CS 580 Client-Server Programming
Spring Semester, 2009
Assignment 3 Part 2 Comments
7 April, 2009

Submit Problems

- only class files, no source files
- no commits labeled part 2
- no files
- no files
- no commits for part 2

Extremes

131

Number of lines in longest Method

626

Number of lines in the largest class

1

Smallest number of classes

What Happens When login fails

```
public class StartState extends SDWitterState{

    public SDWitterState login(  String clientRequest, SDWitterServer parent)   {
        try {
            parent.login(clientRequest);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return new AuthState();
    }
```

Login for Previous Slide

```
public void login(String messageString) throws SQLException, IOException{
        int readIndex=messageString.indexOf(":");
        String userName= messageString.substring(readIndex+1, messageString.indexOf(";password",readIndex));
        readIndex=messageString.indexOf("d:");
        String userNamePassword=
                messageString.substring(readIndex+2, messageString.indexOf(";;",readIndex));
        Statement getTables = bismarck.createStatement();
        ResultSet tableList=getTables.executeQuery(
                "SELECT password FROM usernameinformation
                where usernameinformation.username='"+escapeRid(userName)+"';");
        if(tableList.next()){
                String tablePassword=tableList.getString(1);
                if (tablePassword.equals(escapeRid(userNamePassword))){
                        send("ok:success;;");
                        loggedUser=userName;
                }else{
                        send("error:Invalid password;;");
                }
        }
        else{
                send("error:No such user;;");
        }
}
```

Enough Tests?

```
public class SDWitterServerTest {
	@Test
	public void testescape() throws SQLException{
		SDWitterServer newServer=new SDWitterServer(8010,"MyLogFile.log");
		assertEquals("x\\:z",newServer.escape("x:z"));
		assertEquals("x\\;z",newServer.escape("x;z"));
		assertEquals("x\\\\z",newServer.escape("x\\z"));
	}

	@Test
	public void testescapeRid() throws SQLException{
		SDWitterServer newServer=new SDWitterServer(8010,"MyLogFile.log");
		assertEquals("x:z",newServer.escapeRid("x\\:z"));
		assertEquals("x;z",newServer.escapeRid("x\\;z"));
		assertEquals("x\\z",newServer.escapeRid("x\\\\z"));
	}

	@Test
	public void testupTo()throws IOException{
		UpToStream in=new UpToStream(new
ByteArrayInputStream("\u65e5;;".getBytes("UTF-8")));
		assertEquals("\u65e5;;",in.upTo());
	}
}
```

Good Enough?

```
public class TestUpToStream extends TestCase {
    UpToStream uts;


    public void testUpto() throws IOException {
        ByteArrayInputStream in = new ByteArrayInputStream("aa;;bb".getBytes());
        uts = new UpToStream(in);
        assertEquals(uts.upto(";;"), "aa;;");
    }
```

Test Boundary Conditions

```
public class TestUpToStream extends TestCase {
    UpToStream uts;

    public void testUpto() throws IOException {
        ByteArrayInputStream in = new ByteArrayInputStream("aa;;bb".getBytes());
        uts = new UpToStream(in);
        assertEquals(uts.upto(";;"), "aa;;");
        assertEquals(uts.upto(";;"), "bb");

        in = new ByteArrayInputStream("aa;;bb".getBytes());
        uts = new UpToStream(in);
        assertEquals(uts.upto("x"), "aa;;bb");

        in = new ByteArrayInputStream("".getBytes());
        uts = new UpToStream(in);
        assertEquals(uts.upto("x"), "");
    }
}
```

Constructor as Main

```
public ServerSDWitter() throws IOException{
        names = new Vector();
        loginName = new String();
        initialize();
        connectToDatabase();
        listen();
        loginState();
        acceptingCommandState();
}
```

What does this Do?

```
private void readFile(String text) {
        try {
                FileInputStream file = new FileInputStream(text);
                DataInputStream data = new DataInputStream(file);
                BufferedReader buffer = new BufferedReader(new InputStreamReader(data));
                String line;

                while ((line = buffer.readLine()) != null){
                        parseMessage(line);
                }

                data.close();
        } catch (FileNotFoundException e) {
                System.err.println("File not found");
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

Comments?

```
private void getMessagesFromDatabase() throws SQLException, IOException {
        java.sql.Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery("select messages from messagetable");
        String tmpString;
        if (rs != null){
                while (rs.next()){
                        tmpString = rs.getString("messages");
                        sendMessagesToClient(tmpString);
                }
                sendMessagesToClient(";");
        }
}

private void sendMessagesToClient(String tmpString) throws IOException {
        System.err.println(tmpString);
        String msgToClient = new String(tmpString.getBytes(), "UTF-8");
        output.write(msgToClient.getBytes());
        writeToLogfile(getDateAndTime()+" "+msgToClient);
}
```

What happens to socket?

```
private void listen(){
        try{
                serverSocket = new ServerSocket(PORT);
                socket = serverSocket.accept();
                if (serverSocket.isBound()){
                        System.err.println(serverSocket.getInetAddress()+" Connected");
                }
        }catch (IOException e) {
                System.out.println("Could not listen of port 6655");
        }
}
```

Configuration File

```
port=8012
logFile=logFromConfig.txt
```

HardCoded

```
public void open() {

    if (!testMode) {
        String dbUrl = "jdbc:mysql://localhost";
        String user = "foo";
        String pass = "bar";
        try {
            witter = DriverManager.getConnection(dbUrl, user, pass);

            Statement showTables = witter.createStatement();
            showTables.executeUpdate("USE cs580");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

}
```

Protocol

```
responseToClient = cmh.handle(receivedString);
out.write(responseToClient + "\r\n");
out.flush();
```

```
PrintWriter parsedOutput = new PrintWriter(this.socket.getOutputStream());
etc.
parsedOutput.println(command.getResponse());
parsedOutput.flush();
```

Rule of Thumb

Be lax on what you accept

Be strict on what you generate

State and...

```
public String handleTransmitMessages(String clientMessage) {
        String serverMessage = "error:Invalid command;;";
        String[] commTrim = clientMessage.split("transmitMessage:");

        if (commTrim[0].equals("") && commTrim.length == 2) {
                commTrim[1] = commTrim[1].substring(0, (commTrim[1].length() - 2));//Remove trailing
semicolons

                //Create new method to check if proper sql query is formed
                try {
                        queryToSend = "INSERT INTO messages (mtext, muser) VALUES('"
                                + commTrim[1] + "','" + state.getUser() + "');";

                        if (wdb.queryUpdate(queryToSend)) {
                                serverMessage = "ok:success;;";
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                        serverMessage = "error:Message not uploaded;;";
                }
        }
        return serverMessage;
}
```

State

```
public String handleNewUser(String clientMessage) {
        String serverMessage = "error:Invalid command;;";
        String[] commTrim = clientMessage.split(";screenName:|;password:");

        if (commTrim[0].equals("newUser") && commTrim.length == 3) {
                if (this.handleScreenName("screenName:" + commTrim[1] + ";;").equals("ok:available;;")) {
                        commTrim[2] = commTrim[2].substring(0, (commTrim[2].length() - 2));//Remove trailing
semicolons
                        queryToSend = "INSERT INTO users (uname, upass) VALUES('"
                                + commTrim[1] + "',PASSWORD('" + commTrim[2] + "'));";

                        if (wdb.queryUpdate(queryToSend)) {

                                state.logInUser(commTrim[1]);
                                return "ok:success;;";
                        }
                        return "error:Could not register user;;";
                }
                else {
                        serverMessage = "error:User already registered;;";
                }
        }
        return serverMessage;
}
```

Which one?

```
if (loginMatcher.matches()) {
        if (state.getState() == State.States.START.ordinal()) {
                Request loginRequest = new LoginRequest(inputLine, out);
                loginRequest.process();
        } else {
                logger.log(Level.WARNING, "Cannot proces request. Wrong state. Please login.");
        }
}
```

Verses

```
if (loginMatcher.matches()) {
        if (state.isLoggedIn()) {
                Request loginRequest = new LoginRequest(inputLine, out);
                loginRequest.process();
        } else {
                logger.log(Level.WARNING, "Cannot proces request. Wrong state. Please login.");
        }
}
```

Testing Private Methods

```
@Test
public void testGetUsername() throws ClassNotFoundException, SecurityException,
        NoSuchMethodException, IllegalArgumentException, IllegalAccessException,
        InvocationTargetException  {

    LoginRequest login =
        new LoginRequest("login;screenName:foo;password:password;;", null);
    Class<LoginRequest> c = LoginRequest.class;
    Method getUserName = c.getDeclaredMethod("getUserName", null);
    getUserName.setAccessible(true);
    String username = getUserName.invoke(login, null).toString();
    assertEquals(username, "foo");
    getUserName.setAccessible(false);
}
```

How to Change the File location?

```
        private static String dBconfigFile = "src\\resources\\databaseConfig.txt";

        private static DatabaseConnector instance = new
DatabaseConnector(dBconfigFile);
```

Good & Bad

```
final String addMessageSQl = "select * from messages where sender = ";
static final String insertMessageSQl =
      "insert into messages(messagestring,messageid,username) values( ";
final static String allMessagesSQl = "select messagestring from messages";
final static String someMessagesSQl =
      "select messagestring from messages where username = ";
```

Blocks of Messages

"SELECT * FROM messages WHERE user_id = ? LIMIT ? OFFSET ?"

"SELECT * FROM messages LIMIT ? OFFSET ?"

TimeStamp?

```
final static String allMessagesSQl = "select messagestring from messages";
```

Table Schema

Where documented?

The database uses the assigned PostgreSQL account and organizes the data into two tables as follows:

```
create table users (
    id INTEGER NOT NULL,
    username VARCHAR(20) NOT NULL,
    password VARCHAR(20) NOT NULL );
```

```
create table messages (
    user_id INTEGER NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    msg TEXT NOT NULL );
```

Program usage:

```
javac sdsu\cs580\SDwitter\*.java
java sdsu.cs580.SDwitter.SDwitterServerGUI [-p portNum] [-c configFile] [-l logFile]
```

Field as Local Variable

```
public void run() {
    try {
        getIOStreams();
        ClientRequest request;
        ServerResponse response;
        do {
            request = in.readClientRequest();
            response = handleClientRequest(request);
            out.writeMessage(response);
        } while (OkErrorResponse.QUIT_RESPONSE != response);
    etc



private void getIOStreams() throws IOException {
    in = new MessageInputStream(connection.getInputStream());
    out = new MessageOutputStream(connection.getOutputStream());
 }
```

Field as Local Variable

```
public void processRequest(InputStream in, OutputStream out) throws Exception
{
	state = new StartState();
	stream = new UpToStream(in,"UTF-8");
	do {
		message = new MessageReader(stream.upTo(";;"), database);
		log.receivedCommand(message.next());
		if(stream.isValid())
		{etc.
		}
		else
			break;
	}while(state.getState().compareTo("quit")!=0);
}
```

Field as Method Parameter Part 1

```
    public void start() {
        try
        {
            server = new ServerSocket(port);
            connection = server.accept();
            OSW = new OutputStreamWriter(connection.getOutputStream(), "UTF-8");
            ISR = new InputStreamReader(connection.getInputStream(),
"UTF-8");
            readStream = new UptoReader();
            setLogFile();
            serverLogger.log(Level.FINE, "Accepted connection from" +
connection.toString());
            trackStates();
        }
        catch(IOException io){}
    }
```

Field as Method Parameter Part 2

```
private void trackStates()
{
      SdServerState serverState = new NotAuthentic();
      int count = 0;
      do
      {
            try
            {
                  Message clientMessage = getNextCommand();
                  serverState = changeState(serverState, clientMessage);
            }
            catch(IlligalCommand i)
            {
                  count++ ;
                  sendMessages("Not authorized");
                  continue;
            }
      }while(isQuit(serverState) || count != 4);
}
```

Extra Blank Lines

```
private void trackStates()
{
        SdServerState serverState = new NotAuthentic();
        int count = 0;



        do
        {
                try
                {
                        Message clientMessage = getNextCommand();
                        serverState = changeState(serverState, clientMessage);
                }
                catch(IlligalCommand i)
                {
                        count++ ;
                        sendMessages("Not authorized");
                        continue;
                }


        }while(isQuit(serverState) || count != 4);



}
```

Configure logger to also use System.out

```
System.out.println("Request:"+request);
witterLog.log(Level.ALL,"[client "+clientSocket.getInetAddress()+" ] "+request);
```

Which is just for Testing?

```
public LoginHandler(String aString,Statement getTables) //only written for test case
{
	messageContents=parseInput(aString);
	resultResponse=handleRequest(getTables);
}

public LoginHandler(String aString)
{
	//	messageContents=parseInput(aString);
	testString=aString;
}
```

Two Queries - Slower and Wrong

```
ResultSet tableList = getLoginData.executeQuery("SELECT count(*) FROM usernames
        where userName = '"+messageContents[2]+"'");
while (tableList.next() )
{
    if(Integer.parseInt(tableList.getString(1))==0)
        return "error: User does not exist;;";
}

tableList = getLoginData.executeQuery("SELECT count(*) FROM usernames
    where password = '"+messageContents[4]+"'"); //Checks for the correct password
while (tableList.next() )
{
    if(Integer.parseInt(tableList.getString(1))==0)
        return "error: Invalid Password;;";
}
```

Files

a.txt

```
LogFile\ Location=/
Port=8010
```

b.txt

```
# Default Configuration
#Sat Mar 14 21:28:18 PDT 2009
LogFile\ Location=/
Port=8010
```

Dangerous?

```
public void clearTables() {
    try {
        dbHandler.clearTables();
    } catch (SQLException e) {
        LOGGER.throwing(CLASSNAME, "clearTables", e);
    }
}
```

Inside Thread handling requests

```
try {
    //keep CPU usage down since it doesn't block on a read
    Thread.sleep(100);
 } catch (InterruptedException ie) {}
```

LinkedBlockingQueue

put(E e)

Inserts the specified element at the tail of this queue,
waiting if necessary for space to become
available.

take()

Retrieves and removes the head of this queue,
waiting if necessary until an element becomes available.

poll(long timeout, TimeUnit unit)

Retrieves and removes the head of this queue,
waiting up to the specified wait time if necessary for
an element to become available.

Formating and ...

```
public void run()
{
        // the reply to the client is sent in UTF-8 encoding. the server replies
        // with the same string to each client.
        try
        {
                Messages msg = new Messages();
                String output = new String();


                        Writer out = new OutputStreamWriter(connection.getOutputStream( ), "UTF8");


                        StringBuffer buffer = new StringBuffer();
                        String command = new String();
                        boolean flag = false;


                                InputStreamReader isr = new InputStreamReader(connection.getInputStream(),
"UTF8");
                                Reader in = new BufferedReader(isr);
                                int ch = 0;
                                while (ch != 59 & flag == true)
```

Thread management

```
private void runServer() throws IOException {
    LOG.info("Server Started at port " + serverConfig.getPortNumber());
    while (true) {
        Socket connection = serverSocket.accept();
        LOG.info("Connection request from " + connection.getInetAddress());
        // REW you lose any reference to the instance
        // Student: For now that is not of importance.
        // REW but it will be an issue in part 3
        new ServerInstance(connection, serverConfig).start();
    }
}
```

Comment

```
public String getCommand()
{
        if(messageString.startsWith("login"))
                clientRequest = "login";

        else if(messageString.startsWith("newUser"))
                clientRequest = "newUser";

        else if(messageString.startsWith("screenName"))
                clientRequest = "screenName";

        else if(messageString.startsWith("transmitMessage"))
                clientRequest = "transmit";

        else if(messageString.startsWith("messages"))
                clientRequest = "messages";

        else if(messageString.startsWith("quit"))
                clientRequest = "quit";

        return clientRequest;
}
```

Is Reflection Worth it Here?

```
public boolean processCommand() throws Exception
{
        Method executeCommand;
        executeCommand = this.getClass().getMethod(clientRequest);
        command = (SDwitterMessage) executeCommand.invoke(this);
        return true;
}
```

Security

```
"SELECT user_id FROM users WHERE username='"+user+"' AND
      password=MD5('"+password+"')"
```

Names

```
temp = stripped.indexOf(";screenName:") + ";screenName:".length();
temp2 = stripped.indexOf(";password:", temp);
String username = line.substring(temp, temp2);
temp2 += ";password:".length();
```

Good Comments

```
public void run()
{
        while(!serverThread.isInterrupted())
        {
                try
                {
                        connection = serverSocket.accept();
                        ClientConnection clientConn = new ClientConnection(connection);

                        // FIXME Calling run directly rather than start.  Not trying to run a thread at
this point.
                        clientConn.run();
                }
                catch (SocketTimeoutException e)
                {
                        // Ignore
                }
                catch (IOException e)
                {
                        logger.severe("[" + hostAddress + "] Got IOException in run() " +
e.getMessage());
                }
        }
```