第 3 章　下棋数据的保存

为了保存用户下棋的数据，用户必须先注册、登录服务器才能下棋。棋局结束后，将棋局保存在数据库中，将棋谱保存在数据文件中。将来可以根据用户名查询出该用户下过的所有棋局，可以选择其中的某一个棋局，将棋谱读出来回放。在这里我们使用的数据库是 MySQL 数据库。

3.1　创建数据库

3.1.1　数据库设计

为了保存用户信息和棋局信息，我们先创建一个数据库 fiveChess，数据库中包含两个数据表 user 和 game。

user 表用于保存用户信息，字段包括用户 ID、姓名、密码、电子邮箱、级别和注册日期。user 表结构如表 3.1 所示。

表 3.1　user 表结构

字　段　名	类　　型	长　　度	允　许　空	备　　注
userid	INT		否	主键，自动增长
userName	VARCHAR	10	否	用户名
passWord	VARCHAR	10	否	密码
email	VARCHAR	30	否	邮箱
level	INT		否	级别
regDate	Date		否	注册日期

game 表用于保存棋局信息，字段包括用户 ID、下棋日期、黑方、白方、赢棋方和记录棋谱的文件名。game 表结构如表 3.2 所示。

表 3.2　game 表结构

字　段　名	类　　型	长　　度	允　许　空	备　　注
gameid	INT		否	主键，自动增长
gameDate	Date		否	下棋日期
playerBlack	VARCHAR	10	否	黑方
playerWhite	VARCHAR	10	否	白方
winner	VARCHAR	10	否	赢棋方
manualFileName	VARCHAR	100	否	记录棋谱的文件名

3.1.2 创建数据库

首先创建工程 NetFive。在后面的程序中还会设计很多类，因此我们建立 3 个包，将实现下棋功能的类放在 five 包中，与用户有关的类放在 user 包中，与棋局和棋谱有关的类放在 game 包中。

首先建立 five 包，将第 2 章程序中的类全部移到 five 包中。

为了在 Java 程序中操作数据库中的数据，需要配置数据库驱动程序 JDBC。如果计算机中没有，可从 MySQL 网站上下载 mysql-connector-java 压缩文件，然后解压缩。

在 eclipse 中右击工程名 NetFive，在弹出的快捷菜单中选择 build path | add external Archives 菜单项，在 JAR Selection 对话框中找到上面解压缩的 jar 文件，如图 3.1 所示，单击“打开”按钮，将该文件导入到我们的工程中。

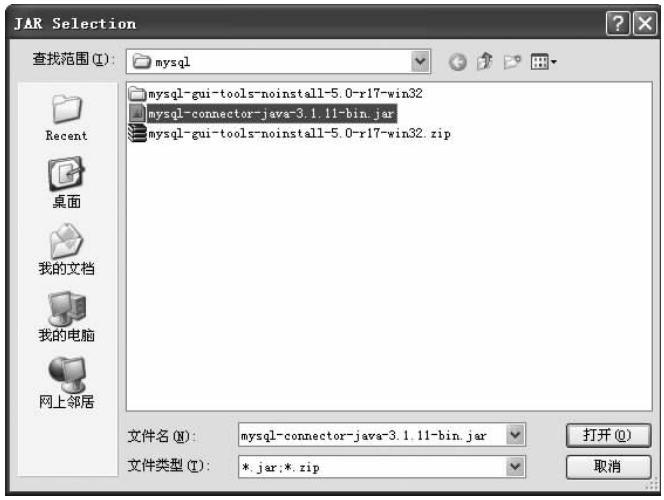

图 3.1 JAR Selection 对话框

新建 user 包，在 user 包中新建类 CreateDatabase，该类负责创建数据库 fiveChess，以及数据库中的两个表 user 和 game。代码如下：

```
public class CreateDatabase {
    public static void main(String[] args) {
        Connection conn=null;
        Statement stmt=null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/",
                 "root","1234");
            stmt=conn.createStatement();
        }
        catch (ClassNotFoundException e) {
```

```
            e.printStackTrace();
            System.exit(-1);
        }
        catch (SQLException e) {
            e.printStackTrace();
            try {
                conn.close();
            }
            catch (SQLException e1) {
                e1.printStackTrace();
                System.exit(-1);
            }
        }
        createDatabase(stmt, "fiveChess");
        try {
            stmt.close();
            conn.close();
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public static void createDatabase(Statement stmt, String dbName){
        try {
            stmt.executeUpdate("create database "+dbName);
            stmt.executeUpdate("use "+dbName);
            String sql;
            sql="create table user(id int auto_increment not null primary key,"+
                                "name VARCHAR(10) not null, "+
                                "password VARCHAR(10) not null,"+
                                "email VARCHAR(30) not null,"+
                                "level int not null,"+
                                "regDate date not null)";
            stmt.executeUpdate(sql);
            sql="insert into user(name,password,email,level,regDate) "+
                "values('test1', 'test1','test1@five.com',1,'2016-01-22')";
            stmt.executeUpdate(sql);
            sql="insert into user(name,password,email,level,regDate) "+
                "values('test2', 'test2','test2@five.com',1,'2016-01-22')";
            stmt.executeUpdate(sql);
            sql="create table game(id int auto_increment not null primary key,"+
                                "gameDate date not null, "+
                                "playerBlack VARCHAR(10) not null, "+
                                "playerWhite VARCHAR(10) not null, "+
```

```
                                "winner VARCHAR(10) not null, "+
                                "manualFileName VARCHAR(100) not null)";
            stmt.executeUpdate(sql);
            sql="insert into game(gameDate, playerBlack, playerwhite, winner,
                manualFileName)"+" values('2016-01-22','test1','test2',
                'test2','game1201601221205.fiv')";
            stmt.executeUpdate(sql);
            sql="insert into game(gameDate, playerBlack, playerwhite, winner,
                manualFileName)"+" values('2016-01-22','test1','test2',
                'test1','game1201601221848.fiv')";
            stmt.executeUpdate(sql);
        }
        catch(SQLException e) {
            e.printStackTrace();
        }
    }
}
```

在主方法中创建数据库连接 Connection 对象和 Statement 对象，然后调用 createDatabase()方法，创建数据库 fiveChess，再创建 user 表，并向 user 表中插入两条记录，最后再创建 game 表，并向 game 表插入两条记录。给两个表分别插入两条记录是为了测试创建数据库和表的功能是否完善，实际中不必插入这些记录。

运行 CreateDatabase 类，创建数据库和表，以后的程序是在这两个表的基础上实现的。创建好数据库之后，就不要再次运行 CreateDatabase 类了，以后如果需要初始化数据库，可以在这个类的基础上稍加修改。

3.2 用户管理

为了实现用户注册和登录的功能，我们首先创建用户管理类，实现用户的添加和查找功能。

3.2.1 数据库连接类

由于在后面的程序中经常要进行数据库连接操作，因此我们首先设计一个连接数据库的类 DBConnection，将其放在 user 包中。代码如下：

```
public class DBConnection {
    Connection conn=null;
    public DBConnection(){
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn=DriverManager.getConnection(
```

```
                "jdbc:mysql://localhost:3306/fiveChess","root","1234");
        }
        catch(ClassNotFoundException e) {
            e.printStackTrace();
            System.exit(-1);
        }
        catch(SQLException e) {
            e.printStackTrace();
            if(conn!=null){
                try {
                    conn.close();
                }
                catch (SQLException e1) {
                    e1.printStackTrace();
                    System.exit(-1);
                }
            }
        }
    }
    public Connection getConn() {
        return conn;
    }
}
```

DBConnection 类的代码也比较简单，在构造方法中创建 Connection 对象。其他类可以调用 DBConnection 类的 getConn()方法获得这个数据库连接对象。

3.2.2 用户管理

1. User 类

在 user 包中添加 User 类，因为在后面的注册程序中，需要在客户端与服务器之间传递 User 对象，因此 User 类要实现 Serializable 接口，代码如下。

```
public class User implements Serializable {
    private static final long serialVersionUID=-8418444131403945274L;
    private String userName;
    private String passWord;
    private String email;
    private int  level;
    private Date  regDate;
    public User(String userName, String passWord, String email, int level, Date
            regDate) {
        this.userName=userName;
```

```
        this.passWord=passWord;
        this.email=email;
        this.level=level;
        this.regDate=regDate;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email=email;
    }
    public String getPassWord() {
        return passWord;
    }
    public void setPassWord(String passWord) {
        this.passWord=passWord;
    }
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName=userName;
    }
    public int getLevel() {
        return level;
    }
    public void setLevel(int level) {
        this.level=level;
    }
    public Date getRegDate() {
        return regDate;
    }
    public void setRegDate(Date regDate) {
        this.regDate=regDate;
    }
    public String toString(){
        return userName+":"+passWord+":"+email+":"+level+":"+regDate;
    }
}
```

User类中的属性分别对应 user 表中的字段，这里，注册日期的类型 Date 是 java.util.Date。添加 toString()方法是为了在测试程序时，方便输出 User 对象的信息。

2. 用户管理类

在 user 包中创建用户管理类 UserDao，完成向 user 表添加用户以及在 user 表中根

据用户名和密码查找用户,代码如下。

```
public class UserDao {
    Connection conn=null;
    Statement st=null;
    PreparedStatement pstmt;
    ResultSet rs=null;
    public UserDao(){
        conn=new DBConnection().getConn();
    }
    public boolean addUser(User user){
        String userName=user.getUserName();
        String passWord=user.getPassWord();
        String email=user.getEmail();
        int level=user.getLevel();
        Date regDate=user.getRegDate();
        try {
            st=conn.createStatement();
            rs=st.executeQuery("select * from user where name='"+userName+"'");
            if(rs.next()){
                return false;
            }
            else{
                String sql="insert into user(userName,passWord,email,level,
                           regDate)"+"values(?,?,?,?,?)";
                pstmt=conn.prepareStatement(sql);
                pstmt.setString(1,userName);
                pstmt.setString(2, passWord);
                pstmt.setString(3, email);
                pstmt.setInt(4, level);
                pstmt.setDate(5, new java.sql.Date(regDate.getTime()));
                pstmt.executeUpdate();
                return true;
            }
        }
        catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
        finally{
            if(rs!=null){
                try {
                    rs.close();
                } catch (SQLException e) {
```

```
                e.printStackTrace();
            }
        }
        if(st!=null){
            try {
                st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
public User getUser(String userName, String passWord) {
    try {
        st=conn.createStatement();
        String sql="select * from user where name='"+userName+
                "' and passWord='"+passWord+"'";
        rs=st.executeQuery(sql);
        if(rs.next()){
            String email=rs.getString("email");
            int level=rs.getInt("level");
            Date regDate=rs.getDate("regDate");
            return new User(userName,passWord,email,level,regDate);
        }
        else{
            return null;
        }
    }
    catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
    finally{
        if(rs!=null){
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(st!=null){
            try {
                st.close();
            } catch (SQLException e) {
```

```
                    e.printStackTrace();
                }
            }
        }
    }
}
```

addUser(User user)方法向用户表添加一个用户。因为表中不允许有同名的用户，因此首先在用户表中查找和 user 同名的记录，如果找到，则不能添加，返回 false。如果找不到，则将 user 对象代表的用户数据添加到用户表中，返回 true。

getUser(String userName, String passWord)方法在用户表中查找参数指定的用户名和密码对应的记录，如果找到，则由这条记录的数据创建 User 对象，并返回，如果找不到，则返回 null。

3. 测试用户类和用户管理类

完成了 User 类和 UserDao 类之后，可以添加一个临时的测试类，测试 UserDao 类添加用户和查找用户的功能，例如下面的程序。

```
public class Test {
    public static void main(String[] args) {
        UserDao dao=new UserDao();
        User user=new User("test3","aaa","tetsts",1,new Date());
        if(dao.addUser(user))
            System.out.println("添加成功!");
        else
            System.out.println("添加失败!");
        User user2=dao.getUser("test3", "aaa");
        if(user2!=null){
            System.out.println("找到该用户!");
            System.out.println(user2);
        }else{
            System.out.println("未找到该用户!");
        }

        User user3=dao.getUser("test3", "dddd");
        if(user3!=null){
            System.out.println("找到该用户!");
        }else{
            System.out.println("未找到该用户!");
        }
    }
}
```

根据测试结果，对照数据库中的 user 表，如果有不合理的地方，则需要修改 User 类

或 UserDao 类，测试成功后，可将测试类 Test 删除。

3.3 用户注册和登录

在客户端提供一个注册和登录界面，用户填好数据后向服务器发送注册或登录命令，服务器处理后，将处理结果通知客户端。运行界面如图 3.2 所示。

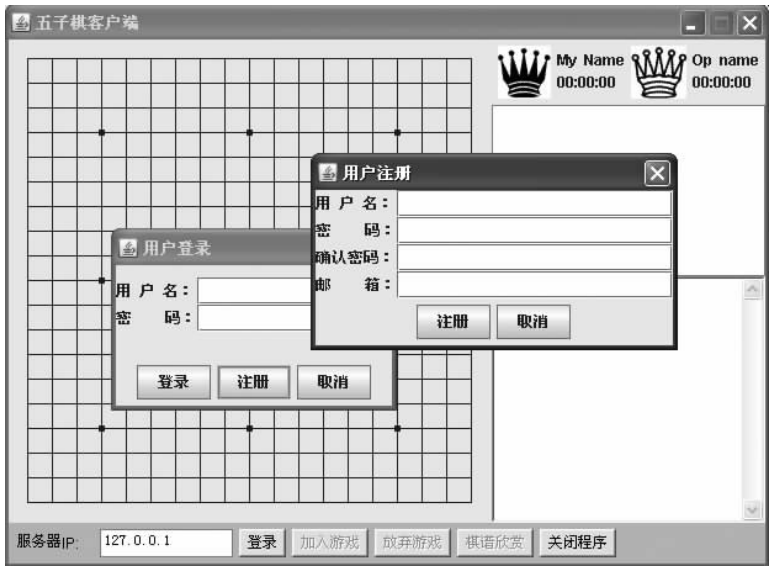

图 3.2　注册和登录界面

在客户端界面单击“登录”按钮，出现“用户登录”对话框，可以填写用户名和密码，单击“登录”按钮，向服务器发送 login 命令。如果用户还没有注册，可以单击“注册”按钮，在出现的“用户注册”对话框中输入相关信息后，单击“注册”按钮，向服务器发送 register 命令。

客户端与服务器之间发送命令的流程如图 3.3 所示。

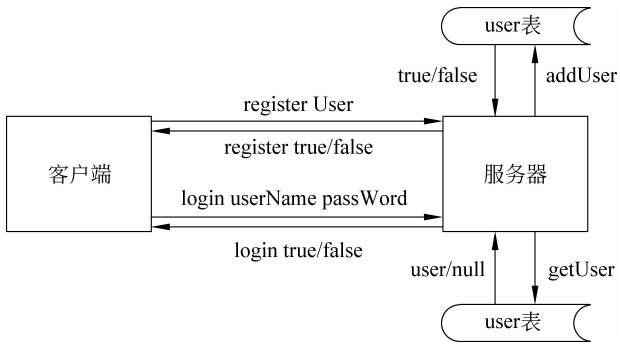

图 3.3　注册/登录流程

注册时，客户端发送 register 命令，参数是 User 对象，服务器通过 UserDao 类向用户

表添加记录。如果成功,则向客户端发送参数为 true 的 register 命令;如果失败,则向客户端发送参数为 false 的 register 命令。

登录时,客户端发送 login 命令,参数是用户名和密码,服务器通过 UserDao 类从用户表中查找用户名和密码指定的记录,如果找到,则登录成功,向客户端发送参数为 true 的 login 命令;如果没找到,则登录失败,向客户端发送参数为 false 的 login 命令。

3.3.1 准备工作

1. Command 类添加命令常量

在 Command 类中添加下面两个命令常量:

```
public static final String REGISTER="register";
public static final String LOGIN="login";
```

2. 修改 PanelControl 类

将 PanelControl 类中的“连接主机”按钮改为“登录”按钮,即将 connectButton 改为 loginButton,“连接主机”改为“登录”,然后将程序中的所有 connectButton 替换为 loginButton。

3. FiveClient 类中添加方法

由于在“用户登录”对话框类中要用到 FiveClient 类中的 Communication 属性 c,而登录对话框类和 FiveClient 类不在一个包中,因此需要在 FiveClient 类中添加 getC()方法,以便在其他类中获取 Communication 对象,代码如下。

```
public Communication getC(){
    return c;
}
```

3.3.2 用户登录

1. Communication 类的登录方法

由于“连接服务器”已经被“登录”代替了,因此可以删除 FiveClient 类中的 connect()方法。将 Communication 类中的 connect()方法改为 login()方法,代码如下:

```
public boolean login(String ip, String userName, String passWord) {
    try {
        s=new Socket(ip,FiveServer.TCP_PORT);
        dis=new DataInputStream(s.getInputStream());
        dos=new DataOutputStream(s.getOutputStream());
```

```
            dos.writeUTF(Command.LOGIN+":"+userName+":"+passWord);
            String msg=dis.readUTF();
            String[] words=msg.split(":");
            if(words[0].equals(Command.LOGIN) && words[1].equals("true")){
                fc.isConnected=true;
                new ReceaveThread(s).start();
                fc.control.exitGameButton.setEnabled(true);
                fc.control.loginButton.setEnabled(false);
                fc.control.joinGameButton.setEnabled(true);
                fc.control.cancelGameButton.setEnabled(false);
                return true;
            }
            else{
                s.close();
                return false;
            }
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            if(s!=null){
                try {
                    s.close();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
            e.printStackTrace();
        }
        return false;
    }
```

连接服务器后，向服务器发送 login 命令，然后等待服务器返回的结果。如果返回“login：true”，说明登录成功，为相关变量设置值，创建接收消息的线程并启动，设置按钮的状态；如果返回“login：false”，说明登录失败。

2. 登录对话框

在 user 包中创建登录对话框 DialogLogin 类，代码如下：

```
public class DialogLogin extends JDialog implements ActionListener{
    JTextField tfUserName=new JTextField(14);
    JTextField tfPassword=new JTextField(14);
    JButton jbLogin=new JButton("登录");
    JButton jbRegister=new JButton("注册");
    JButton jbCancel=new JButton("取消");
```

```
String ip;
FiveClient fc;
public DialogLogin(Window parent, String ip) {
    super(parent, "用户登录",Dialog.ModalityType.APPLICATION_MODAL);
    fc= (FiveClient) parent;
    this.ip=ip;
    createGUI();
}
public void actionPerformed(ActionEvent e) {
    String str=e.getActionCommand();
    if ("登录".equals(str)) {
        String userName=tfUserName.getText();
        String passWord=tfPassword.getText();
        if((userName==null)||(userName.isEmpty()) ||
            (passWord==null)||(passWord.isEmpty())){
            JOptionPane.showMessageDialog(this, "各项数据不能为空!");
            return;
        }
        else {
            if(fc.getC().login(ip, userName, passWord)){
                JOptionPane.showMessageDialog(this, "登录成功!");
                this.dispose();
            }
            else{
                JOptionPane.showMessageDialog(this, "用户名或密码不符!");
            }
        }
    }
    else if ("注册".equals(str)) {
        //DialogRegister rd=new DialogRegister(this,ip);
    }
    else if("取消".equals(str)){
        this.dispose();
    }
}
public void createGUI() {
    this.setLayout(new BorderLayout());
    JPanel jpWest=new JPanel();
    JPanel jpCenter=new JPanel();
    JPanel jpSouth=new JPanel();
    jpWest.setLayout(new GridLayout(3, 1));
    jpCenter.setLayout(new GridLayout(3, 1));
    jpSouth.setLayout(new FlowLayout());
    jpWest.add(new JLabel("用　户　名:"));
```

```
        jpWest.add(new JLabel("密          码:"));
        jpCenter.add(tfUserName);
        jpCenter.add(tfPassWord);
        this.add(new JPanel(),BorderLayout.NORTH);
        this.add(jpWest,BorderLayout.WEST);
        this.add(jpCenter,BorderLayout.CENTER);
        jpSouth.add(jbLogin);
        jpSouth.add(jbRegister);
        jpSouth.add(jbCancel);
        jbLogin.addActionListener(this);
        jbRegister.addActionListener(this);
        jbCancel.addActionListener(this);
        this.add(jpSouth,BorderLayout.SOUTH);
        this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        this.setLocation(450,250);
        this.pack();
        this.setVisible(true);
    }
}
```

由于对话框本身负责事件监听,因此DialogLogin类实现了ActionListener接口。为了便于看清程序的逻辑结构,我们将创建界面的代码放在了一个单独方法createGUI()中。在构造方法中设置对话框的父窗口(由参数指定)、标题以及显示模式,然后调用createGUI()方法创建对话框界面并显示。

在actionPerformed()方法中,处理3个按钮的单击事件。如果是“登录”按钮,则首先获取编辑框中的用户名和密码,如果用户名和密码都不为空,则调用Communication类中的login()方法登录,如返回true,则登录成功,返回false,则登录失败。

如果是“注册”按钮,则显示注册对话框,完成注册功能(注册功能的实现稍后给出)。如果是“取消”按钮,则使对话框消失。

3. 显示登录对话框

在fiveClient中修改登录按钮的监听程序段,显示登录对话框,代码如下:

```
else if(e.getSource()==control.loginButton){
    c=new Communication(FiveClient.this);
    DialogLogin dr=new DialogLogin(FiveClient.this, FiveClient.this.control
                    .inputIP.getText());
}
```

在创建DialogLogin对象时,要通过构造方法的参数指定对话框的父窗口和服务器的ip地址。

4. 服务器处理login命令

由于现在已经不需要服务器为用户起名了,因此可以删除服务器类的静态成员

clientNameNum，然后修改 startServer()方法。修改后的代码如下：

```
public void startServer(){
    try {
        ss=new ServerSocket(TCP_PORT);
        while(true){
            Socket s=ss.accept();
            InputStream is=s.getInputStream();
            OutputStream os=s.getOutputStream();
            DataInputStream dis=new DataInputStream(is);
            DataOutputStream dos=new DataOutputStream(os);
            String msg=dis.readUTF();
            String[] words=msg.split(":");
            if(words[0].equals(Command.LOGIN)){
                String userName=words[1];
                String passWord=words[2];
                UserDao ud=new UserDao();
                if(ud.getUser(userName, passWord)!=null){
                    dos.writeUTF(Command.LOGIN+":true");
                    clientNum++;
                    Client c=new Client(userName, s);
                    clients.add(c);
                    lStatus.setText("连接数"+clientNum);
                    taMessage.append(s.getInetAddress().getHostAddress()+" "
                                    +userName+"\n");
                    tellName(c);
                    addAllUserToMe(c);
                    addMeToAllUser(c);
                    new ClientThread(c).start();
                }
                else{
                    dos.writeUTF(Command.LOGIN+":false");
                    s.close();
                }
            }
        }
    } catch(IOException e) {
        e.printStackTrace();
    }
}
```

现在客户端连接服务器有注册和登录两个目的，因此服务器在接收到客户端连接时，要分别处理这两种情况。如果是 login 命令，则将用户名和密码取出，利用 UserDao 的 getUser()方法在 user 表中查找指定的用户，如果返回值不为空，则登录成功，向客户端

发送登录成功的命令，然后进行相应的处理，创建接收该客户端命令的线程并启动。为了不改变以前的程序结构，我们仍通过 tellName()方法向客户端发送客户名，当然这个客户名就是客户端传过来的名字。

如果登录失败，则向客户端发送登录失败的命令，然后将该 Socket 关闭。

3.3.3 用户注册

与登录后就准备下棋不同，注册是一个相对独立的功能，因此对于注册功能，我们不使用 Communication 类与服务器通信，而是在注册对话框类中添加自己的 Socket 属性实现与服务器的连接。

1. 注册对话框类

在 user 包中创建注册对话框 DialogRegister 类，代码如下：

```
public class DialogRegister extends JDialog implements ActionListener {
    JTextField tfUserName=new JTextField(20);
    JTextField tfPassword=new JTextField(20);
    JTextField tfRePassword=new JTextField(20);
    JTextField tfEmail=new JTextField(20);
    JButton jbRegister=new JButton("注册");
    JButton jbCancel=new JButton("取消");
    String ip;
    public DialogRegister(Window parent, String ip) {
        super(parent, "用户注册", Dialog.ModalityType.APPLICATION_MODAL);
        this.ip=ip;
        createGUI();
    }
    public void actionPerformed(ActionEvent e) {
        String str=e.getActionCommand();
        if ("注册".equals(str)) {
            String userName=tfUserName.getText();
            String passWord=tfPassWord.getText();
            String rePassWord=tfRePassWord.getText();
            String email=tfEmail.getText();
            if((userName==null)||(userName.isEmpty()) || (passWord==null)||
                (passWord.isEmpty()) || (rePassWord==null)||
                (rePassWord.isEmpty()) || (email==null)||(email.isEmpty())){
                JOptionPane.showMessageDialog(this, "各项数据不能为空!");
                return;
            }
            if (!(passWord.equals(rePassWord))) {
                JOptionPane.showMessageDialog(this, "两次密码不一致!");
```

```
                return;
            }
            else {
                if(register(new User(userName,passWord,email,1,new Date()))){
                    JOptionPane.showMessageDialog(this, "注册成功!");
                }
                else{
                    JOptionPane.showMessageDialog(this, "注册失败!可能重名。");
                }
            }
        }
        else if("取消".equals(str)){
            this.dispose();
        }
    }
    public boolean register(User u){
        Socket s=null;
        InputStream is=null;
        OutputStream os=null;
        DataInputStream dis=null;
        DataOutputStream dos=null;
        ObjectOutputStream oos=null;
        try {
            s=new Socket(ip, FiveServer.TCP_PORT);
            is=s.getInputStream();
            os=s.getOutputStream();
            dis=new DataInputStream(is);
            dos=new DataOutputStream(os);
            dos.writeUTF(Command.REGISTER);
            oos=new ObjectOutputStream(os);
            oos.writeObject(u);
            String msg=dis.readUTF();
            if(msg.equals(Command.REGISTER+":true")){
                return true;
            }
            else{
                return false;
            }
        } catch(UnknownHostException e) {
            e.printStackTrace();
            return false;
        } catch(IOException e) {
            e.printStackTrace();
            return false;
```

```
            }
            finally{
                try {
                    oos.close();
                    dos.close();
                    dis.close();
                    s.close();
                } catch(IOException e) {
                    e.printStackTrace();
                }
            }
        }
        public void createGUI() {
            this.setLayout(new BorderLayout());
            JPanel jpWest=new JPanel();
            JPanel jpCenter=new JPanel();
            JPanel jpSouth=new JPanel();
            jpWest.setLayout(new GridLayout(4, 1));
            jpCenter.setLayout(new GridLayout(4, 1));
            jpSouth.setLayout(new FlowLayout());
            jpWest.add(new JLabel("用  户  名:"));
            jpWest.add(new JLabel("密         码:"));
            jpWest.add(new JLabel("确认密码:"));
            jpWest.add(new JLabel("邮         箱:"));
            jpCenter.add(tfUserName);
            jpCenter.add(tfPassWord);
            jpCenter.add(tfRePassWord);
            jpCenter.add(tfEmail);
            this.add(jpWest, BorderLayout.WEST);
            this.add(jpCenter, BorderLayout.CENTER);
            jpSouth.add(jbRegister);
            jpSouth.add(jbCancel);
            jbRegister.addActionListener(this);
            jbCancel.addActionListener(this);
            this.add(jpSouth, BorderLayout.SOUTH);
            this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
            this.setLocation(450, 250);
            this.pack();
            this.setVisible(true);
        }
    }
```

在 actionPerformed()方法中,如果是“注册”按钮的事件,则先检查用户名、密码、重复密码以及邮箱是否有空的,如果没有空的再检查两次密码输入是否一致,如果一致,则

调用 register()方法注册，如果方法返回 true，表示注册成功，如果返回 false，表示注册失败。

在 register()方法中，首先连接服务器，随后向服务器发送 register 命令，接着再发一个 User 对象。之后，再从服务器读取字符串，如果是“register：true”，表示注册成功，如果是“register：false”，表示注册失败。

在登录对话框类 DialogLogin 中，通过单击“注册”按钮显示注册对话框。将原来类中注释掉的一行的注释去掉，代码如下：

```
else if ("注册".equals(str)) {
    DialogRegister rd=new DialogRegister(this,ip);
}
```

2. 服务器处理注册命令

在服务器类的 startServer()方法中，与登录代码并列，加入注册的处理代码如下：

```
if(words[0].equals(Command.REGISTER)){
    ObjectInputStream ois=null;
    try {
        ois=new ObjectInputStream(is);
        User u=(User) ois.readObject();
        UserDao ud=new UserDao();
        if(ud.addUser(u)){
            dos.writeUTF(Command.REGISTER+":true");
        }
        else{
            dos.writeUTF(Command.REGISTER+":false");
        }
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    finally{
        dis.close();
        ois.close();
        dos.close();
        s.close();
    }
}
```

如果服务器收到 register 命令，则继续读一个 User 对象，然后调用 UserDao 类的 addUser()方法向数据库中添加一个记录，如果返回值为 true，表明添加成功，则向客户端发送“register：true”，否则，向客户端发送“register：false”。

3.4 记录棋局和棋谱

下棋结束后，将棋局信息保存到数据库中，将棋谱信息保存到棋谱文件中。这里我们只记录连成五子赢棋后的棋局，对于认输和超时的情况没有记录，方法都是一样的。

记录棋局和棋谱的流程如图 3.4 所示，当客户端向服务器发送 win 命令后，服务器将棋局信息保存到数据库，同时也将棋谱信息保存到棋谱文件中。

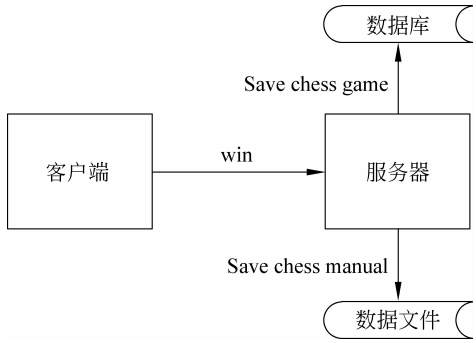

图 3.4 保存棋局和棋谱的流程图

3.4.1 记录棋局

1. 棋局管理

为了方便处理棋局数据，我们首先创建棋局类 Game，以及保存棋局和查找棋局的类 GameDao。

建立一个 game 包，在 game 包中创建 Game 类和 GameDao 类。

Game 类用于保存棋局数据，代码如下：

```
public class Game implements Serializable {
    private static final long serialVersionUID=-1301902854300541648L;
    String bUser;                                    //黑方用户
    String wUser;                                    //白方用户
    Date date;                                       //下棋日期, java.util.Date
    String winner;                                   //赢棋用户
    String fileName;                                 //保存棋谱的文件名
    public Game(String bUser, String wUser, Date date,String winner) {
        this.bUser=bUser;
        this.wUser=wUser;
        this.date=date;
        this.winner=winner;
        fileName=initFileName();
    }
```