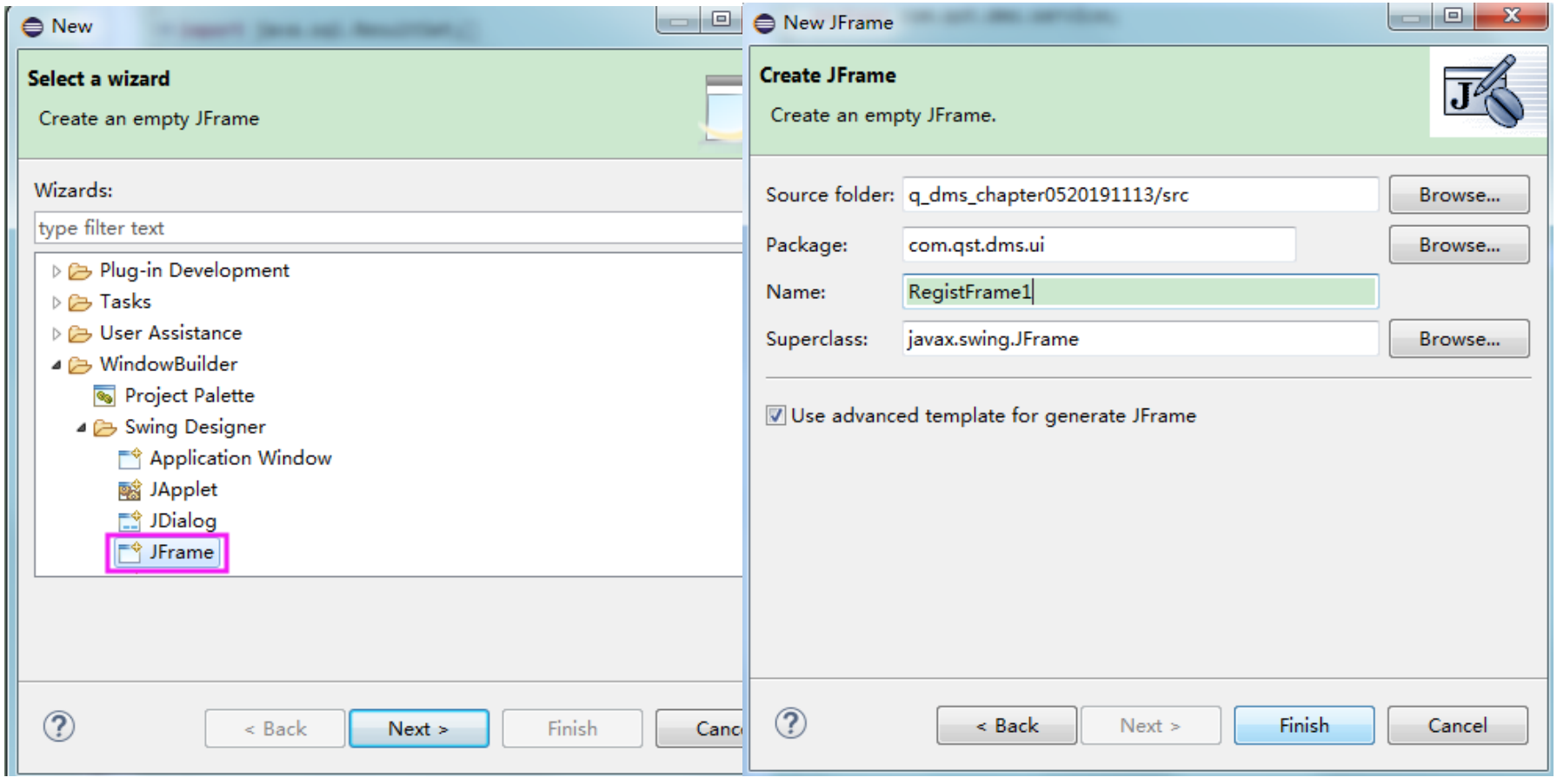
WindowBuilder插件实现用户注册功能步骤：

1. 创建JFrame窗体，并设置窗体属性及布局
2. 添加并设置组件
3. 添加事件处理

创建窗体

使用WindowBuilder新建JFrame窗体：

在Eclipse项目中，单击File-New-Other-WindowBuilder-JFrame，在弹出的对话框中，输入类名，完成JFrame窗体创建。



创建窗体

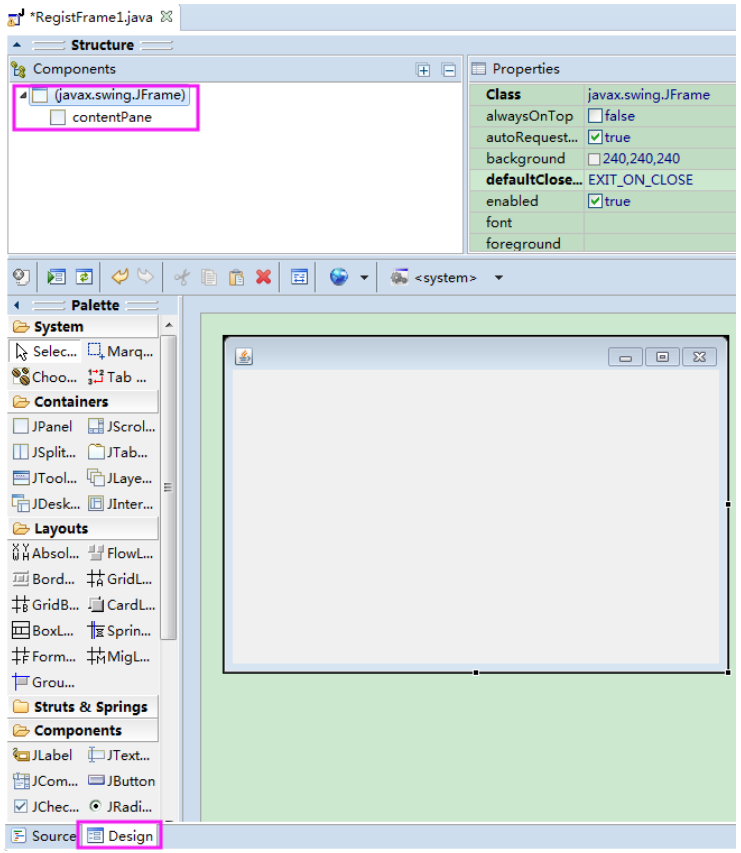
代码编辑窗口有Source和Design选项卡，在源代码和设计界面之间切换。

界面采用拖拽方式加载控件到窗体，自动生成相应Source：所见即所得

默认为BorderLayout，选择Absolute layout，设置窗口的大小，标题，背景等

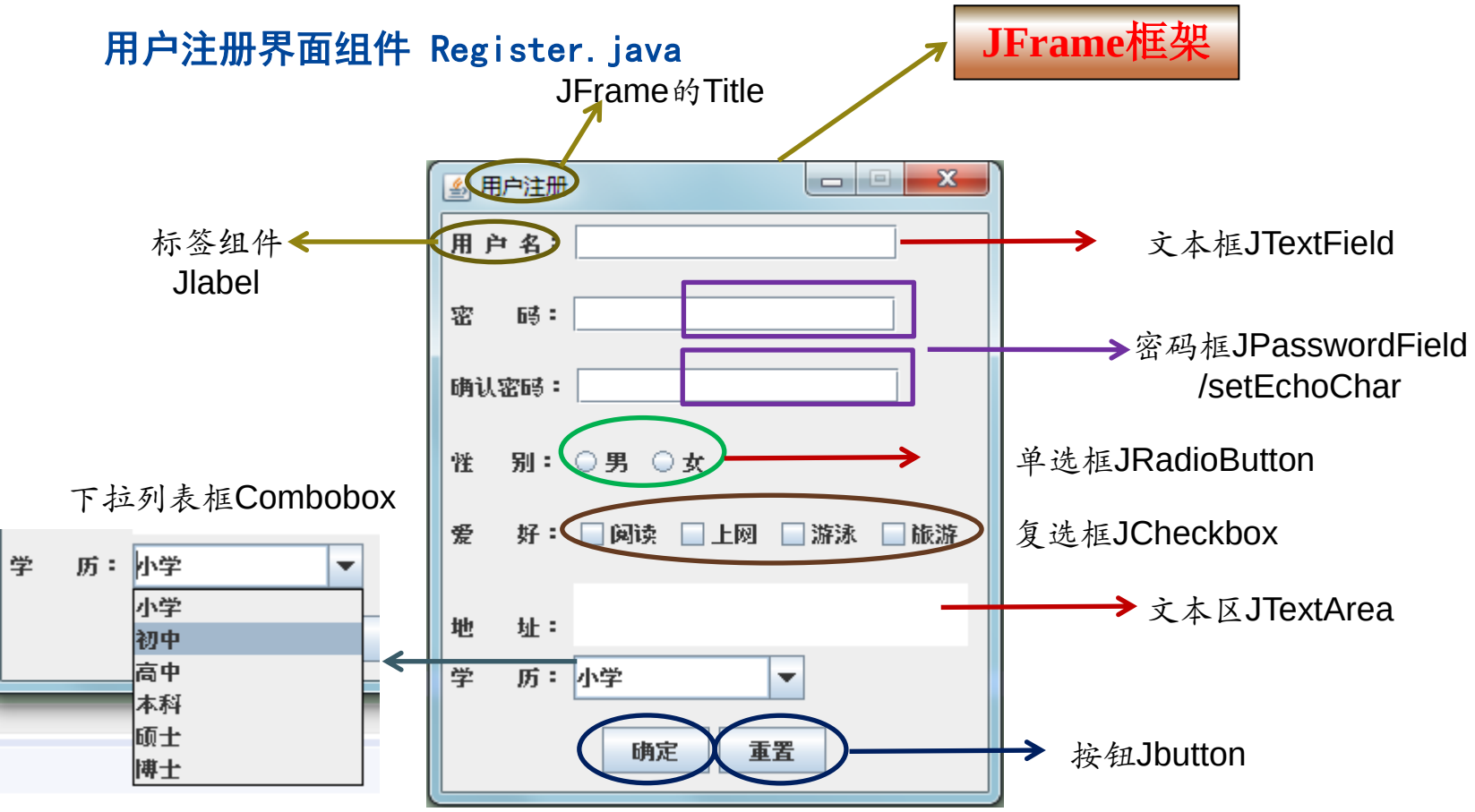
```
1  package com.qst.dms.ui;
2
3  import java.awt.BorderLayout;
9
10 public class RegistFrame1 extends JFrame {
11
12     private JPanel contentPane;
13
14     /**
15      * Launch the application.
16      */
17     public static void main(String[] args) {
18         EventQueue.invokeLater(new Runnable() {
19             public void run() {
20                 try {
21                     RegistFrame1 frame = new RegistFrame1();
22                     frame.setVisible(true);
23                 } catch (Exception e) {
24                     e.printStackTrace();
25                 }
26             }
27         });
28     }
29
30     /**
31      * Create the frame.
32      */
33     public RegistFrame1() {
34         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35         setBounds(100, 100, 450, 300);
36         contentPane = new JPanel();
37         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
38         contentPane.setLayout(new BorderLayout(0, 0));
39         setContentPane(contentPane);
40     }
41
42 }
43
```

Source Design



注册界面组件

用户注册界面组件 Register.java

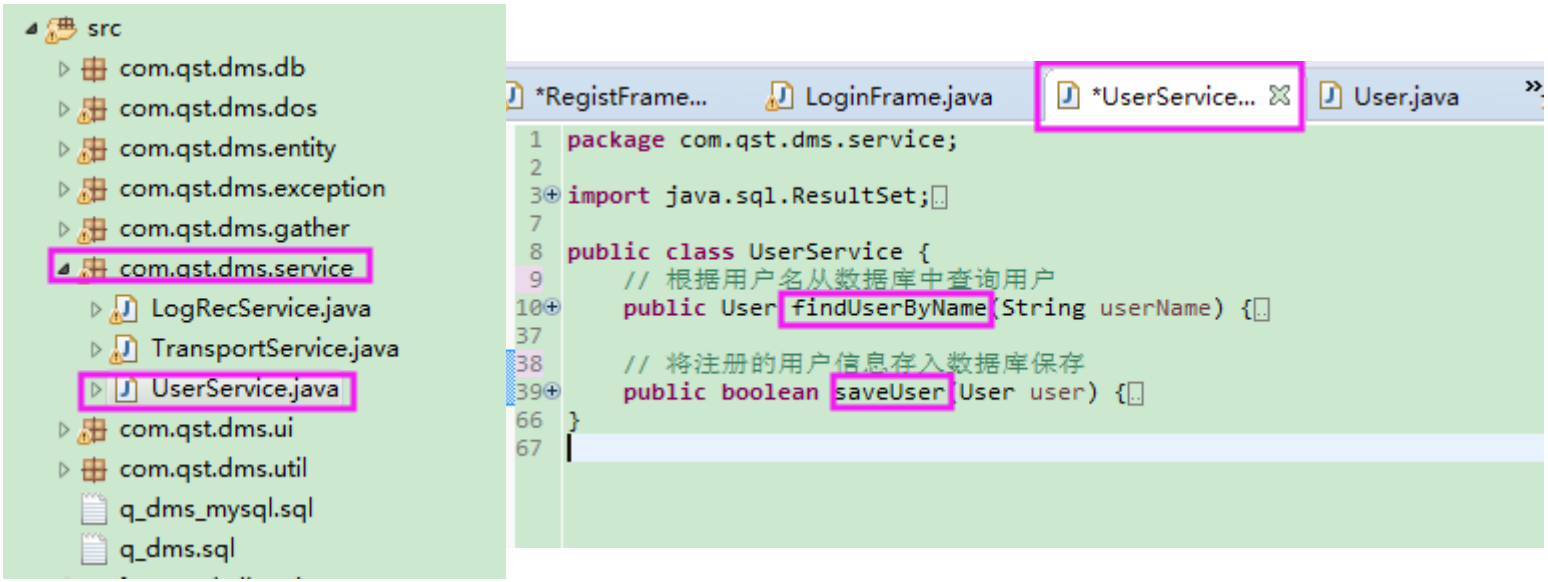


事件响应

用户单击“确定”或“重置”按钮，触发事件

当用户点击确定后，注册的用户信息存入数据库，为确定按钮添加事件处理代码， 代码

当用户点击重置后，相应的输入信息被清除



事件响应

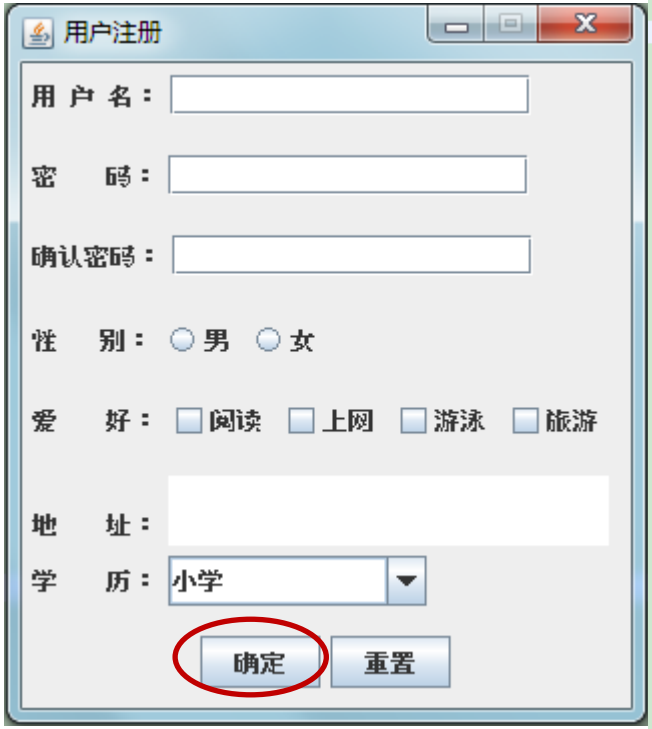
用户单击“确定”或“重置”按钮组件（Source），触发了事件(Event)，并将事件的信息发送给监听者（Listeners），由监听者对象的方法来处理~~

即界面程序响应用户的操作！将代码写在监听者对象的方法里~

当用户点击确定按钮，注册的用户信息存入数据库
当用户点击重置按钮，输入的界面信息被清除

事件响应

- 点击确定按钮触发的事件：将注册信息保存到数据库中saveUser

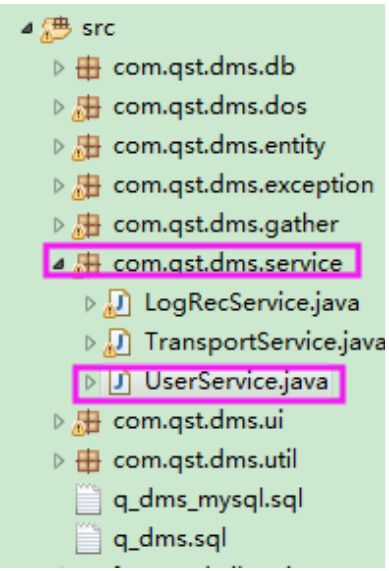



```
btnOk = new JButton("确定");
// 注册监听器，监听确定按钮
btnOk.addActionListener(new RegisterListener());
```

```
// 监听类，负责处理确认按钮的业务逻辑
private class RegisterListener implements ActionListener {
    // 重写actionPerFormed()方法，事件处理方法
    public void actionPerformed(ActionEvent e) {
        // 获取用户输入的数据
        String userName = txtName.getText().trim();
        String password = new String(txtPwd.getPassword());
        String rePassword = new String(txtRePwd.getPassword());
        // 将性别"男""女"对应转化为"1""0"
        int sex = Integer.parseInt(rbFemale.isSelected() ? "0" : "1");
        String hobby = (ckbRead.isSelected() ? "阅读" : "")
                + (ckbNet.isSelected() ? "上网" : "")
                + (ckbSwim.isSelected() ? "游泳" : "")
                + (ckbTour.isSelected() ? "旅游" : "");
        String address = txtAdress.getText().trim();
        String degree = cmbDegree.getSelectedItem().toString().trim();
        // 判断两次输入密码是否一致
        if (password.equals(rePassword)) {
            // 将数据封装到对象中
            user = new User(userName, password, sex, hobby, address, degree);
            // 保存数据
            if (userService.saveUser(user)) {
                // 输出提示信息
                //System.out.println("注册成功！");
                JOptionPane.showMessageDialog(null,"注册成功！","成功提示",JOptionPane.PLAIN_MESSAGE);
            } else {
                // 输出提示信息
                //System.out.println("注册失败！");
                JOptionPane.showMessageDialog(null,"注册失败！","错误提示",JOptionPane.ERROR_MESSAGE);
            }
        } else {
            // 输出提示信息
            //System.out.println("两次输入的密码不一致！");
            JOptionPane.showMessageDialog(null,"两次输入的密码不一致！","错误提示",JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

事件响应

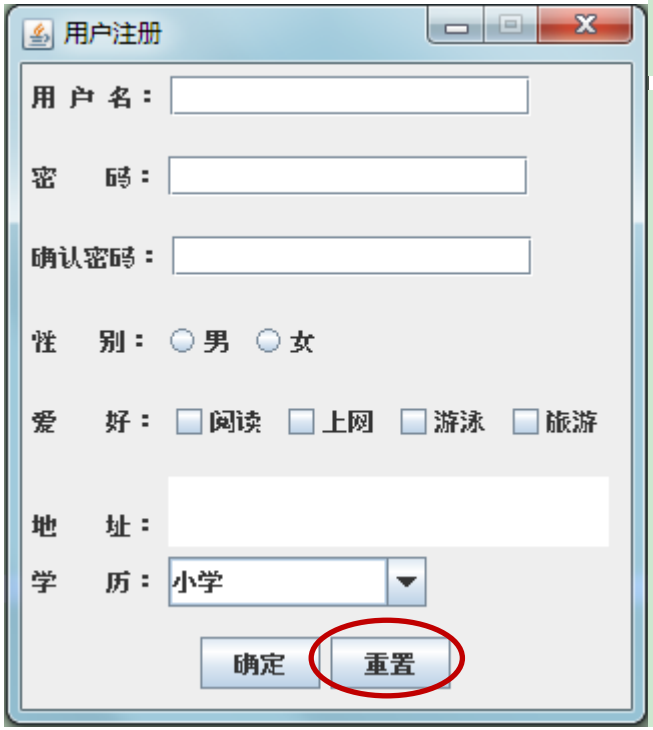
- 用户信息存入数据库： saveUser




```java
// 保存用户信息
public boolean saveUser(User user) {
    // 定义一个布尔返回值，初始值为false
    boolean r = false;
    DBUtil db = new DBUtil();
    try {
        // 获取数据库连接
        db.getConnection();
        //start revised by yyc 20191126
        String sqltest = "SELECT * FROM userdetails WHERE username='"+user.getUsername()+"'";
        ResultSet rs = db.executeQuery(sqltest,null);
        if(rs.next())
            return r;
        //end
            // 使用PreparedStatement发送sql语句
        String sql = "INSERT INTO userdetails(username,password,sex,hobby,address,degree) VALUES (?,?,?,?,?,?)";
        // 设置参数
        Object[] param = new Object[] { user.getUsername(),
                user.getPassword(), user.getSex(), user.getHobby(),
                user.getAddress(), user.getDegree() };
        // 判断数据是否保存成功
        if (db.executeUpdate(sql, param) > 0) {
            // 保存成功，设置返回值为true
            r = true;
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        // 关闭数据库的连接
        db.closeAll();
    }
    // 返回
```

事件响应

- 点击重置按钮触发的事件：清空填写内容，重置选择按钮

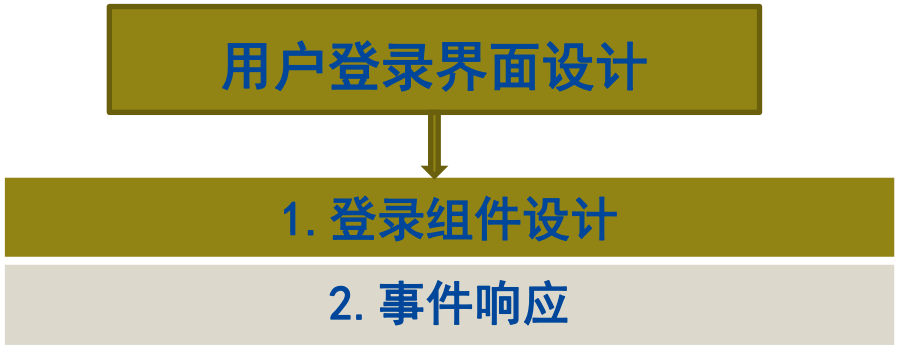



```
btnCancle = new JButton("重置");
// 注册监听器，监听重置按钮
btnCancle.addActionListener(new ResetListener());
```

```
// 监听类，负责处理重置按钮
public class ResetListener implements ActionListener {
    // 重写actionPerFormed()方法，重置组件内容事件处理方法
    public void actionPerformed(ActionEvent e) {
        // 清空姓名、密码、确认密码内容
        txtName.setText("");
        txtPwd.setText("");
        txtRePwd.setText("");
        // 重置单选按钮为未选择
        rbMale.setSelected(false);
        rbFemale.setSelected(false);
        // 重置所有的复选按钮为未选择
        ckbRead.setSelected(false);
        ckbNet.setSelected(false);
        ckbSwim.setSelected(false);
        ckbTour.setSelected(false);
        // 清空地址栏
        txtAdress.setText("");
        // 重置组合框为未选择状态
        cmbDegree.setSelectedIndex(0);
    }
}
```

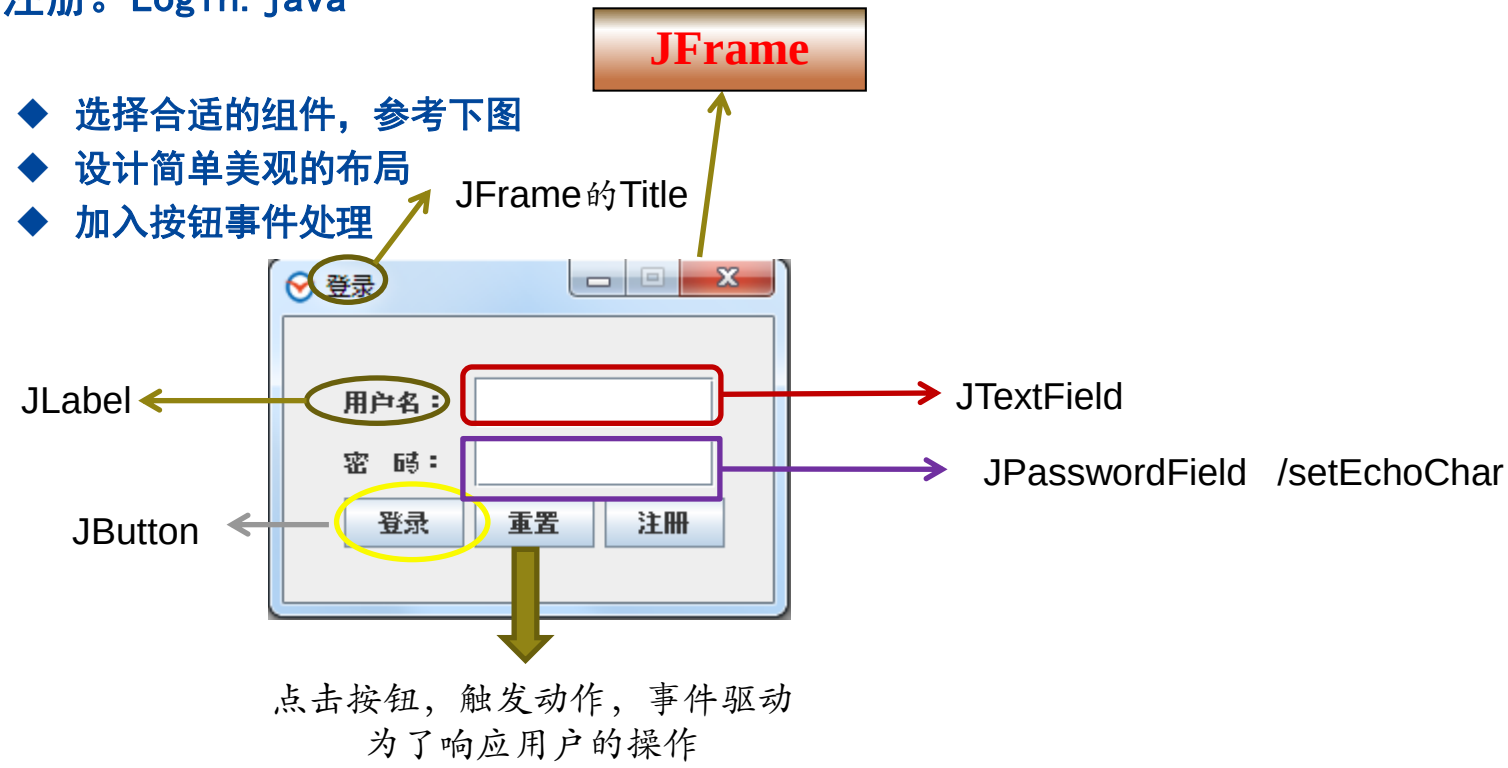
实验内容二
用户登录界面设计与实现

实验内容



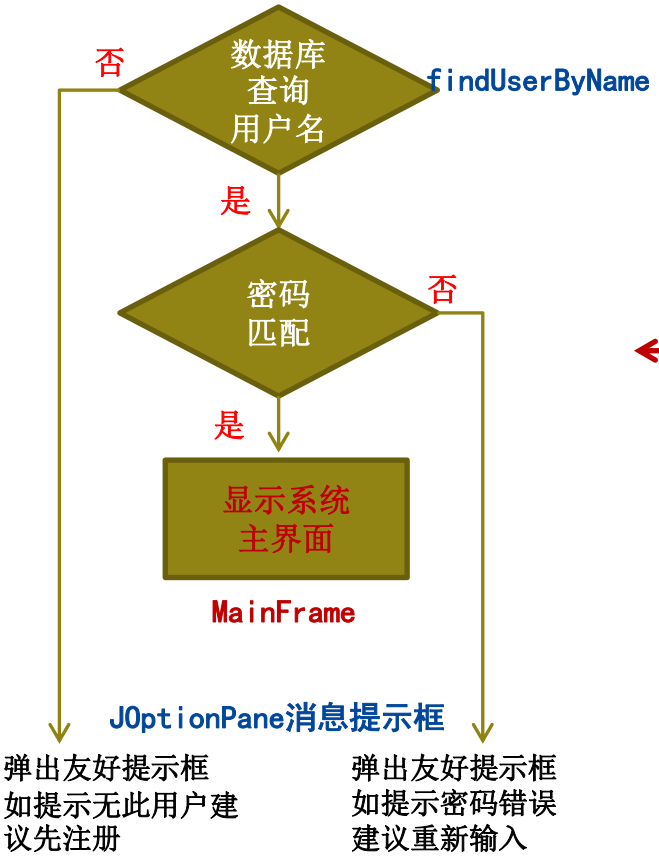
用户登录界面

- 自行设计用户登录界面，点击登录按钮，和数据库信息进行匹配，匹配成功后进入主界面；点击重置，还可进行用户信息重新填写；点击注册按钮，跳转到注册界面进行注册。Login. java



用户登录界面

➢ 用户登录界面事件



实验内容三
系统主界面设计与实现

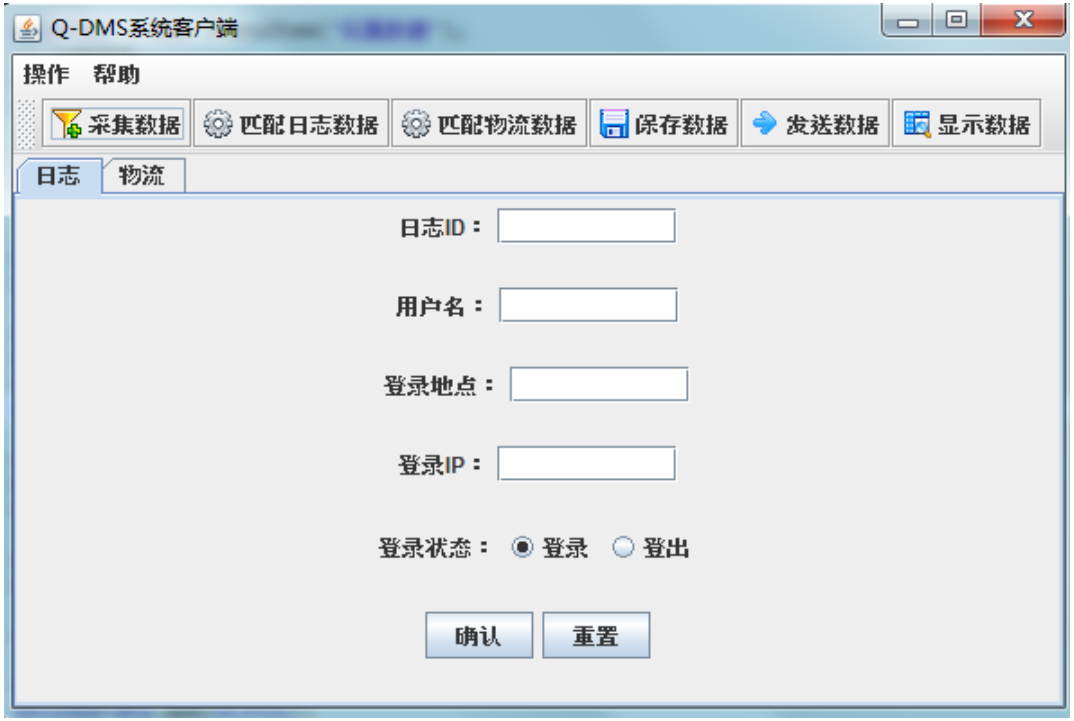
实验内容

主界面设计

1. 《匹配数据和系统帮助》菜单项
2. 《采集物流数据》选项卡
3. 《匹配物流数据》工具条
4. 《保存数据》事件响应
5. 《显示数据》物流表格数据

系统主界面

- 授权登录后进入的系统界面(MainFrame.java)，完成以下功能：
  - 匹配数据和帮助菜单设计及实现
  - 采集数据任务栏中的物流采集界面设计
  - 匹配物流数据任务栏设计及实现
  - 数据保存任务栏的事件响应，即补写事件处理方法
  - 显示数据任务栏的物流数据显示设计，即以表格形式显示匹配后的物流数据记录

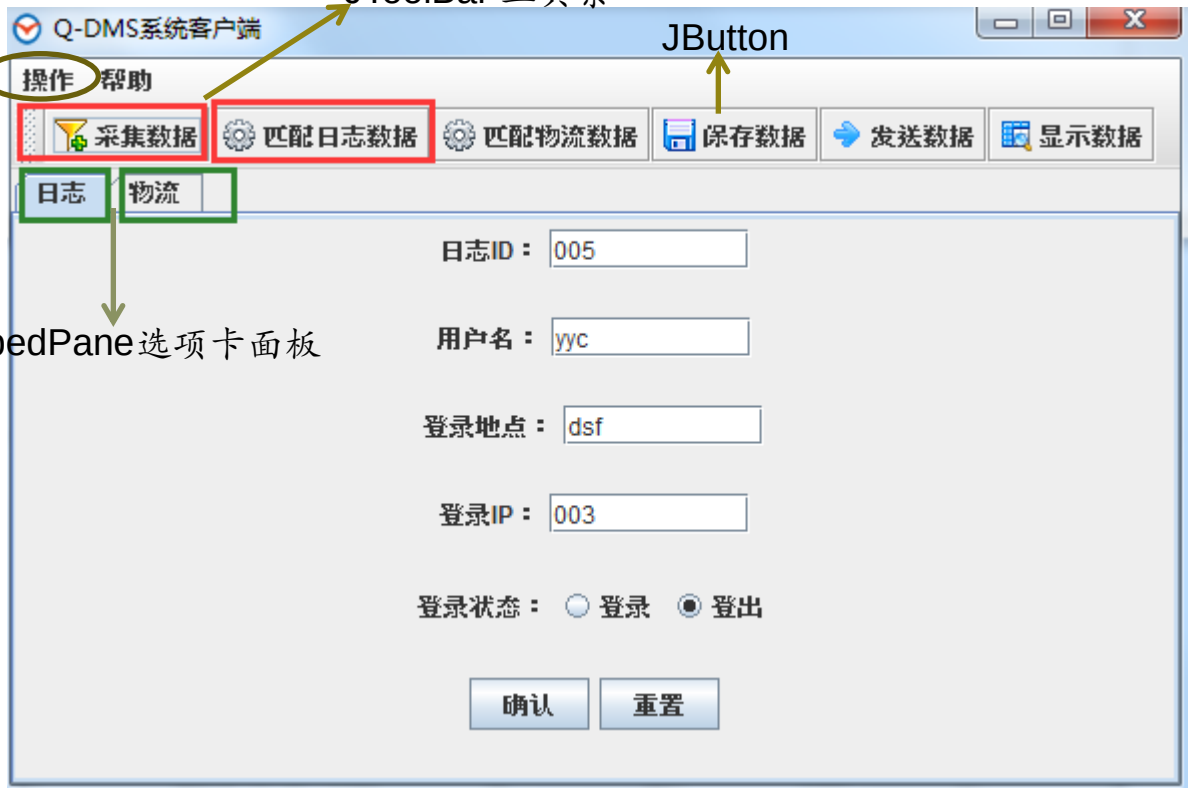


系统主界面

JMenuBar 菜单条
Jmenu 菜单
JMenuItem 菜单项

JToolBar 工具条

JButton



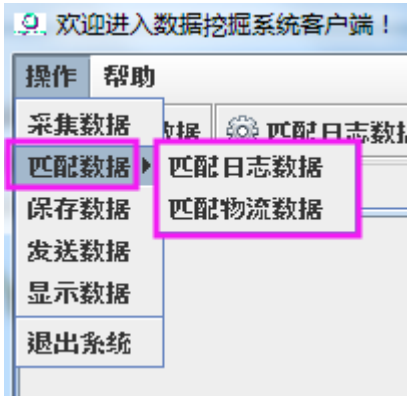
JTabbedPane选项卡面板

```
// 初始化菜单组件
menuBar = new JMenuBar();
this.setJMenuBar(menuBar);
menuOperate = new JMenu("操作");
menuBar.add(menuOperate);
```

窗口添加JMenuBar、JMenu和JMenuItem三个类实现。
分别对应菜单条、菜单和菜单项。
同时，可以给JMenuItem注册侦听器，但不能对JMenuBar，JMenu注册侦听器。

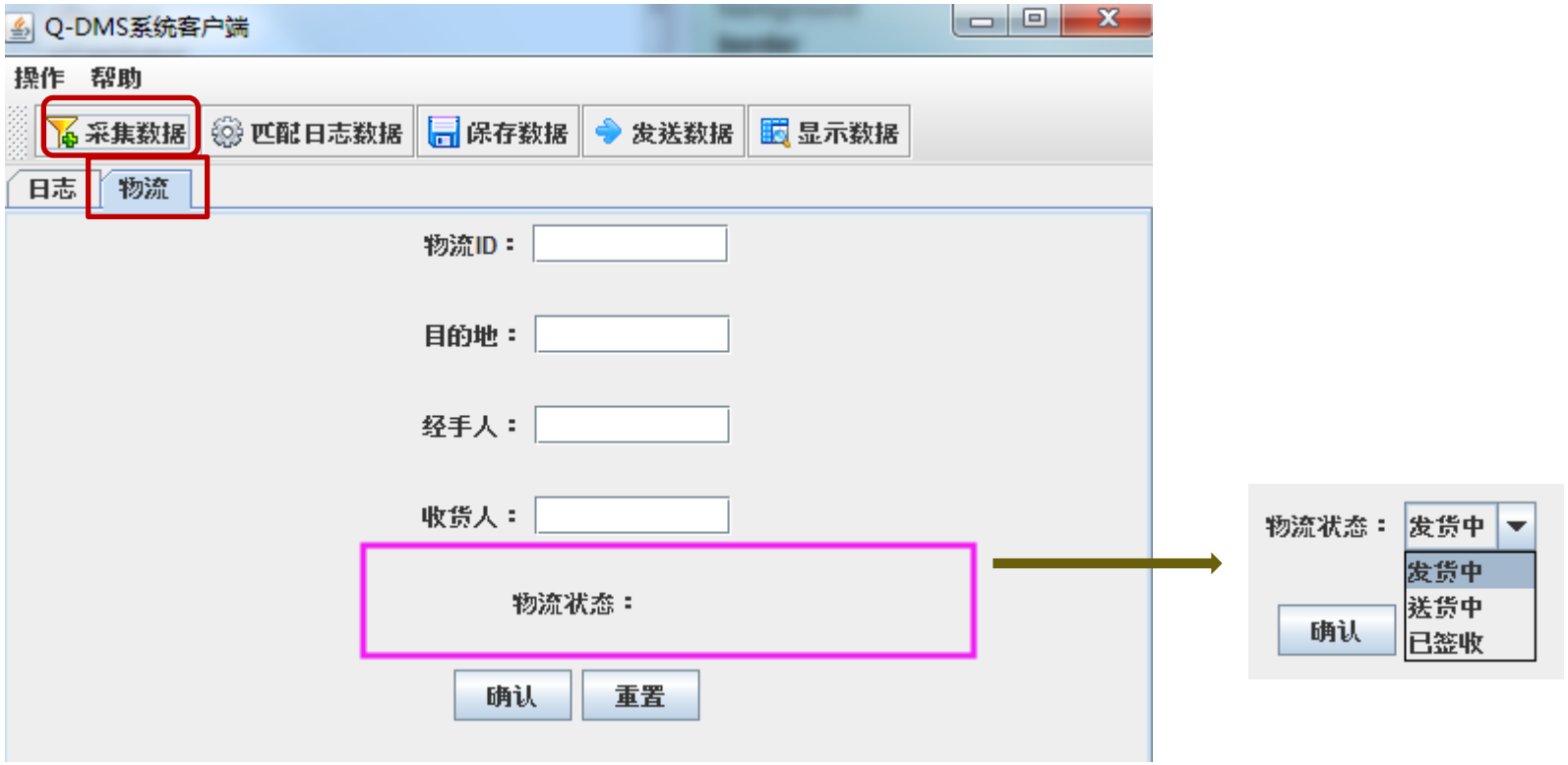
系统主界面

- 菜单栏中的匹配数据菜单设计，即匹配日志数据和物流数据菜单项的功能实现
- 帮助菜单栏的响应事件完善



系统主界面

- 任务栏中采集数据的物流采集UI设计
  要求能按下图将物流数据采集界面物流状态设计并实现



系统主界面

- 任务栏中的匹配物流数据设计
  并给出友好界面提示框

系统主界面

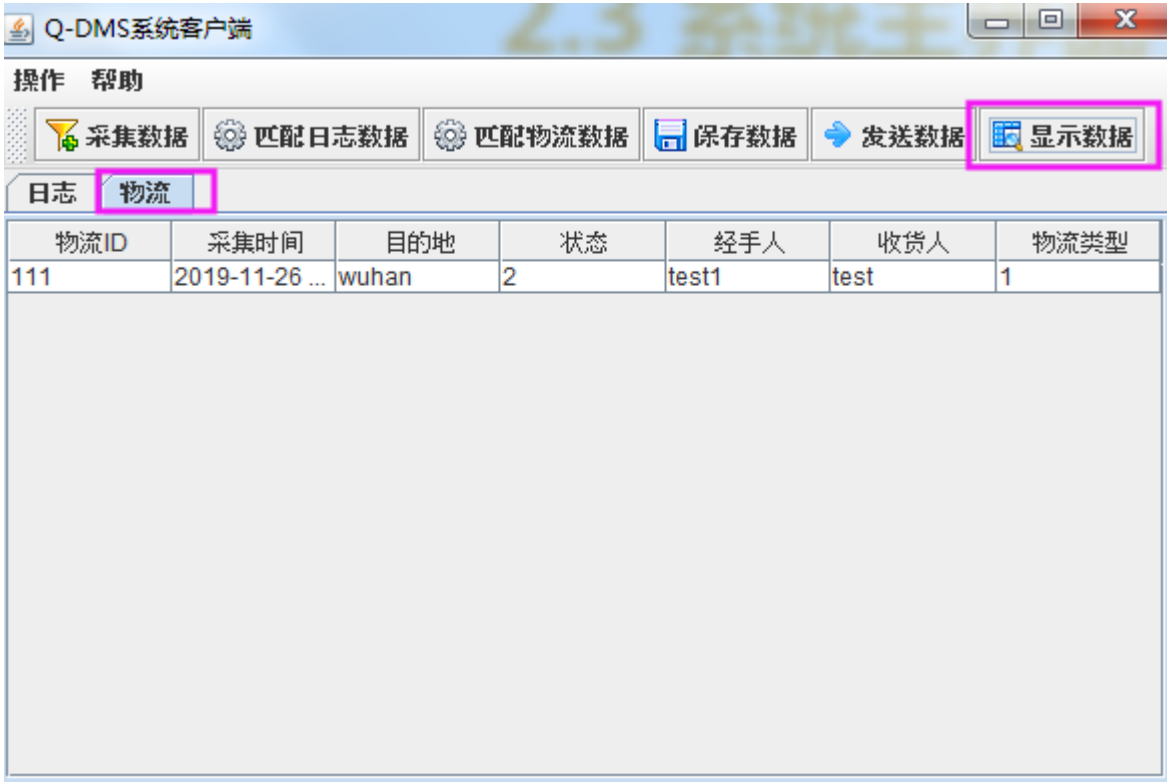
- 任务栏中的保存数据的事件响应

  将匹配后的日志和物流数据保存到文件和数据库，并给出消息提示

```
// 保存数据监听类
private class SaveDataListener implements ActionListener {
    // 数据保存的事件处理方法
    public void actionPerformed(ActionEvent e) {

        // 补充数据保存的事件处理方法

        // 保存匹配的物流信息
        //若保存成功，弹出提示框：匹配的日志数据以保存到文件和数据库中！",
        //若没有保存成功，则弹出相应的告警提示框

        // 保存匹配的物流信息
        //若保存成功，弹出提示框：匹配的物流数据以保存到文件和数据库中！",
        //若没有保存成功，则弹出相应的告警提示框
    }
}
```

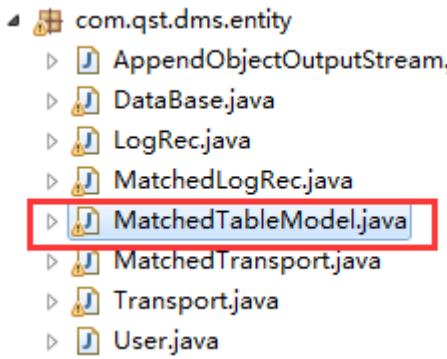
系统主界面

- 工具栏中显示物流数据设计，以表格形式显示匹配后的物流数据记录

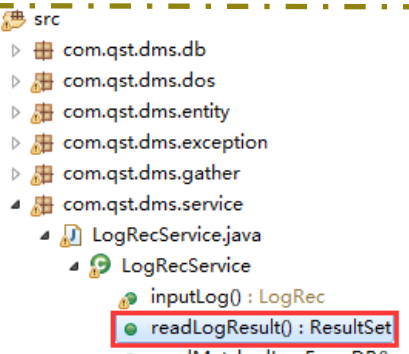


系统主界面

- 以日志为例，以表格形式显示匹配后的物流数据记录



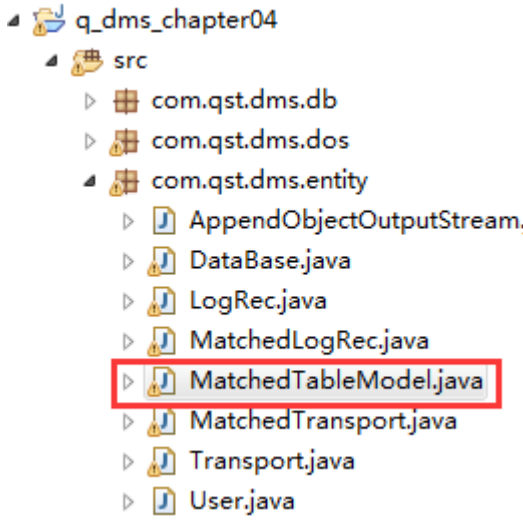

```
// 刷新日志选项卡，显示日志信息表格
private void flushMatchedLogTable() {
    // 创建tableModel，通过标志为区分日志和物流：1，日志；0，物流
    MatchedTableModel logModel = new MatchedTableModel(
            logRecService.readLogResult(), 1);
    // 使用tableModel创建JTable
    JTable logTable = new JTable(logModel);
    // 通过JTable对象创建JScrollPane，显示数据
    scrollPane = new JScrollPane(logTable);
    // 添加日志选项卡
    showPane.addTab("日志", scrollPane);
}
```




```
//获取数据库中的所有匹配的日志信息，返回一个ResultSet
public ResultSet readLogResult() {
    DBUtil db = new DBUtil();
    ResultSet rs=null;
    try {
        // 获取数据库链接
        Connection conn=db.getConnection();
        // 查询匹配的日志，设置ResultSet可以使用除了next()之外的方法操作结果集
        Statement st=conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);

        String sql = "SELECT * from gather_logrec";
        rs = st.executeQuery(sql);
    }catch (Exception e) {
        e.printStackTrace();
    }
    return rs;
}
```

系统主界面

- 自定义MatchedTableModel类
- JTable表格控件中数据与表格是分开的：表格内的数据是单独存放的，与JTable本身无关。表格内能看到数据是因为 JTable与数据之间存在一一映射关系
- 在Java中JTable的数据是以TableModel表模式的方式存放的
- Java提供了实现TableModel接口的两个类：
  - ✓ AbstractTableModel
  - ✓ DefaultTableModel




```java
public class MatchedTableModel extends AbstractTableModel {
    // 使用ResultSet来创建TableModel
    private ResultSet rs;
    private ResultSetMetaData rsmd;
    // 标志位，区分日志和物流：1，日志；0，物流
    private int sign;
    public MatchedTableModel(ResultSet rs, int sign) {
        this.rs = rs;
        this.sign = sign;
        try {
            rsmd = rs.getMetaData();
        } catch (Exception e) {
            rsmd = null;
        }
    }
    // 获取表格的行数
    public int getRowCount() {
        try {
            rs.last();
            // System.out.println(count);
            return rs.getRow();
        } catch (Exception e) {
            return 0;
        }
    }
```

知识点

JAVA GUI

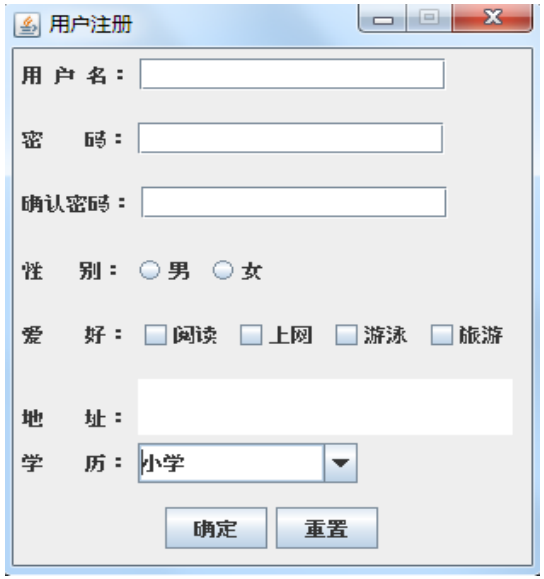
GUI,Graphics User Interface图形用户界面/接口：采用图形方式显示的计算机操作用户界面

键盘输入文本/字符命令完成例行任务的字符界面

```
**************************
欢迎进入日志物流信息数据系统!
* 1、数据采集          2、数据匹配 *
* 3、数据记录          4、数据显示 *
* 5、数据发送          0、退出应用 *
**************************
请输入菜单项（0~5）：
```

便于用户与程序之间交互！

窗口、菜单、按钮、文本框及其相应响应机制的图形用户界面



◆ **Java中对于GUI的支持是通过两个类库：**

1.AWT包(java.awt.*)

2.Swing包(javax.swing.*)

JAVA GUI

◆AWT-Abstract Windows Tools

- Sun公司最早提供的GUI库
- 界面不美观，功能有限
- 运行在不同的平台上，呈现不同的外观效果

◆ Swing

- 是对AWT的扩展和补充
- Java实现，与平台无关，保证不同平台上用户一致感观效果
- 用户界面组件丰富，使用便捷

◆ JAVAFX

高性能媒体引擎

JAVA GUI 分类

① 组件Component

基本图形元素，如按钮，标签，文本框，复选框等

② 容器Container

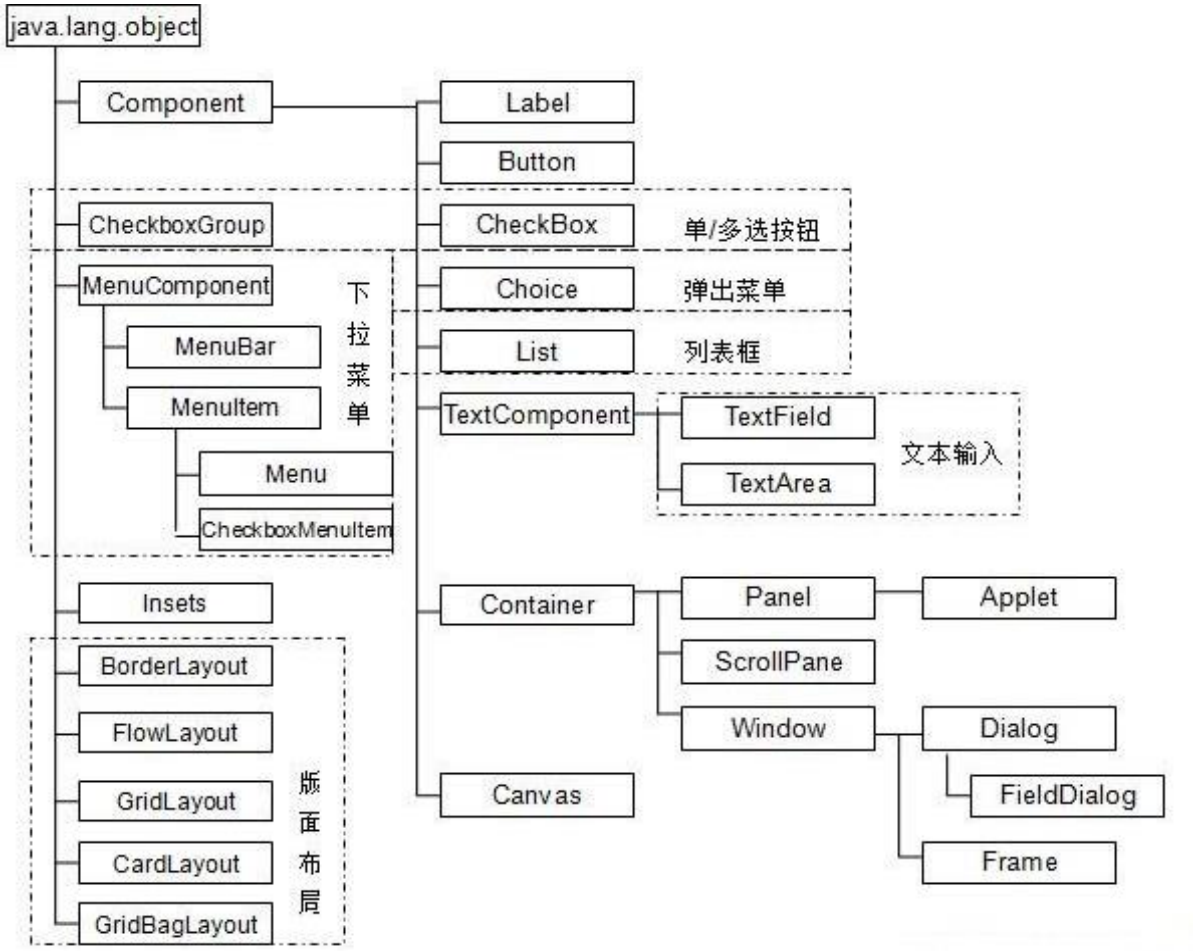
可容纳组件的区域，在容器上可添加别的组件，如Frame/Window/Panel等

说明：容器中既可以容纳组件，也可以容纳比它“容量级别小”的容器

Swing GUI开发步骤

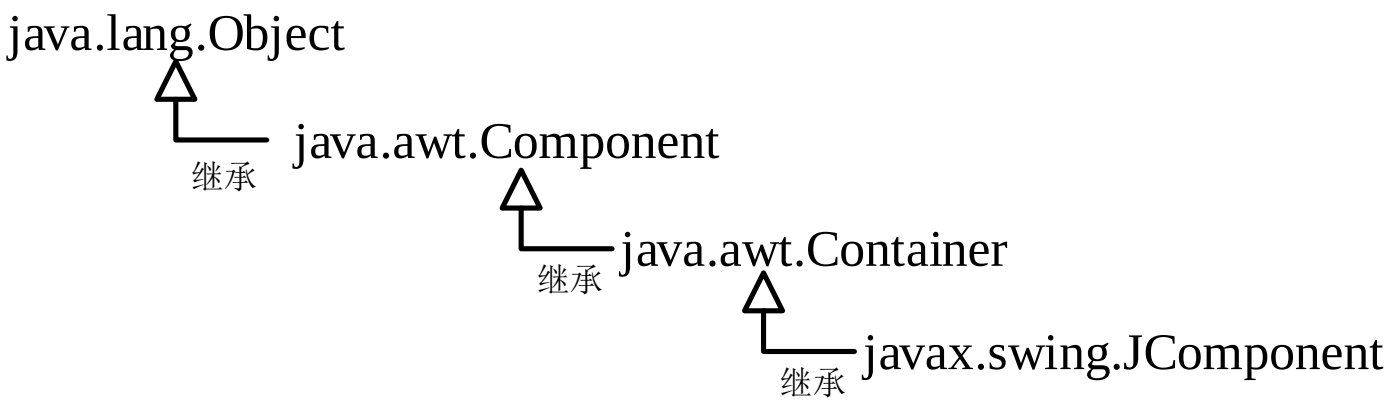
1. 创建应用程序框架，选定顶层容器
2. 在顶层容器中创建中间容器，将组件加入中间容器，并设组件属性
3. 选择布局管理器，进行合理布局
4. 实现事件处理代码，完成与用户交互

Java GUI

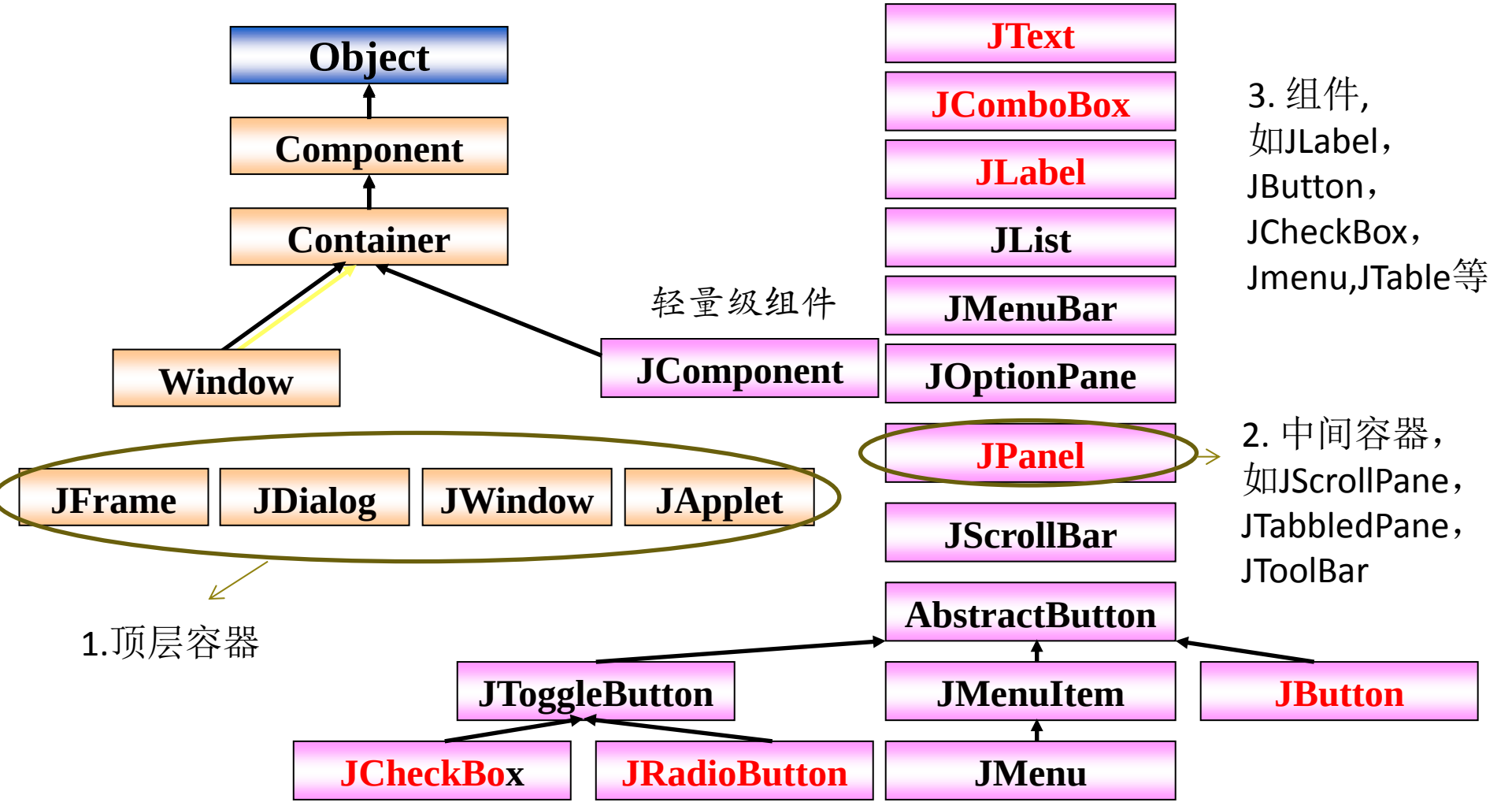

java.lang.object
Component
Label
Button
CheckboxGroup
CheckBox
单/多选按钮
MenuComponent
下拉菜单
Choice
弹出菜单
MenuBar
List
列表框
MenuItem
TextComponent
TextField
文本输入
TextArea
Menu
CheckboxMenuItem
Insets
Container
Panel
Applet
ScrollPane
BorderLayout
FlowLayout
Window
Dialog
GridLayout
FieldDialog
Canvas
CardLayout
Frame
GridBagLayout
版面布局

Java Swing组件层次

- 大部分Swing组件都是JComponent抽象类的直接或间接子类。



Java Swing组件层次



Swing 中的容器

- Swing 中的容器有两类：
  - **顶级容器：**
    一般是一个顶层窗口（JFrame）
  - **中间容器：**
    需要包含在顶层容器中使用的容器（JPanel）

与AWT不同，Swing组件不能直接添加到顶层容器，必须添加到一个与顶层容器相关联的内容面板content pane上

内容面板是顶层容器包含的一个普通容器，是一个轻量级组件，所以原则是：要把Swing组件放入一个顶层的Swing容器的内容面板上

顶级容器

- JFrame：用于框架窗口的类

  此窗口带有边框、标题、用于关闭和最小化窗口的图标等
  带 GUI 的应用程序通常至少使用一个框架窗口
- JDialog：用于对话框的类
- JApplet：用于使用 Swing 组件的 Java Applet 的类

JFrame类

- JFrame（窗口框架）
  - 是可以独立存在的顶级窗口容器
  - JFrame能够包含其他子容器，但不能被其他容器所包含。
- JFrame类常用的构造方法
  - JFrame()
  - JFrame(String title)

方法	功能描述
protected void frameInit()	在构造方法中调用该方法用来初始化窗体
public Component add(Component comp)	向窗口中添加组件
public void setLocation(int x,int y)	设置窗口的位置坐标（以像素为单位）
public void setSize(int width,int height)	置窗口的大小（以像素为单位）
public void setSize(int width,int height)	设置是否可视，当参数为true则可视，false则隐藏
public void setContentPane(Container contentPane)	设置容器面板
public void setIconImage(Image image)	设置窗体左上角的图标
public void setJMenuBar(JMenuBar menubar)	设置窗体的菜单栏
public void setTitle(String title)	设置窗口的标题

JFrame类

JFrame是一个顶层容器，JFrame上只能有唯一个组件，为JRootPane，调用JFrame.getContentPane()方法可获得JFrame中内置的JRootPane对象

*应用程序不能直接在JFrame实例对象上增加组件和设置布局管理器，只能在JRootPane对象上增加子组件和设置布局管理器

如何对JFrame添加组件？

1.用getContentPane()方法获得JFrame的内容面板，再对其加入组件 frame.getContentPane().add(nameLabel)

2.建立一个JPanel之类的中间容器，把组件添加到容器中，用setContentPane()方法把该容器置为JFrame的内容面板

中间容器

- JPanel：面板，它是最灵活、最常用的中间容器
- JScrollPane：与 JPanel 类似，但还可在大的组件或可扩展组件周围提供滚动条
- JTabbedPane：包含多个组件，但一次只显示一个组件。用户可在组件之间方便地切换
- JToolBar：按行或列排列一组组件（通常是按钮）

中间容器-JPanel

- JPanel是一种中间容器，不能独立存在
- 在使用JPanel时，通常先将其他组件添加到JPanel中间容器中，再将JPanel中间容器添加到JFrame顶级容器中

■ JPanel类常用的构造方法：

- JPanel ()
- JPanel(LayoutManager layout)

■ JPanel类常用方法及功能：

方法	功能描述
public Component add(Component comp)	该方法从Container类中继承而来，用于向面板容器中添加其他组件
public void setLayout(LayoutManager mgr)	该方法从Container类中继承而来，用于设置面板的布局方式

组件-JButton

- JButton类提供一个可接受单击操作的按钮功能，其构造方法：
  - JButton(String str)
  - JButton(Icon icon)
  - JButton(String str,Icon icon)
- JButton类常用方法

方法	功能描述
String getText()	获取按钮上的文本内容
void setText(String str)	设置按钮上的文本内容
void setIcon(Icon icon)	设置按钮上的图标

本实验需要了解的组件及分类

文本组件JTextComponent

- JTextField 单行文本框
- JTextArea 多行滚动
- JLabel 提示标签
- JPasswordField 口令输入

选择性输入表单

- JCheckBox 复选框
- JRadioButton 单选框
- JList 列表框
- JComboBox 下拉列表组合框

其他用户组件

- JTabbedPane 页标签面板
- JScrollBar 滚动条
- JMenuBar/JMenu/JMenuItem 菜单

组件观感 | 对话框JDialog | 文件对话框JFileChooser | 消息框JOptionPane

showMessageDialog 信息提示
showOptionDialog 选择可选项
showInputDialog
showConfirmDialog 选择Y/N

组件-JLabel

- JLabel标签具有标识和提示作用。
- 标签没有边界，也不会响应用户操作。

- JButton类常用方法
  - JLabel(String text)
  - JLabel(Icon icon)
  - JLabel(String text, Icon icon, int horizontalAlignment)
- Jlabel构造方法：

方法	功能描述
void setText(String txt)	设置标签中的文本内容
void setIcon(Icon icon)	设置标签中的图标
String getText()	获取标签中的文本内容

组件-文本组件

Swing常用的文本组件有三种:

- JTextField：文本框
- JTextArea：文本域
- JPasswordField：密码框

JTextField

- JTextField类只能接收单行的文本输入

方法	功能描述
JTextField(int cols)	构造方法，创建一个内容是空的、指定长度的文本框
JTextField(String str)	构造方法，创建一个指定文本内容的文本框
JTextField(String s, int cols)	构造方法，创建一个指定文本内容的、指定长度的文本框
String getText()	获取文本框中用户输入的文本内容
void setText(String str)	设置文本框中的文本内容为指定字符串内容

JTextArea

- JTextArea文本域组件可以编辑多行多列文本，且具有换行能力

方法	功能描述
JTextArea(int rows,int columns)	构造方法，创建一个内容是空的、指定行数及列数的文本域
JTextArea(String text)	构造方法，创建一个指定文本内容的文本域
JTextArea(String text,int rows,int columns)	构造方法，添加组件创建一个指定文本内容的、指定行数及列数的文本域
String getText()	获取文本域中用户输入的文本内容
void setText(String str)	设置文本域中的文本内容为指定字符串内容

JPasswordField

- JPasswordField类不显示原始字符，用于接收用户输入的密码

方法	功能描述
JPasswordField (int cols)	构造方法，创建一个内容是空的、指定长度的密码框
JPasswordField (String str)	构造方法，创建一个指定密码信息的密码框
JPasswordField (String s, int cols)	获取密码框中用户输入的密码，以字符型数组形式返回
char[] getPassword()	获取密码框中用户输入的密码，以字符型数组形式返回
setEchoChar(char c)	设置密码框中显示的字符为指定的字符

JComboBox组合框

- JComboBox组合框是一个文本框和下拉列表的组合，常用构造方法：
  - JComboBox()
  - JComboBox(Object[] listData)
  - JComboBox(Vector<?> listData)
- JComboBox类常用方法：

方法	功能描述
int getSelectedIndex()	获取用户选中的选项的下标
Object getSelectedValue()	获得用户选中的选项值
void addItem(Object obj)	添加一个新的选项内容
void removeAllItems()	从项列表中移除所有项

JList列表框

- JList列表框中的选项以列表的形式显示出来，JList类常用的构造方法：
  - JList ()
  - JList(Object[] listData)
  - JList(Vector<?> listData)
- JList类常用方法：

方法	功能描述
int getSelectedIndex()	获得选中选项的下标
Object getSelectedValue()	获得列表中用户选中的选项的值
Object[] getSelectedValues()	以对象数组的形式返回所有被选中选项的值
void setModel(ListModel m)	设置表示列表内容或列表“值”的模型
void setSelectionMode(int selectionMode)	设置列表的选择模式

JRadioButton单选按钮

- JRadioButton单选按钮可被选择、或被取消选择，常用的构造方法：
  - JRadioButton(String str)
  - JRadioButton(String str,boolean state)
- JRadioButton类常用方法：

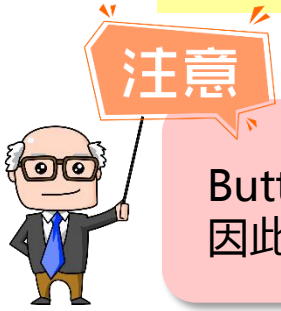
方法	功能描述
void setSelected(boolean state)	设置单选框的选中状态
boolean isSelected()	判断单选按钮是否被选中，返回一个布尔值

JRadioButton单选按钮

使用单选按钮要经过两个步骤：

- 实例化所有的JRadioButton单选按钮对象
- 创建一个ButtonGroup按钮组对象，并用其add()方法将所有的单选按钮添加到该组中

```
// 创建单选按钮
JRadioButton rbMale = new JRadioButton("男", true);
JRadioButton rbFemale = new JRadioButton("女");
// 创建按钮组
ButtonGroup bg = new ButtonGroup();
// 将rb1和rb2两个单选按钮添加到按钮组中,这两个单选按钮只能选中其一
bg.add(rbMale);
bg.add(rbFemale);
```



ButtonGroup只是为了实现单选规则的逻辑分组，而不是物理上的分组，因此仍要将JRadioButton单选按钮对象添加到容器对象中。

JCheckBox复选按钮

- JCheckBox复选框可以控制选项的开启或关闭，常用的构造方法：
  - JCheckBox(String str)
  - JCheckBox(String str,boolean state)
- JCheckBox类常用方法：

方法	功能描述
void setSelected(boolean state)	设置复选按钮的选中状态
boolean isSelected()	判断复选按钮是否被选中

JOptionPane消息提示框

- JOptionPane用于显示消息或获取信息
- JOptionPane类提供了四个静态方法

方法	功能描述
showConfirmDialog()	显示确认对话框，等待用户确认（OK/Cancle）
showInputDialog()	显示输入对话框，等待用户输入信息，并以字符串形式返回用户输入的信息
showMessageDialog()	只有一个确认OK按钮的消息提示框，有告警，错误提示等消息提示，等待用户点击OK按钮
showOptionDialog()	自定义选择按钮的显示选择对话框，等待用户在一组选项中选择，并返回用户选择的选项下标值

确认对话框

- JOptionPane.showConfirmDialog()静态方法用于显示确认对话框
- 常用的构造方法：
  - int showConfirmDialog(Component component,Object message)
  - int showConfirmDialog(Component component,Object message,String title,int optionType)
  - showConfirmDialog(Component component, Object message, String title, int optionType, int messageType)

确认对话框

- optionType参数代表选项类型
- 在JOptionPane类中提供了四种选项类型的静态变量：
  - DEFAULT_OPTION：默认选项
  - YES_NO_OPTION：Yes和No选项
  - YES_NO_CANCEL_OPTION：Yes、No和CANCEL选项
  - OK_CANCEL_OPTION：Ok和Cancel选项
- 【示例】确认对话框

```
JOptionPane.showConfirmDialog(null,
                "您确定要删除吗？",
                "删除",
                JOptionPane.YES_NO_OPTION);
```

JFileChooser文件对话框

- JFileChooser类用于打开和保存文件时所显示的对话框
- JFileChooser常用的构造方法：
  - JFileChooser()
  - JFileChooser(String currentDirectoryPath)
- JFileChooser类常用的方法：

方法	功能描述
int showOpenDialog(Component parent)	显示打开文件对话框
int showSaveDialog(Component parent)	显示保存文件对话框
File getSelectedFile()	获取选中的文件对象
File getCurrentDirectory()	获取当前文件路径

JColorChooser颜色对话框

- JColorChooser类,提供一个用于用户操作和选择颜色的控制器窗格
- JColorChooser常用的构造方法：
  - JColorChooser()
  - JColorChooser(Color initialColor)
  - JColorChooser(ColorSelectionModel model)

JMenu菜单

- 菜单是用户用来输入有关操作命令的简便工具，菜单可以将多种操作组合在一起，通过下来方式或者弹出方式供用户使用
- 常用的菜单有两种样式：
  - 下拉式菜单
  - 弹出式菜单(JPopupMenu)
- Java菜单组件由3种菜单对象组成:
  - 菜单条JMenuBar
  - 菜单JMenu
  - 菜单项JMenuItem

  菜单项作用与按钮类似，在单击时引发一个动作

JMenuBar菜单条/栏

- 菜单栏是一个水平栏，用来管理菜单
- Swing中的菜单栏是通过使用JMenuBar类来创建

【示例】创建菜单栏

```
// 创建菜单栏对象
JMenuBar menuBar = new JMenuBar();
```

【示例】添加菜单栏到窗口顶部

```
frame.setJMenuBar(menuBar);
```

在Swing GUI图形界面中，顶级窗口可以是JFrame和JDialog，这两个类中都提供了setJMenuBar()方法，用于添加菜单栏对象

JMenu菜单

- 菜单用来整合管理菜单项，组成一个下拉列表形式的菜单
- Swing中使用JMenu类可以创建一个菜单对象，其常用的构造方法

- JMenu()
- JMenu(String str)
- JMenu(String str, boolean bool)

【示例】创建菜单

```
// 初始化菜单组件
menuBar = new JMenuBar();
this.setJMenuBar(menuBar);
```

JMenuItem菜单项

- 菜单项是菜单系统中最基本的组件，菜单项对象可以添加到菜单中
- 使用JMenuItem类可以创建一个菜单选项对象，其构造方法
  - JMenuItem ()
  - JMenuItem(Icon icon)
  - JMenuItem(String text)
  - JMenuItem(String text, Icon icon)
- JMenuItem类的方法

方法	功能描述
void addActionListener(ActionListenerr l)	从AbstractButton类中继承的方法，将监听对象添加到菜单项中
void setIcon(Icon icon)	设置图标
void setText(String text)	设置文本

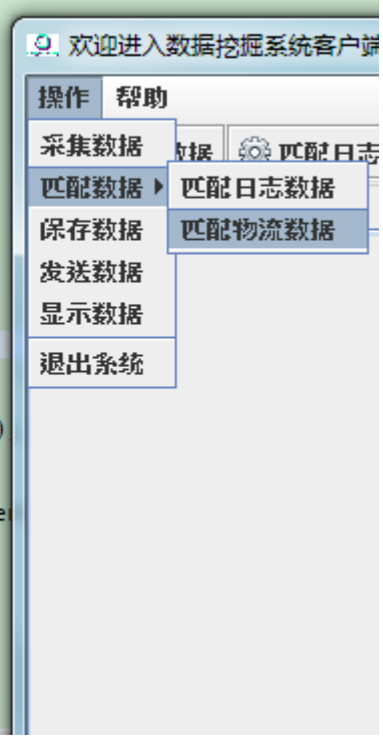
JMenuItem菜单项

- 使用JMenuBar、JMenu和JMenuItem实现下拉式菜单步骤：
  - 创建一个JMenuBar菜单栏对象，调用顶级窗口的setJMenuBar()方法将其添加到窗体顶部
  - 创建若干JMenu菜单对象，调用JMenuBar的add()方法将菜单添加到菜单栏中
  - 创建若干个JMenuItem菜单项，调用JMenu的add()方法将菜单项添加到菜单中

示例

```
// 声明界面组件
private JMenuBar menuBar;
private JMenu menuOperate, menuHelp, menuMatch;
private JMenuItem miGather, miMatchLog, miMatchTrans, miSave, miSend,
        miShow, miExit, miCheck, miAbout;
```

```
// 初始化菜单的方法
private void initMenu() {
    // 初始化菜单组件
    menuBar = new JMenuBar();
    this.setJMenuBar(menuBar);
    menuOperate = new JMenu("操作");
    menuBar.add(menuOperate);

    miGather = new JMenuItem("采集数据");
    // 注册监听
    miGather.addActionListener(new GatherListener());
    menuOperate.add(miGather);

    // 二级菜单
    menuMatch = new JMenu("匹配数据");
    miMatchLog = new JMenuItem("匹配日志数据");
    // 注册监听
    miMatchLog.addActionListener(new MatchLogListener())
    miMatchTrans = new JMenuItem("匹配物流数据");
    // 注册监听
    miMatchTrans.addActionListener(new MatchTransListene
    menuMatch.add(miMatchLog);
    menuMatch.add(miMatchTrans);
    menuOperate.add(menuMatch);

    miSave = new JMenuItem("保存数据");
    miSave.addActionListener(new SaveDataListener());
    menuOperate.add(miSave);
```



工具栏

- 工具栏是应用程序所提供的快速访问常用命令的按钮栏
- Swing中提供了JToolBar类，用于实现工具栏的功能，其构造方法：

- JToolBar()
- JToolBar(int orientation)
- JToolBar(String name)
- JToolBar(String name, int orientation)

- 工具栏常用方法

方法	功能描述
public Component add(Component c)	在工具栏中添加组件
public void addSeparator()	将分隔线添加到工具栏的末尾
public void setMargin(Insets m)	设置工具栏边框与按钮之间的空白
public Insets getMargin()	返回工具栏边框与按钮之间的空白

示例


```java
// 初始化工具栏的方法
private void initToolBar() {
    // 创建工具栏
    toolBar = new JToolBar();
    // 将工具栏添加到窗体北部（上面）
    getContentPane().add(toolBar, BorderLayout.NORTH);

    // 添加带有图标的工具栏按钮
    ImageIcon gatherIcon = new ImageIcon("images\\gatherData.png");
    btnGather = new JButton("采集数据", gatherIcon);
    // 注册监听
    btnGather.addActionListener(new GatherListener());
    toolBar.add(btnGather);

    ImageIcon matchIcon = new ImageIcon("images\\matchData.png");
    btnMatchLog = new JButton("匹配日志数据", matchIcon);
    // 注册监听
    btnMatchLog.addActionListener(new MatchLogListener());
    toolBar.add(btnMatchLog);

    //补充匹配物流数据 组件和注册监听器

    ImageIcon saveIcon = new ImageIcon("images\\saveData.png");
    btnSave = new JButton("保存数据", saveIcon);
    // 注册监听
    btnSave.addActionListener(new SaveDataListener());
    toolBar.add(btnSave);

    ImageIcon sendIcon = new ImageIcon("images\\sendData.png");
    btnSend = new JButton("发送数据", sendIcon);
    btnSend.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        }
    });
    toolBar.add(btnSend);
```

JTable

- JTable类用于创建一个表格对象，显示和编辑常规二维单元表，其构造方法：

- JTable()
- JTable(int numRows, int numColumns)
- JTable(Object[][] rowData, Object[]columnNames)
- JTable(TableModel dm)
- JTable(TableModel dm, TableColumnModel cm)
- JTable(Vector rowData, Vector columnNames)

JTable

JTable构造方法：

- JTable()： 构造一个默认的 JTable，使用默认的数据模型、默认的列模型和默认的选择模型对其进行初始化
- JTable(int numRows, int numColumns) ：使用 DefaultTableModel 构造具有 numRows 行和 numColumns列个空单元格的JTable
- JTable(Object[][] rowData, Object[] columnNames) ：构造一个 JTable 来显示二维数组 rowData 中的值，其列名称为 columnNames
- JTable(TableModel dm) ：构造一个 JTable，使用数据模型 dm、默认的列模型和默认的选择模型对其进行初始化
- JTable(TableModel dm, TableColumnModel cm) ：构造一个 JTable，使用数据模型dm、列模型 cm和默认的选择模型对其进行初始化
- JTable(TableModel dm, TableColumnModel cm, ListSelectionModel sm) ：构造一个 JTable，使用数据模型dm、列模型cm 和选择模型 sm 对其进行初始化
- JTable(Vector rowData, Vector columnNames)： 构造一个 JTable 来显示 Vector 所组成的 Vector rowData中的值，其列名称为 columnNames

JTable

- JTable类常用方法：

方法	功能描述
void addColumn(TableColumn aColumn)	添加列
void removeColumn(TableColumn aColumn)	移除列
TableCellEditor getCellEditor()	返回活动单元格编辑器
TableCellEditor getCellEditor(int row,int column)	返回指定单元格的编辑器
int getColumnCount()	返回表格的列数
TableColumnModel getColumnModel()	返回该表的列模型对象
TableColumnModel getColumnModel()	返回表格的行数
int getSelectedRow()	返回第一个选定行的索引
int[] getSelectedRows()	返回所有选定行的索引
int getSelectedColumn()	返回第一个选定列的索引
int[] getSelectedColumns()	返回所有选定列的索引
int getSelectedRowCount()	返回选定行数
getSelectedColumnCount()	返回选定列数

TableModel表模式

- JTable表格控件中数据与表格是分开的：表格内的数据是单独存放的，与JTable本身无关。表格内能看到数据是因为 JTable与数据之间存在一一映射关系
- 在Java中JTable的**数据**是以**TableModel表模式**的方式存放的
- TableModel接口定义在javax.swing.table包中，接口中的方法如下：

方法	功能描述
void addTableModelListener(TableModelListener l)	注册TableModelEvent监听
Class getColumnClass(int columnIndex)	返回列数据类型的类名称
int getColumnCount()	返回列的数量
String getColumnName(int columnIndex)	返回指定下标列的名称
int getRowCount()	返回行数
Object **getValueAt**(int rowIndex,int columnIndex)	返回指定单元格（cell）的值
boolean isCellEditable(int row,int column)	返回单元格是否可编辑
void removeTableModelListener(TableModelListener l)	移除一个监听
void setValueAt(Object aValue,int row,int column)	设置指定单元格的值

TableModel表模式

- Java提供了实现TableModel接口的两个类：
  - AbstractTableModel
  - DefaultTableModel

通常将JTable对象放在JScrollPane中显示，使用JScrollPane盛放JTable时不仅可以为JTable增加滚动条，还可以让JTable的列标题显示出来；如果不将JTable放在JScrollPane中显示，JTable默认不会显示对应的列标题。

TableModel

- JTable(TableModel dm) ：构造一个 JTable，使用数据模型 dm、默认的列模型和默认的选择模型对其进行初始化
- 在这个构造函数中，需要传递一个数据模型TableModel 用来存放数据，当表格显示时就直接通过这个TableModel来获取表格的信息以及数据
- TableMode是一个**接口**，需要实现**AbstractTableModel**的方法，而AbstractTableModel又是一个**抽象的类**，也就是说在使用自己的TableModel的时候要重写一个自己的TableMode类，通过这个Model来控制自己表格的数据，AbstractTableModel中有些方法是已经实现的了，所以我们只需要对自己需要的方法进行重写

TableModel

- AbstractTableModel的基本方法

方法	功能描述
int getColumnCount()	返回该模型中的列数
String getColumnName(int columnIndex)	返回 columnIndex 位置的列的名称
int getRowCount()	返回该模型中的行数
Object **getValueAt**(int rowIndex,int columnIndex)	返回指定单元格（cell）的值
boolean isCellEditable(int rowIndex,int columnIndex)	如果 rowIndex 和 columnIndex位置的单元格是可编辑的，则返回 true。就是设置当前位置的单元格的数据是否可以被编辑
void setValueAt(Object aValue,int row,int column)	将 columnIndex和 rowIndex 位置的单元格中的值设置为 aValue。

新建一个Test类

```java
// 刷新日志选项卡，显示日志信息表格
private void flushMatchedLogTable() {
    // 创建tableModel，通过标志为区分日志和物流：1，日志；0，物流
    MatchedTableModel logModel = new MatchedTableModel(
            logRecService.readLogResult(), 1);
    // 使用tableModel创建JTable
    JTable logTable = new JTable(logModel);
    // 通过JTable对象创建JScrollPane，显示数据
    scrollPane = new JScrollPane(logTable);
    // 添加日志选项卡
    showPane.addTab("日志", scrollPane);
}
```

```java
package com.qst.dms.service;

import javax.swing.table.AbstractTableModel;

public class test extends AbstractTableModel {

    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        return null;
    }

}
```

MatchedTableModel类

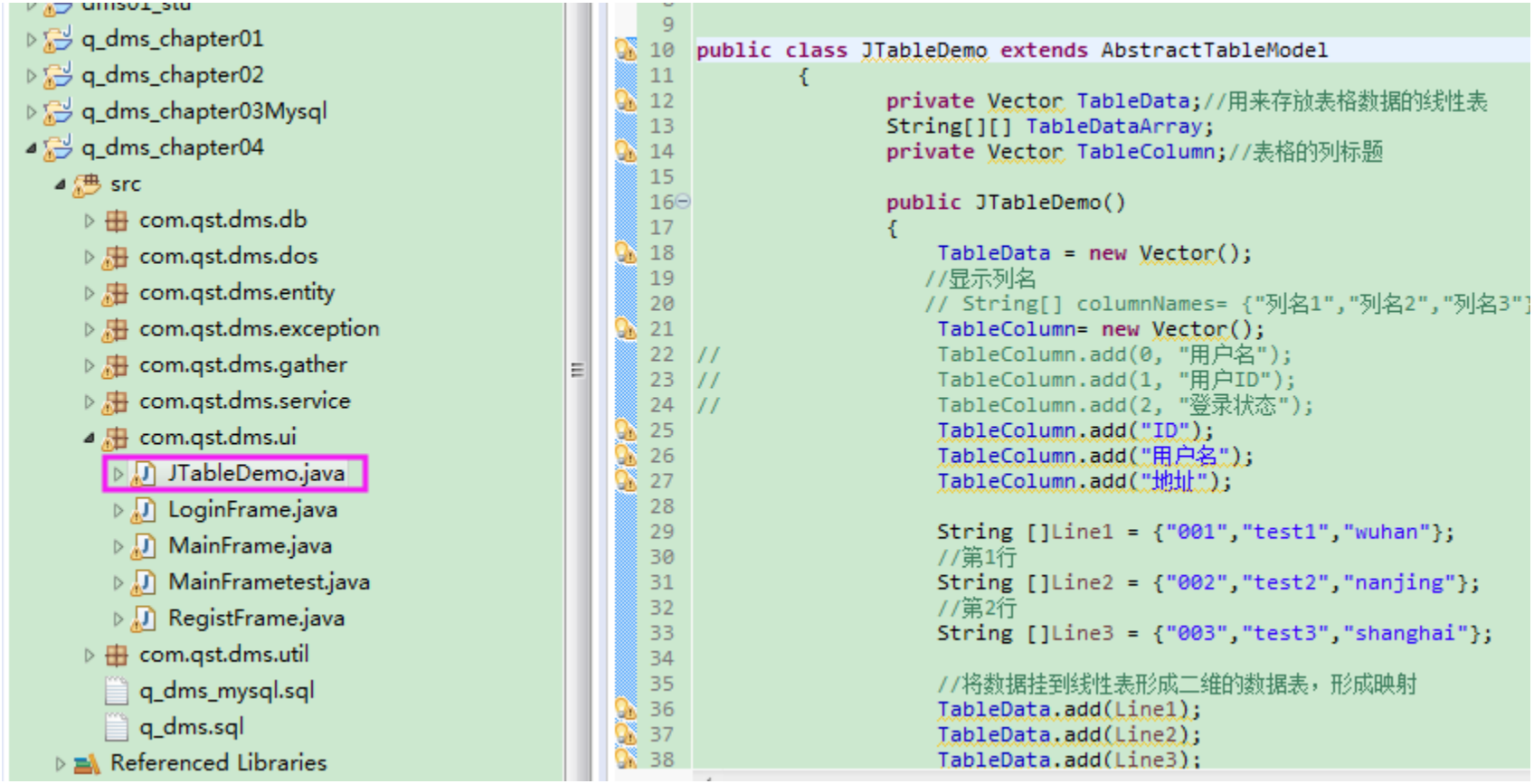
entity 包中的自定义的MatchedTableModel类，继承AbstractTableModel类

学会使用JTable(TableModel dm)

```java
package com.qst.dms.entity;

import java.sql.ResultSet;

public class MatchedTableModel extends AbstractTableModel {

    // 使用ResultSet来创建TableModel
    private ResultSet rs;
    private ResultSetMetaData rsmd;
    // 标志位，区分日志和物流：1，日志；0，物流
    private int sign;

    public MatchedTableModel(ResultSet rs, int sign) {
        this.rs = rs;
        this.sign = sign;
        try {
            rsmd = rs.getMetaData();
        } catch (Exception e) {
            rsmd = null;
        }
    }

    // 获取表格的行数
    public int getRowCount() {
        try {
            rs.last();
            // System.out.println(count);
            return rs.getRow();
        } catch (Exception e) {
            return 0;
        }
    }

    // 获取表格的列数
    public int getColumnCount() {
        try {
            // System.out.println(rsmd.getColumnCount());
```

```java
// 获取表格的列数
public int getColumnCount() {
    try {
        // System.out.println(rsmd.getColumnCount());
        return rsmd.getColumnCount();
    } catch (Exception e) {
        return 0;
    }
}

// 获取指定位置的值
public Object getValueAt(int rowIndex, int columnIndex) {
    try {
        rs.absolute(rowIndex + 1);
        return rs.getObject(columnIndex + 1);
    } catch (Exception e) {
        return null;
    }
}

// 获取表头信息
public String getColumnName(int column) {
    String[] logArray = { "用户名", "登录时间", "登出时间","登录地点" };
    String[] tranArray = { "物流ID", "采集时间", "目的地", "状态", "经手人", "收货人",
            "物流类型" };
    return sign == 1 ? logArray[column] : tranArray[column];
}
```

JTable示范Demo




```
9
10  public class JTableDemo extends AbstractTableModel
11          {
12                  private Vector TableData;//用来存放表格数据的线性表
13                  String[][] TableDataArray;
14                  private Vector TableColumn;//表格的列标题
15
16                  public JTableDemo()
17                  {
18                      TableData = new Vector();
19                     //显示列名
20                     // String[] columnNames= {"列名1","列名2","列名3"}
21                      TableColumn= new Vector();
22  //                  TableColumn.add(0, "用户名");
23  //                  TableColumn.add(1, "用户ID");
24  //                  TableColumn.add(2, "登录状态");
25                      TableColumn.add("ID");
26                      TableColumn.add("用户名");
27                      TableColumn.add("地址");
28
29                      String []Line1 = {"001","test1","wuhan"};
30                      //第1行
31                      String []Line2 = {"002","test2","nanjing"};
32                      //第2行
33                      String []Line3 = {"003","test3","shanghai"};
34
35                      //将数据挂到线性表形成二维的数据表，形成映射
36                      TableData.add(Line1);
37                      TableData.add(Line2);
38                      TableData.add(Line3);
```

TableColumnModel接口

- 在创建一个指定表格列模型的JTable对象时，需要使用TableColumnModel类型的参数来指定表格的列模型。
- TableColumnModel接口中提供了有关表格列模型的方法：

方法	功能描述
void addColumn(TableColumn aColumn)	添加一列
void moveColumn(int columnIndex,int newIndex)	将指定列移动到其他位置
void moveColumn(int columnIndex,int newIndex)	删除指定的列
TableColumn getColumn(int columnIndex)	获取指定下标的列
int getColumnCount()	获得表格的列数
int getSelectedColumnCount()	获取选中的列数

TableColumnModel接口

- 一般通过调用JTable对象中的getColumnModel()方法来获取TableColumnModel对象

- 【示例】使用表格列模型获取选中的列

```
//获取表格列模型
TableColumnModel columnModel = table.getColumnModel();
//获取选中的表格列
TableColumn column = columnModel.getColumn(table.getSelectedColumn());
```

ListSelectionModel接口

- JTable使用ListSelectionModel来表示表格的选择状态
- ListSelectionModel接口提供了三种不同的选择模式
  - ListSelectionModel.SINGLE_SELECTION
  - ListSelectionModel.SINGLE_INTERVAL_SELECTION
  - ListSelectionModel.MULTIPLE_INTERVAL_SELECTION
- 通过调用JTable对象的getSelectionModel()方法来获取ListSelectionModel对象

ListSelectionModel接口

- 当用户选择表格内的数据时会产生ListSelectionEvent事件，要处理此类事件就必须实现ListSelectionListener监听接口

- 该接口中定义了一个事件处理方法：

```
void valueChanged(ListSelectionEvent e)
```

布局（Layout）管理器

- 容器内的组件之间的位置和大小关系称为布局
- Java使用布局管理器来管理组件在容器中的布局
- 通过容器的setLayout()和getLayout()方法来确定布局
- JAVA的布局管理器包括以下几种
  - FlowLayout（流式布局）
  - BorderLayout （边界布局）
  - GridLayout（网格布局）
  - GridBagLayout（网格包布局）
  - CardLayout （卡片布局）
  - BoxLayout （ 箱式布局）
  - SpringLayout （弹簧布局）

事件处理

- Java采用委托事件模型来处理事件
- 将用户界面对象与程序处理逻辑分开
- 把事件的处理委托给独立的对象
- 在Java事件处理体系结构中，主要涉及的对象：
  - 事件（Event）：用户何种操作/动作？ 单击按钮
  - 事件源（Event Source）：操作谁？如按钮组件
  - 事件监听器（Event Listener）：事件发生时处理的方法？ 如接收发生事件的通知并处理事件

*如何能接收到发生事件的通知？--它必须向一个或多个事件源注册

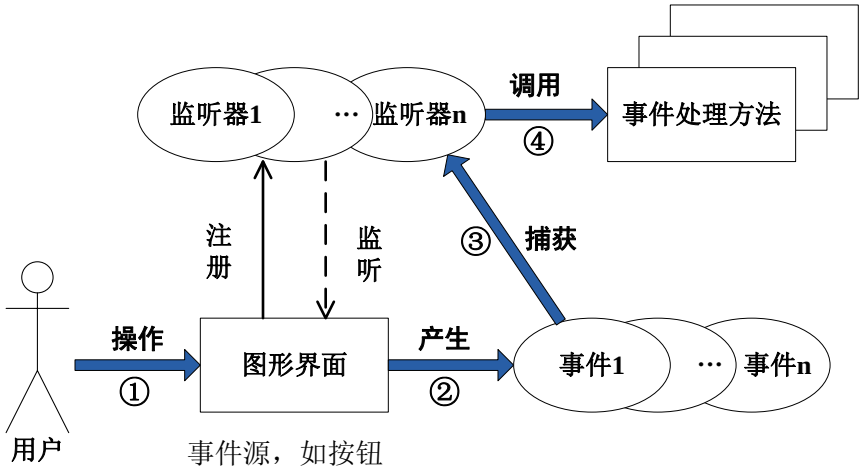
如何能处理事件呢？---实现一些方法

*任何类都能接收和管理事件

java.awt.event包中定义许多事件类，需要：import java.awt.event.*

“委托”事件处理模型

- 基于监听器的事件处理机制是一种委派式的事件处理方式，实现步骤：
  - 委托：组件将事件处理委托给特定的事件处理对象
  - 通知：触发指定的事件时，就通知所委托的事件处理对象
  - 处理：事件处理对象调用相应的事件处理方法来处理该事件



事件处理步骤

1.**生成监听器**：每个实现监听接口的类都可以作为监听器来使用。例如，LoginListener实现ActionListener接口

2.**注册监听器**：采用add***Listerner方式，监听器注册到事件源（即组件）中。例如，btnOK.addActionListener

3.事件监听:注册绑定后，监听器开始随时监听是否有事件发生

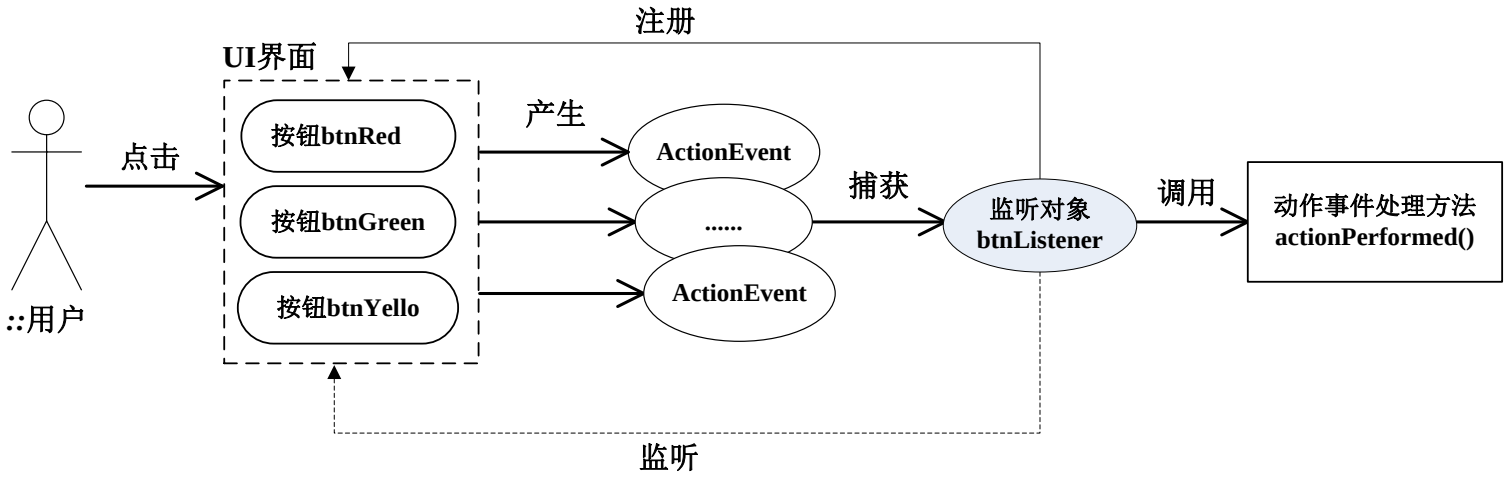
4.事件产生：如单击按钮，产生该事件的对象

5.事件通知：系统把产生的事件对象返回给已注册的监听器

6.事件响应：监听器来指派相应的方法来处理事件，由开发者编写处理代码

```
// 监听类，负责处理登录按钮
public class LoginListener implements ActionListener {
    // 重写actionPerformed()方法，事件处理逻辑
    public void actionPerformed(ActionEvent e) {
        // 根据用户名查询用户
        user = userService.findUserByName(txtName.getText().trim());
        // 判断用户是否存在
        if (user != null) {
            // 判断输入的密码是否正确
            if (user.getPassword().equals(new String(txtPwd.getPassword()))) {
                // 登录成功，隐藏登录窗口
                LoginFrame.this.setVisible(false);
                // 显示主窗口
                new MainFrame();
            } else {
                // 输出提示信息
                System.out.println("密码错误！请重新输入！");
                // 清空密码框
                txtPwd.setText("");
            }
        } else {
            // 输出提示信息
            System.out.println("该用户不存在，请先注册！");
        }
    }
}

btnOk = new JButton("登录");
btnOk.addActionListener(new LoginListener());

btnCancle = new JButton("重置");
btnCancle.addActionListener(new ResetListener());

btnRegist = new JButton("注册");
btnRegist.addActionListener(new RegistListener());
```

事件处理

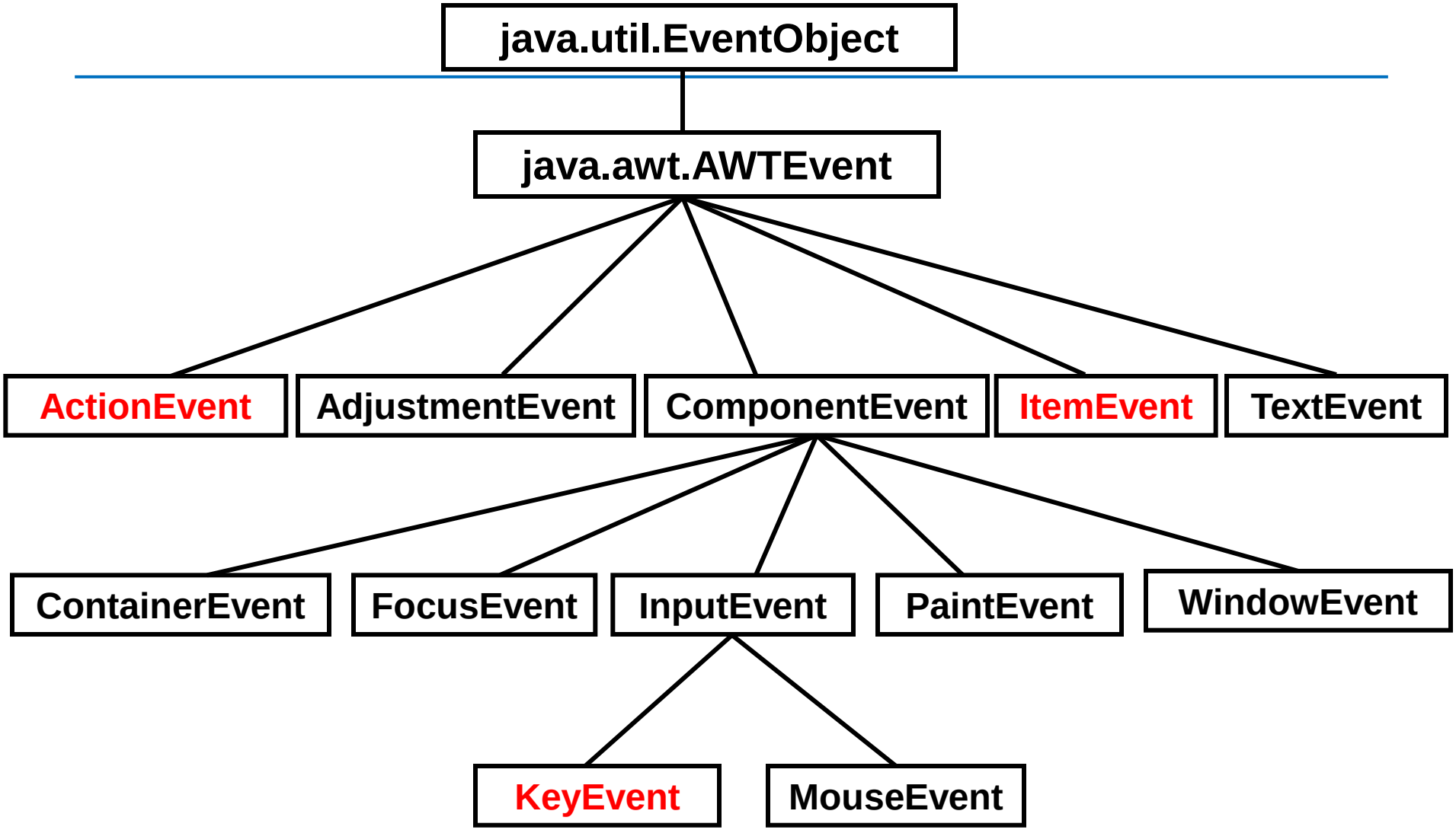

注册
UI界面
::用户
点击
按钮btnRed
按钮btnGreen
按钮btnYello
产生
ActionEvent
.....
ActionEvent
捕获
监听对象
btnListener
调用
动作事件处理方法
actionPerformed()
监听

事件类

- 事件用于封装事件处理所必须的基本信息
- 所有AWT的事件类都是AWTEvent类的子类
- AWTEvent是所有AWT事件的根事件类
- AWTEvent常用方法如下：

方法	功能描述
int getID()	返回事件类型
String paramString()	返回此Event状态的字符串，此方法仅在进行调试的时候使用
Object getSource()	从EventObject类继承的方法，用于返回事件源的对象

AWT事件处理模型—事件类



事件类

- 低级事件—在组件上发生事件则触发（MouseEvent,KeyEvent）
- 高级事件-- 语义事件(ActionEvent、AdjustmentEvent)

如ActionEvent动作事件：对应一个动作事件，不代表一个具体动作，而是一种语义，比如按钮按下。

- 每个事件类都有相应的事件监听者，一个事件监听器对象负责处理一类事件，一类事件的每一种发生情况，分别由事件监听器对象中的一个方法来具体处理。
- Java语言的事件监听者绝大多数以接口形式给出。（只要实现了其接口即可）在事件源和事件监听器对象中进行约定的接口类，成为事件监听器接口。其名称与事件类的名称相对应。

  如ActionEvent事件类的监听器接口名为ActionEventListener

事件类

- 每当在事件源上发生一个操作时，就会产生相应的事件对象
- 每个事件类都有相应的事件监听者，Java语言的事件监听者绝大多数以接口形式给出，由事件监听器对象中的一个方法来具体处理。在事件源和事件监听器对象中进行约定的接口类，成为事件监听器接口。其名称与事件类的名称相对应。

事件	监听器接口	方法
ActionEvent	ActionListener	actionPerformed(ActionEvent e)
KeyEvent	KeyListener	keyPressed(KeyEvent e)
		keyReleased(KeyEvent e)
		keyTyped(KeyEvent e)
AdjustmentEvent	AdjustmentListener	adjustmentValueChanger(TextEvent e)

事件源	产生事件的类型
JButton	ActionEvent
JTextField	ActionEvent
JMenuItem	ActionEvent
JScrollbar	AdjustmentEvent