All of the Code - Hunter and Dennis

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**import** javax.swing.border.\*;

**public** **class** **Messenger** **extends** **JFrame**

{

**private** **static** **JFrame** *thisWindow*;

**private** **static** **Server** *server*;

**private** **JTextField** nameField;

**private** **JPasswordField** passwordField;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Constructor \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **Messenger**(**String** title, **Server** server)

{

**super**(title);

**this**.*server* = server;

*thisWindow* = **this**;

**JLabel** **nameLabel** = **new** JLabel("Login name:", **JLabel**.***RIGHT***);

nameField = **new** JTextField(20);

**LoginListener** **loginListener** = **new** LoginListener();

**JLabel** **passwordLabel** = **new** JLabel("Password:", **JLabel**.***RIGHT***);

passwordField = **new** JPasswordField(20);

passwordField.addActionListener(loginListener);

**JButton** **loginBtn** = **new** JButton("Login");

loginBtn.addActionListener(loginListener);

**JButton** **registerBtn** = **new** JButton("New user...");

registerBtn.addActionListener(**new** RegistrationListener());

**JPanel** **fieldsPanel** = **new** JPanel(**new** GridLayout(3, 3, 10, 10));

fieldsPanel.setBorder(**new** EmptyBorder(10, 10, 10, 10));

fieldsPanel.add(nameLabel);

fieldsPanel.add(nameField);

fieldsPanel.add(**new** JPanel()); // filler

fieldsPanel.add(**new** JLabel("Password:", **JLabel**.***RIGHT***));

fieldsPanel.add(passwordField);

fieldsPanel.add(**new** JPanel()); // filler

fieldsPanel.add(**new** JPanel()); // filler

fieldsPanel.add(loginBtn);

fieldsPanel.add(registerBtn);

**Container** **c** = getContentPane();

c.add(fieldsPanel);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* passwordField and "Login" button events handling \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**private** **class** **LoginListener** **implements** ActionListener

{

**public** **void** **actionPerformed**(**ActionEvent** e)

{

**String** **name** = nameField.getText().trim().toLowerCase();

**String** **password** = **String**.*valueOf*(passwordField.getPassword()).trim().toLowerCase();

tryLogin(name, password);

nameField.setText("");

passwordField.setText("");

}

**private** **void** **tryLogin**(**String** name, **String** password)

{

**String** **errorMsg** = "";

**int** **result** = *server*.login(name, password);

**if** (result < 0)

{

**if** (result == -1)

errorMsg = "User unknown";

**else** **if** (result == -2)

errorMsg = "Invalid password";

**else** **if** (result == -3)

errorMsg = "User already logged in";

**else**

errorMsg = "Unknown error code";

**JOptionPane**.*showMessageDialog*(*thisWindow*, errorMsg,

"Login failed", **JOptionPane**.***ERROR\_MESSAGE***);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\* "New user..." button events handling \*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**private** **class** **RegistrationListener** **implements** ActionListener

{

**public** **void** **actionPerformed**(**ActionEvent** e)

{

**while** (!registered());

}

**private** **boolean** **registered**()

{

**JLabel** **regNameLabel** = **new** JLabel("Login name (4-10 chars):", **JLabel**.***RIGHT***);

**JTextField** **regNameField** = **new** JTextField(20);

**JLabel** **regPasswordLabel** = **new** JLabel("Password (2-10 chars):", **JLabel**.***RIGHT***);

**JPasswordField** **regPasswordField** = **new** JPasswordField(20);

**JLabel** **regPasswordLabel2** = **new** JLabel("Confirm password:", **JLabel**.***RIGHT***);

**JPasswordField** **regPasswordField2** = **new** JPasswordField(20);

**JPanel** **fieldsPanel** = **new** JPanel();

fieldsPanel.setLayout(**new** GridLayout(3, 2, 10, 10));

fieldsPanel.setBorder(**new** EmptyBorder(10, 10, 10, 10));

fieldsPanel.add(regNameLabel);

fieldsPanel.add(regNameField);

fieldsPanel.add(regPasswordLabel);

fieldsPanel.add(regPasswordField);

fieldsPanel.add(regPasswordLabel2);

fieldsPanel.add(regPasswordField2);

**final** **String** **optionNames**[] = {"Register", "Cancel"};

**if** (**JOptionPane**.*showOptionDialog*(*thisWindow*, fieldsPanel,

"New user registration", **JOptionPane**.***OK\_CANCEL\_OPTION***,

**JOptionPane**.***PLAIN\_MESSAGE***, **null**, optionNames,

optionNames[0]) != 0)

**return** **true**; // User pressed "Cancel"

**String** **name** = regNameField.getText().trim().toLowerCase();

**String** **password** = **String**.*valueOf*(regPasswordField.getPassword()).trim().toLowerCase();

**String** **password2** = **String**.*valueOf*(regPasswordField2.getPassword()).trim().toLowerCase();

**String** **errorMsg** = "";

**int** **result** = password.compareTo(password2);

**if** (result != 0)

{

errorMsg = "Passwords mismatch, re-enter";

}

**else**

{

result = *server*.addUser(name, password);

**if** (result < 0)

{

**if** (result == -1)

errorMsg = "Login name must be 4-10 characters long";

**else** **if** (result == -2)

errorMsg = "Password must be 2-10 characters long";

**else** **if** (result == -3)

errorMsg = "Login name already taken, choose another one";

**else**

errorMsg = "Unknown error code";

}

}

**if** (result != 0)

{

**JOptionPane**.*showMessageDialog*(*thisWindow*, errorMsg,

"Registration failed", **JOptionPane**.***ERROR\_MESSAGE***);

nameField.setText("");

passwordField.setText("");

**return** **false**;

}

**else**

{

**JOptionPane**.*showMessageDialog*(*thisWindow*, "Added " + name,

"Registration successful", **JOptionPane**.***INFORMATION\_MESSAGE***);

nameField.setText(name);

passwordField.setText(password);

**return** **true**;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* main \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **static** **void** **main**(**String**[] args)

{

*server* = **new** Server();

*server*.addUser("drgodwin", "physics");

*server*.addUser("drhendrick", "land");

*server*.addUser("drwagner", "chem");

*server*.addUser("drsalazar", "math");

**Messenger** **window** = **new** Messenger("Java Messanger", *server*);

window.addWindowListener(**new** WindowAdapter()

{ **public** **void** **windowClosing**(**WindowEvent** e) { **System**.*exit*(0); }});

window.setBounds(0, 0, 360, 140);

window.setVisible(**true**);

}

}

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**import** java.util.Set;

**public** **class** **MsgWindow** **extends** **JFrame**

**implements** KeyListener

{

**private** **static** **final** **String** ***PROMPT*** = ">> ";

**private** **MsgUser** myUser;

**private** **JComboBox** buddiesList;

**private** **JTextArea** textArea;

**public** **MsgWindow** (**MsgUser** u, Set buddies)

{

**super**(u.toString());

addWindowListener(**new** WindowAdapter()

{ **public** **void** **windowClosing**(**WindowEvent** e) { myUser.quit(); }});

myUser = u;

**Object** **buddiesArray**[] = buddies.toArray();

buddiesList = **new** JComboBox(buddiesArray);

**JPanel** **talkTo** = **new** JPanel(**new** FlowLayout(**FlowLayout**.***RIGHT***));

talkTo.add(**new** JLabel("Talk to:", **JLabel**.***RIGHT***));

talkTo.add(buddiesList);

textArea = **new** JTextArea(10, 20);

textArea.setFont(**new** Font("Serif", **Font**.***PLAIN***, 20));

textArea.setLineWrap(**true**);

textArea.setWrapStyleWord(**true**);

textArea.append(***PROMPT***);

textArea.addKeyListener(**this**);

**JScrollPane** **areaScrollPane** = **new** JScrollPane(textArea);

areaScrollPane.setVerticalScrollBarPolicy(

**JScrollPane**.***VERTICAL\_SCROLLBAR\_ALWAYS***);

**Container** **c** = getContentPane();

c.add(talkTo, **BorderLayout**.***NORTH***);

c.add(areaScrollPane, **BorderLayout**.***CENTER***);

**int** **x** = (**int**)(**Math**.*random*()\* 500);

**int** **y** = (**int**)(**Math**.*random*()\* 300);

setBounds(x, y, 300, 300);

setVisible(**true**);

}

**public** **void** **addBuddy**(**MsgUser** u)

{

buddiesList.addItem(u);

}

**public** **void** **removeBuddy**(**MsgUser** u)

{

buddiesList.removeItem(u);

}

**public** **void** **showMessage**(**String** text)

{

textArea.append(text);

textArea.append("\n" + ***PROMPT***);

}

**private** **void** **sendMessage**(**String** text)

{

**MsgUser** **u** = (**MsgUser**)buddiesList.getSelectedItem();

u.receiveMessage("[" + myUser.toString() + "] " + text);

textArea.append(***PROMPT***);

}

/\*\*

\* Implement KeyListener interface:

\*/

**public** **void** **keyReleased** (**KeyEvent** e)

{

**String** **msg**;

**int** **code** = e.getKeyCode();

**switch**(code)

{

**case** **KeyEvent**.***VK\_ENTER***:

msg= textArea.getText();

**int** **tail** = msg.lastIndexOf(***PROMPT***) + ***PROMPT***.length();

msg = msg.substring(tail).trim();

**if** (msg.length() > 0)

sendMessage(msg);

**break**;

}

}

**public** **void** **keyPressed** (**KeyEvent** e) {}

**public** **void** **keyTyped** (**KeyEvent** e) {}

}

**import** java.util.Set;

/\*\*

\* Msg User is a class that stores the

\* username, password, and server its on.

\* Users are able to message eachothers

\* **@authors** Hunter Damron and Dennis Perea

\* //MsgUser.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**public** **class** **MsgUser** **implements** Comparable<MsgUser> {

**private** **String** name;

**private** **String** password;

**private** **Server** server;

**private** **MsgWindow** myWindow;

/\*\*

CONSTRUCTOR: saves a reference to the server and initializes this user’s

screen name and password

\*/

**public** **MsgUser**(**Server** server, **String** name, **String** password) {

**this**.server = server;

**this**.name = name;

**this**.password = password;

}

// METHODS:

/\*\*

returns this user’s screen name

\*/

**public** **String** **toString**() {

**return** name;

}

/\*\*

returns this user’s password

\*/

**public** **String** **getPassword**() {

**return** password;

}

/\*\*

returns true if this user’s name is equal to other’s (case insensitive), false otherwise

\*/

**public** **boolean** **equals** (**MsgUser** other) {

**if**(**this**.name.equalsIgnoreCase(other.name))

**return** **true**;

**else** **return** **false**;

}

/\*\*

Compares user’s screen name to other’s screen name, case insensitive.

Note: we have implemented Comparable<MsgUser> so no need to make other

an Object type

\*/

**public** **int** **compareTo**(**MsgUser** other) {

**return** **this**.name.compareToIgnoreCase(other.name);

}

/\*\*

Only needed for the extra credit – commented out until you get to that point

\*/

// public int hashCode() { }

/\*\*

Creates a dialog window passing this user and the buddies set to its constructor.

Saves a reference to the new dialog window in the myWindow field.

\*/

**public** **void** **openDialog**(Set<MsgUser> buddies) {

myWindow = **new** MsgWindow(**this**, buddies);

}

/\*\*

If myWindow is initialized, adds u to this user’s ‘buddy list’ by

calling myWindow.addBuddy(u);

\*/

**public** **void** **addBuddy**(**MsgUser** u) {

//addBuddy() refers to the addBuddy method in MsgWindow

**if**(myWindow != **null**)

myWindow.addBuddy(u);

}

/\*\*

If myWIndow is initialized, removes u from this user’s ‘buddy list’ by calling

myWindow.removeBuddy(u)

\*/

**public** **void** **removeBuddy**(**MsgUser** u) {

//removeBuddy() refers to the removeBuddy method in MsgWindow

**if**(myWindow != **null**)

myWindow.removeBuddy(u);

}

/\*\*

if myWindow is initialized, shows text by calling myWindow.showMessage(text)

\*/

**public** **void** **receiveMessage**(**String** text) {

**if**(myWindow != **null**)

myWindow.showMessage(text);

}

/\*\*

Disposes of this user’s dialog window. Logs out this user by calling server’s logout method.

(This method is called from the MsgWindow class when the ‘close’ button is clicked on the dialog window.)

\*/

**public** **void** **quit**() {

// dispose() closese the window

myWindow.dispose();

//LogOut removes the user from the active list

server.logout(**this**);

}

}

/\*\*

\* Server manages registered MsgUsers and logged in MsgUsers

\* **@authors** Hunter Damron and Dennis Perea

\* //Server.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**import** java.util.TreeMap;

**import** java.util.TreeSet;

/\*\*

fill in the comment section

\*/

**public** **class** **Server**

{

**private** **TreeMap**<String, MsgUser> users;

**private** **TreeSet**<MsgUser> loggedIn;

/\*\*

CONSTRUCTOR:

Initializes the map of registered users and the set of logged-in users to be empty

\*/

**public** **Server**() {

users = **new** TreeMap<String, MsgUser>();

loggedIn = **new** TreeSet<MsgUser>();

}

// METHODS:

/\*\*

Registers a new user with a given screen name and password. Returns 0 if the

registration is successful. Returns one of the following negative integer codes,

if the registration failed.

-1 invalid screen name, must be from 4 to 10 characters in length

-2 invalid password, must be from 2 to 10 characters in length

-3 the screen name is already taken

\*/

**public** **int** **addUser**(**String** name, **String** password) {

**if**(name.length() < 4 || name.length() > 10)

**return** -1;

**if**(password.length() < 2 || password.length() > 10)

**return** -2;

**if**(users.containsKey(name))

**return** -3;

users.put(name, **new** MsgUser(**this**, name, password));

**return** 0;

}

/\*\*

Logs in a new user with a given screen name and password. Returns 0, if successful,

and a negative integer error code, if failed.

Error codes:

-1 user not found

-2 invalid password

-3 user is already logged in

This method creates a new MsgUser object and adds it to the “buddy lists” of all

previously logged-in users by calling their addBuddy method. If opens a dialog

window for this user by calling its openDialog method and passing all previously

logged-in users to it as a “buddy list.” It then adds the new user to the set of

logged-in users.

\*/

**public** **int** **login**(**String** name, **String** password) {

**if**(!users.containsKey(name))

**return** -1;

**MsgUser** **attempt** = users.get(name);

**if**(!password.equals(attempt.getPassword()))

**return** -2;

**if**(loggedIn.contains(attempt))

**return** -3;

**for**(**MsgUser** **entry** : loggedIn) {

entry.addBuddy(attempt);

}

loggedIn.add(attempt);

attempt.openDialog(loggedIn);

**return** 0;

}

/\*\*

Removes a given user from the set of logged-in users and from the ‘buddy lists’ of

all other logged-in users.

\*/

**public** **void** **logout**(**MsgUser** u) {

loggedIn.remove(u);

**for**(**MsgUser** **entry** : loggedIn) {

entry.removeBuddy(u);

}

}

}

For Hashing Extra Credit:

/\*\*

\* Msg User is a class that stores the

\* username, password, and server its on.

\* Also Implements hashCode() for use in HashSet

\* **@authors** Hunter Damron and Dennis Perea

\* //hashedVersion/MsgUser.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**package** hashedVersion;

**import** java.util.Set;

/\*\*

\* fill in the comment block - author, etc.

\*/

**public** **class** **MsgUser** **implements** Comparable<MsgUser> {

**private** **String** name;

**private** **String** password;

**private** **Server** server;

**private** **MsgWindow** myWindow;

/\*\*

CONSTRUCTOR: saves a reference to the server and initializes this user’s

screen name and password

\*/

**public** **MsgUser**(**Server** server, **String** name, **String** password) {

**this**.server = server;

**this**.name = name;

**this**.password = password;

}

// METHODS:

/\*\*

returns this user’s screen name

\*/

**public** **String** **toString**() {

**return** name;

}

/\*\*

returns this user’s password

\*/

**public** **String** **getPassword**() {

**return** password;

}

/\*\*

returns true if this user’s name is equal to other’s (case insensitive), false otherwise

\*/

**public** **boolean** **equals** (**MsgUser** other) {

**if**(**this**.name.equalsIgnoreCase(other.name))

**return** **true**;

**else** **return** **false**;

}

/\*\*

Compares user’s screen name to other’s screen name, case insensitive.

Note: we have implemented Comparable<MsgUser> so no need to make other

an Object type

\*/

**public** **int** **compareTo**(**MsgUser** other) {

**return** **this**.name.compareToIgnoreCase(other.name);

}

/\*\*

Only needed for the extra credit – commented out until you get to that point

\*/

**public** **int** **hashCode**() {

**return** name.hashCode();

}

/\*\*

Creates a dialog window passing this user and the buddies set to its constructor.

Saves a reference to the new dialog window in the myWindow field.

\*/

**public** **void** **openDialog**(Set<MsgUser> buddies) {

myWindow = **new** MsgWindow(**this**, buddies);

}

/\*\*

If myWindow is initialized, adds u to this user’s ‘buddy list’ by

calling myWindow.addBuddy(u);

\*/

**public** **void** **addBuddy**(**MsgUser** u) {

**if**(myWindow != **null**)

myWindow.addBuddy(u);

}

/\*\*

If myWIndow is initialized, removes u from this user’s ‘buddy list’ by calling

myWindow.removeBuddy(u)

\*/

**public** **void** **removeBuddy**(**MsgUser** u) {

**if**(myWindow != **null**)

myWindow.removeBuddy(u);

}

/\*\*

if myWindow is initialized, shows text by calling myWindow.showMessage(text)

\*/

**public** **void** **receiveMessage**(**String** text) {

**if**(myWindow != **null**)

myWindow.showMessage(text);

}

/\*\*

Disposes of this user’s dialog window. Logs out this user by calling server’s logout method.

(This method is called from the MsgWindow class when the ‘close’ button is clicked on the dialog window.)

\*/

**public** **void** **quit**() {

myWindow.dispose();

server.logout(**this**);

}

}

/\*\*

\* Manages registered and logged in MsgUsers

\* using HashSets and HashMaps

\* **@authors** Hunter Damron and Dennis Perea

\* //hashedVersion/Server.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**package** hashedVersion;

**import** java.util.HashMap;

**import** java.util.HashSet;

/\*\*

fill in the comment section

\*/

**public** **class** **Server**

{

**private** **HashMap**<String, MsgUser> users;

**private** **HashSet**<MsgUser> loggedIn;

/\*\*

CONSTRUCTOR:

Initializes the map of registered users and the set of logged-in users to be empty

\*/

**public** **Server**() {

users = **new** HashMap<String, MsgUser>();

loggedIn = **new** HashSet<MsgUser>();

}

// METHODS:

/\*\*

Registers a new user with a given screen name and password. Returns 0 if the

registration is successful. Returns one of the following negative integer codes,

if the registration failed.

-1 invalid screen name, must be from 4 to 10 characters in length

-2 invalid password, must be from 2 to 10 characters in length

-3 the screen name is already taken

\*/

**public** **int** **addUser**(**String** name, **String** password) {

**if**(name.length() < 4 || name.length() > 10)

**return** -1;

**if**(password.length() < 2 || password.length() > 10)

**return** -2;

**if**(users.containsKey(name))

**return** -3;

users.put(name, **new** MsgUser(**this**, name, password));

**return** 0;

}

/\*\*

Logs in a new user with a given screen name and password. Returns 0, if successful,

and a negative integer error code, if failed.

Error codes:

-1 user not found

-2 invalid password

-3 user is already logged in

This method creates a new MsgUser object and adds it to the “buddy lists” of all

previously logged-in users by calling their addBuddy method. If opens a dialog

window for this user by calling its openDialog method and passing all previously

logged-in users to it as a “buddy list.” It then adds the new user to the set of

logged-in users.

\*/

**public** **int** **login**(**String** name, **String** password) {

**if**(!users.containsKey(name))

**return** -1;

**MsgUser** **attempt** = users.get(name);

**if**(!password.equals(attempt.getPassword()))

**return** -2;

**if**(loggedIn.contains(attempt))

**return** -3;

**for**(**MsgUser** **entry** : loggedIn) {

entry.addBuddy(attempt);

}

loggedIn.add(attempt);

attempt.openDialog(loggedIn);

**return** 0;

}

/\*\*

Removes a given user from the set of logged-in users and from the ‘buddy lists’ of

all other logged-in users.

\*/

**public** **void** **logout**(**MsgUser** u) {

loggedIn.remove(u);

**for**(**MsgUser** **entry** : loggedIn) {

entry.removeBuddy(u);

}

}

}