```
package rpn;

import java.awt.*;
import java.net.*;
import java.io.*;

/** Things that can go wrong:
    1.  Server not running, or not on the right port.
    2.  Student identification invalid
    3.  Can't write to correct file
    4.  Can't read from correct file
 */

abstract class Submittor implements Runnable {
    MessageFrame messageFrame = null;
    String host;
    int port;
    SubmittorPanel sc;
    ServerConnection serverConn;

    /* if id = null, waits until registered
       then sends message:
        save/load byte + strings
        lab\n
        exercise\n
        login id\n
        student number\n
        [+ program text]
      and waits for a reply:
          status byte
          save/load byte
          [+ program text]
      up to some time out */

    public Submittor(String host, int port, SubmittorPanel sc, String t) {
        this.sc = sc;
        this.host = host;
        this.port = port;
        messageFrame = new MessageFrame(this, t);
    }

    public void open() {
        try {
            serverConn = new ServerConnection(host, port);
        } catch (Exception e) {
            messageFrame.messageField.setText("Submittor:  " + e.toString());
        }
    }

    public abstract void run();
    public abstract void stop();

    public synchronized void finalize() {
        System.out.println("Submittor:  closing!");
        try {
            serverConn.finalize();
        } catch(IOException e) {
```

```
        }
    }
}

class Saver extends Submittor {
    String text;
    Thread p;
    public Saver(String host, int port, SubmittorPanel sc) {
        super(host, port, sc, "Save");
        text = sc.inputArea.getText();
        p = new Thread(this);
        p.start();
    }

    public void run() {
        open();
    }

    public void stop() {
        p.stop();
    }

    public void open() {
        super.open();
        try {
            serverConn.write(server.SubmitServer.Checking);
            int OK = serverConn.read();
            System.out.println("saver:  ok?  " + (char) OK);

            serverConn.write(server.SubmitServer.Save);
            serverConn.write(sc.loginId + '\n');
            serverConn.write(sc.studentNumber + '\n');
            serverConn.write(sc.lab + '\n');
            serverConn.write(sc.exercise + '\n');
            serverConn.write(text);
            serverConn.write((byte) 0);
            messageFrame.messageField.setText("Saving program ...");
            int b = serverConn.read();
            System.out.println("saver:  b = " + b);
            if (b == server.SubmitServer.Save) {
                messageFrame.messageField.setText("Program saved!");
            } else {
                messageFrame.messageField.setText("Unable to save program!");
            }
        } catch (Exception e) {
            System.out.println("Oops!");
            messageFrame.messageField.setText(e.toString());
        }
        try {
            serverConn.finalize();
        } catch(Exception e) {
            serverConn = null;
        }
    }
}
```

```
class Loader extends Submittor {
    Thread p;
    public Loader(String host, int port, SubmittorPanel sc) {
        super(host, port, sc, "Save");

        p = new Thread(this);
        p.start();
    }

    public void open() {
        super.open();
        try {
            serverConn.write(server.SubmitServer.Checking);
            int OK = serverConn.read();
            System.out.println("loader:  ok?  " + (char) OK);

            serverConn.write(server.SubmitServer.Load);
            serverConn.write(sc.loginId + '\n');
            serverConn.write(sc.studentNumber + '\n');
            serverConn.write(sc.lab + '\n');
            serverConn.write(sc.exercise + '\n');
            serverConn.write((byte) 0);
            messageFrame.messageField.setText("Loading program ...");
            int b = serverConn.read();
            System.out.println("loader:  b = " + b);
            StringBuffer text = new StringBuffer();
            if (b == server.SubmitServer.Load) {
                while ((b = serverConn.read()) != 0) {
                    text.append((char) b);
                }
                // text.append("***");
                sc.inputArea.setText(new String(text));
                messageFrame.messageField.setText("Program loaded!");
            } else {
                messageFrame.messageField.setText("Unable to load program!");
            }
        } catch (Exception e) {
            System.out.println("Oops!");
            messageFrame.messageField.setText(e.toString());
        }
        try {
            serverConn.finalize();
        } catch(Exception e) {
            serverConn = null;
        }
    }

    public void run() {
        open();
    }

    public void stop() {
        p.stop();
    }
}

class MessageFrame extends Frame {
```

```
    Submittor s;
    private TextArea ta;
    TextField lid;
    Button okButton;
    TextField messageField;

    public MessageFrame(Submittor s, String t) {
        super(t);
        Font f = new Font("Helvetica", Font.BOLD, 14);
        setFont(f);
        this.s = s;
        setLayout(new GridLayout(2, 1));
        Panel p = new Panel();
        p.setLayout(new BorderLayout());
        p.add("West", new Label("Login id:"));
        lid = new TextField(12);
        lid.setText(s.sc.loginId);
        p.add("East", lid);
        add(p);

        p = new Panel();
        messageField = new TextField(36);
        messageField.setEditable(false);
        p.add(messageField);
        add(p);

        pack();
        show();
    }

    public boolean handleEvent(Event evt) {
        if (evt.id == Event.WINDOW_DESTROY) {
            dispose();
            s.stop();
            s.sc.submittor = null;
        }
        return(super.handleEvent(evt));
    }
}

// ---------------------------------------------------------------------------

public class SubmittorPanel extends CalculatorPanel {
    Button saveButton, loadButton;
    int port;
    String host;
    String lab, exercise, name;

    public SubmittorPanel(int r, String s, String t,
            ItemInterpreter it, ItemInstaller ci,
            URL loadBase, int port,
            String lab, String exercise, String name) {
        super(r, s, t, it, ci, loadBase);
        this.port = port;
        this.lab = lab;
        this.exercise = exercise;
        this.name = name;
```

```
        this.host = loadBase.getHost();
    }

    Panel middlePanel() {
        Panel instPanel = new Panel();
        instPanel.setLayout(new GridLayout(1, 3));

        Label instLabel = new Label("Current:  ");
        instLabel.setFont(labelfont);
        instPanel.add(instLabel);

        instField = new TextField(8);
        instField.setEditable(false);
        instField.setBackground(grey);
        instField.setFont(figFont);
        instPanel.add(instField);

        Panel bp = new Panel();
        bp.setLayout(new GridLayout(1, 2));
        saveButton = new Button("Save");
        saveButton.setFont(labelfont);
        bp.add(saveButton);
        loadButton = new Button("Load");
        loadButton.setFont(labelfont);
        bp.add(loadButton);
        instPanel.add(bp);

        return(instPanel);
    }


    String loginId = "cass";
    String studentNumber = "1";
    boolean registered = true;
    /* String loginId = null;
    String studentNumber = null;
    boolean registered = false; */

    void setId(String loginId, String studentNumber) {
        // get name and number
        this.loginId = loginId;
        this.studentNumber = studentNumber;
        System.out.println(loginId);
        System.out.println(studentNumber);
        registered = true;
     }

     Submittor submittor = null;

     void save() {
        if (submittor == null) submittor = new Saver(host, port, this);
     }

      void load() {
        if (submittor == null) submittor = new Loader(host, port, this);
     }
```

```
     final static int Save = 0, Load = 1;
     LoginFrame lf = null;

    public boolean handleEvent(Event evt) {
        if (evt.target == saveButton
                && evt.id == Event.ACTION_EVENT) {
            if (!registered && lf == null) {
                LoginFrame lf = new LoginFrame(host, port,
                    this, title + ":  Registration", Save);
            } else {
                save();
            }

        } else if (evt.target == loadButton
                && evt.id == Event.ACTION_EVENT) {
            if (!registered && lf == null) {
                LoginFrame lf = new LoginFrame(host, port,
                    this, title + ":  Registration", Load);
            } else {
                load();
            }
        }
        return(super.handleEvent(evt));
    }
}
```

```
package rpn;

import java.awt.*;
import java.net.*;
import java.io.*;



class ServerConnection {
    protected Socket serverConn;
    InputStream in;
    OutputStream out;

    public ServerConnection(String host, int port)
        throws IOException, UnknownHostException {

        try {
            System.out.println("Trying " + host + " " + port);
            serverConn = new Socket(host, port);
        } catch(UnknownHostException e) {
            System.out.println("Submittor:  " + e);
            throw new UnknownHostException();
        }
        catch (IOException e) {
            System.out.println("Submittor:  " + e);
            throw new IOException();
        }
        try {
            out = serverConn.getOutputStream();
        } catch (IOException e) {
            System.out.println("Submittor:  " + e);
            throw new IOException();
        }
        try {
            in = serverConn.getInputStream();
        } catch (IOException e) {
            System.out.println("Submittor:  " + e);
            throw new IOException();
        }
        System.out.println("Connection established!");
    }

    public synchronized void finalize() throws IOException {
        System.out.println("Connection:  closing!");
        serverConn.close();
    }

    void write(byte b) throws IOException {
        out.write(b);
    }

    void write(String s) throws IOException {
        System.out.println("Writing " + s);
        for (int i=0;i<s.length();i++) {
            out.write(s.charAt(i));
        }
    }
```

```
    public int read() throws IOException {
        return(in.read());
    }
}
```

```
package rpn;

import java.awt.*;
import java.net.*;
import java.io.*;

public class LoginFrame extends Frame {
    SubmittorPanel s;
    TextField lid, sid;
    Button registerButton;
    TextField messageField;
    String host; int port;
    int mode;

    public LoginFrame(String host, int port, SubmittorPanel s, String t, int mode) {
        super(t);
        Font f = new Font("Helvetica", Font.BOLD, 14);
        setFont(f);
        this.s = s;
        this.host = host;
        this.port = port;
        this.mode = mode;

        setLayout(new GridLayout(4, 1));
        Panel p = new Panel();
        p.setLayout(new BorderLayout());
        p.add("West", new Label("Login id:"));
        lid = new TextField(12);
        p.add("East", lid);
        add(p);

        p = new Panel();
        p.setLayout(new BorderLayout());
        p.add("West", new Label("Student number:"));
        sid = new TextField(12);
        sid.setEchoCharacter('*');
        p.add("East", sid);
        add(p);

        p = new Panel();
        registerButton = new Button("Register");
        p.add(registerButton);
        add(p);

        p = new Panel();
        messageField = new TextField(36);
        // messageField.setEditable(false);
        p.add(messageField);
        add(p);

        pack();
        show();
    }

    ServerConnection serverConn = null;
    public boolean handleEvent(Event evt) {
        if (evt.target == registerButton
```

```
            && evt.id == Event.ACTION_EVENT) {
        // transmit name and number
        boolean valid = false;
        try {
            serverConn = new ServerConnection(host, port);
        } catch (UnknownHostException e) {
            messageField.setText(e.toString());
            serverConn = null;
        } catch (IOException e) {
            messageField.setText(e.toString());
            serverConn = null;
        }
        if (serverConn != null) {
            // validate id
            try {
                serverConn.write(server.SubmitServer.Checking);
                int reply = serverConn.read();
                System.out.println("login:  ok?  " + (char) reply);
                serverConn.write(server.SubmitServer.Validate);
                serverConn.write(lid.getText() + '\n');
                serverConn.write(sid.getText() + '\n');
                reply = serverConn.read();
                System.out.println("reply = " + (char) reply);
                if (reply == server.SubmitServer.Valid) {
                    messageField.setBackground(Color.white);
                    messageField.setForeground(Color.black);
                    messageField.setText("Valid identification");
                    s.setId(lid.getText(), sid.getText());
                    if (mode == s.Save) {
                        serverConn.finalize();
                        dispose();
                        s.save();
                    } else {
                        serverConn.finalize();
                        dispose();
                        s.load();
                    }
                } else {
                    messageField.setBackground(Color.red);
                    messageField.setForeground(Color.white);
                    messageField.setText("Enter valid identification or quit");
                }
            } catch (IOException e) {
                System.out.println("Unable to get reply!");
            }
        }
    } else if (evt.id == Event.WINDOW_DESTROY) {
        dispose();
        s.lf = null;
    }
    return(super.handleEvent(evt));
  }
}
```