

Section C

C1: Good Programming Style

```
/*=====
Program:  Volunteerist.FrontUI                      Author:  Victor Feng
Date:    March 9, 2007                            Teacher: Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1
=====*/

/**
 * the GUI design of the main outline of the program. this contains
 * the two JLists at the left, the central panel,
 * and the menu for opening files
 */

import java.awt.*;
import javax.swing.event.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;

public class FrontUI extends JFrame {
    protected JList volunteerList, projectList;
    protected JPanel innerPanelLeft, innerPanelRight,
innerPanelLeftCenter,
        innerPanelLeftNorth, innerPanelLeftSouth;
    protected File volunteerFile, projectFile;
    protected String[] vNames, pNames;
    private RandomAccessFile rVolunteerFile, rProjectFile;

    public FrontUI() {
        super("Volunteerist"); // set title

        // Construct menu
        JMenuBar menuBar = new JMenuBar();
        setJMenuBar(menuBar);
        JMenu fileMenu = new JMenu("File");
        JMenuItem openVolunteer = new JMenuItem("Open volunteer
database");
        openVolunteer.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                openVolunteerDatabase();
            }
        });
    }
}
```

```

JMenuItem openProject = new JMenuItem("Open Project
database");
openProject.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent event) {
        openProjectDatabase();
    }
});
fileMenu.add(openVolunteer);
fileMenu.add(openProject);
menuBar.add(fileMenu);

// Jlist Test
String defaultList[] = { "no item" };

// set the layout of the frame
setLayout(new BorderLayout());

// set up the JList for volunteers
volunteerList = new JList(defaultList); //
*****
volunteerList.setVisibleRowCount(30);
volunteerList.setFixedCellWidth(120);

volunteerList.setSelectionMode(ListSelectionModel.SINGLE_SELECTIO
N);
volunteerList.addListSelectionListener( //
*****
    new ListSelectionListener() {
        public void
valueChanged(ListSelectionEvent event) {

        }
    });

// set up the JList for projects
projectList = new JList(defaultList); //
*****
projectList.setVisibleRowCount(30);
projectList.setFixedCellWidth(120);

projectList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION)
;
projectList.addListSelectionListener( //
*****
    new ListSelectionListener() {
        public void
valueChanged(ListSelectionEvent event) {

        }
    });

// set up the Jbuttons for creating a new volunteer file
and for
// creating a new project file
// the two buttons will then being organized into the south
panel of the
// innerPanelLeft

```

```

        JButton newVolunteerFileButton = new JButton("New
Volunteer");
        JButton newProjectFile = new JButton("New Project");
        innerPanelLeftSouth = new JPanel();
        newVolunteerFileButton.addActionListener(new
ActionListener() {
            public void actionPerformed(ActionEvent event) {
                innerPanelRight.setVisible(false);
                createNewVolunteerProfile();
            }
        });
        newProjectFile.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                innerPanelRight.setVisible(false);
                createNewProject();
            }
        });
        innerPanelLeftSouth.setLayout(new GridLayout(1, 2));
        innerPanelLeftSouth.add(newVolunteerFileButton);
        innerPanelLeftSouth.add(newProjectFile);

        // create the left inner panel and add two lists and two
buttons into
        // there
        innerPanelLeftCenter = new JPanel();
        innerPanelLeftCenter.setLayout(new GridLayout(1, 2));
        innerPanelLeftCenter.add(new JScrollPane(volunteerList));
        innerPanelLeftCenter.add(new JScrollPane(projectList));

        // set up labels for the two JLists
        innerPanelLeftNorth = new JPanel();
        innerPanelLeftNorth.setLayout(new GridLayout(1, 2));
        innerPanelLeftNorth.add(new JLabel("Volunteer:"));
        innerPanelLeftNorth.add(new JLabel("Project:"));

        innerPanelLeft = new JPanel();
        innerPanelLeft.setLayout(new BorderLayout());
        innerPanelLeft.add(innerPanelLeftNorth,
BorderLayout.NORTH);
        innerPanelLeft.add(innerPanelLeftCenter,
BorderLayout.CENTER);
        innerPanelLeft.add(innerPanelLeftSouth,
BorderLayout.SOUTH);

        // create the right inner panel and
*****
        innerPanelRight = new JPanel();
        innerPanelRight.setLayout(new BorderLayout());

        // add the inner panels into the window
        add(innerPanelLeft, BorderLayout.WEST);
        add(innerPanelRight, BorderLayout.CENTER);

        // close file when exiting the program
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                if (rVolunteerFile != null)

```

```

        closeFile(rVolunteerFile);
        if (rProjectFile != null)
            closeFile(rProjectFile);
    }
});

// set the size of the window
setSize(1000, 800);
setVisible(true);

}

/**
 * creates a new volunteer profile through class
VolunteerInputPanel
 */
private void createNewVolunteerProfile() {
    final VolunteerInputPanel viPanel = new
VolunteerInputPanel();
    innerPanelRight = viPanel;
    innerPanelRight.setVisible(true);
    add(innerPanelRight, BorderLayout.CENTER);
    JButton clearButton = viPanel.clearButton;
    clearButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            viPanel.clearFields();
        }
    });

    JButton saveButton = viPanel.saveButton;
    saveButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            saveVolunteerFile(viPanel);
        }
    });

    setVisible(true);
}

/**
 * creates a new project through class VolunteerInputPanel
 */
private void createNewProject() {
    final ProjectInputPanel proPanel = new ProjectInputPanel();
    innerPanelRight = proPanel;
    innerPanelRight.setVisible(true);
    add(innerPanelRight, BorderLayout.CENTER);
    JButton clearButton = proPanel.clearButton;
    clearButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            proPanel.clearFields();
        }
    });

    // set up save button
    JButton saveButton = proPanel.saveButton;
    saveButton.addActionListener(new ActionListener() {

```

```

        public void actionPerformed(ActionEvent event) {
            saveProjectFile(proPanel);
        }
    });

    setVisible(true);
}

/**
 * save the project file into database
 */
public void saveProjectFile(ProjectInputPanel projectPanel) {

    // test if the format is acceptable
    String errorMessage = projectPanel.isFormatAcceptable();
    if (!errorMessage.equalsIgnoreCase("PASS")) {
        JOptionPane.showMessageDialog(this, errorMessage,
"ERROR",
                                JOptionPane.ERROR_MESSAGE);
        return;
    }

    // save the data into the file
    RandomAccessVolunteer_Project project = new
RandomAccessVolunteer_Project();
    String[] fieldValues;
    fieldValues = projectPanel.getFieldValues();
    project.setName(fieldValues[0]);
    project.setDate(fieldValues[1]);
    project.setAddress(fieldValues[2]);
    project.setPhoneNumber(fieldValues[3]);
    project.setSupervisor(fieldValues[4]);
    project.setEmail(fieldValues[5]);
    project.setHours(Double.parseDouble(fieldValues[6]));
    project.setDescription(fieldValues[7]);

    // set up the skill set
    boolean skills[] = new boolean[Volunteer.SKILL_SET.length];
    for (int i = 0; i < projectPanel.skills.length; i++) {
        if (projectPanel.skills[i].isSelected())
            skills[i] = true;
        else
            skills[i] = false;
    }

    project.setSkillSet(skills);

    // save the file into database and catch IOException
    try {
        int fileSize = (int) rProjectFile.length() /
project.size();
        rProjectFile.seek(rProjectFile.length());
        project.write(rProjectFile);
    } catch (IOException ioException) {
        closeFile(rProjectFile);
    }
}

```

```

        projectPanel.clearFields();
    }

    /**
     * save the volunteer file into database from VolunteerInputPanel
     *
     * @param volunteerPanel
     *         the panel where information will be collected
     */
    public void saveVolunteerFile(VolunteerInputPanel volunteerPanel)
    {

        // test if the format is acceptable
        String errorMessage = volunteerPanel.isFormatAcceptable();
        if (!errorMessage.equalsIgnoreCase("PASS")) {
            JOptionPane.showMessageDialog(this, errorMessage,
"ERROR",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        // save the data into the file
        RandomAccessVolunteer volunteer = new
RandomAccessVolunteer();
        String[] fieldValues;
        fieldValues = volunteerPanel.getFieldValues();
        volunteer.setLastName(fieldValues[0]);
        volunteer.setFirstName(fieldValues[1]);
        volunteer.setAge(Integer.parseInt(fieldValues[2]));
        volunteer.setBirthday(fieldValues[3]);
        volunteer.setAddress(fieldValues[4]);
        volunteer.setSex(fieldValues[5].charAt(0));
        volunteer.setEmail(fieldValues[6]);
        volunteer.setPhoneNumber(fieldValues[7]);

        // set up the skill set
        boolean skills[] = new boolean[Volunteer.SKILL_SET.length];
        for (int i = 0; i < volunteerPanel.skills.length; i++) {
            if (volunteerPanel.skills[i].isSelected())
                skills[i] = true;
            else
                skills[i] = false;
        }

        volunteer.setSkillSet(skills);

        // save the file into database and catch IOException
        try {
            int fileSize = (int) rVolunteerFile.length() /
volunteer.size();
            rVolunteerFile.seek(rVolunteerFile.length());
            volunteer.write(rVolunteerFile);
        } catch (IOException ioException) {
            closeFile(rVolunteerFile);
        }

        volunteerPanel.clearFields();
    }
}

```



```

        return;
    }

    // set up the volunteer List

    try {
        rVolunteerFile = new RandomAccessFile(volunteerFile,
"rw");

        } catch (IOException ioException) {
            JOptionPane.showMessageDialog(this, "ERROR OPENING
FILE", "ERROR",
                JOptionPane.ERROR_MESSAGE);
        }

        volunteerListSetUp();
    }

    /**
     * set up the volunteer JList after file has been opened/created
also read
     * volunteer record and display the names of each volunteer on
JList
     *
     * @precondition the file where input will be drawn from must
have been
     *                 selected
     */
    private void volunteerListSetUp() {
        NameList volunteerNames = new NameList();

        RandomAccessVolunteer aVolunteer = new
RandomAccessVolunteer();
        try {
            for (int i = 0; i < rVolunteerFile.length() /
aVolunteer.size(); i++) {
                rVolunteerFile.seek(i * aVolunteer.size());
                aVolunteer.read(rVolunteerFile);

                volunteerNames.addLast(aVolunteer.getLastName()
+ ", "
                    + aVolunteer.getFirstName());
            }

            vNames = volunteerNames.getArray();

            volunteerList = new JList(vNames);
            volunteerList.addListSelectionListener(new
ListSelectionListener() {
                public void valueChanged(ListSelectionEvent
event) {

                    displayVolunteer(vNames[volunteerList.getSelectedIndex()]);
                }
            });
            volunteerList.setVisibleRowCount(30);

```

```

        volunteerList.setFixedCellWidth(120);
        innerPanelLeftCenter.setVisible(false);
        innerPanelLeftCenter = new JPanel();
        innerPanelLeftCenter.setLayout(new GridLayout(1, 2));
        innerPanelLeftCenter.add(new
JScrollPane(volunteerList));
        innerPanelLeftCenter.add(new
JScrollPane(projectList));
        innerPanelLeftCenter.setVisible(true);
        innerPanelLeft.add(innerPanelLeftCenter,
BorderLayout.CENTER);
        add(innerPanelLeft, BorderLayout.WEST);
        add(innerPanelRight, BorderLayout.CENTER);

    }

    catch (IOException ioException) {
        JOptionPane.showMessageDialog(this, "Error reading
from file",
                                     "Error", JOptionPane.ERROR_MESSAGE);
    }
}

/**
 * display a volunteer's profile with its name selected from
JList. This
 * method is directly called only in volunteerListSetUp()
 *
 * @param aName
 *         the name of the volunteer stored in JList
 */

private void displayVolunteer(String aName) {
    int temp = 1, counter = 0, temp2;
    String last_name, first_name;
    boolean done = false;
    while (!done) {
        if (aName.charAt(counter) == ' ' ||
aName.charAt(counter) == ',') {
            temp = counter;
            done = true;
        }
        counter++;
    }
    counter = temp;
    done = false;
    last_name = aName.substring(0, temp);
    while (!done) {
        if (aName.charAt(counter) != ' ' &&
aName.charAt(counter) != ',') {
            temp = counter;
            done = true;
        }
        counter++;
    }
    while (!done) {

```

```

        if (aName.charAt(counter) == ' ' || counter ==
aName.length()) {
            temp2 = counter;
            done = true;
        }
        counter++;
    }
    first_name = aName.substring(temp, counter);

    RandomAccessVolunteer aVolunteer = new
RandomAccessVolunteer();
    try {
        for (int i = 0; i < rVolunteerFile.length() /
aVolunteer.size(); i++) {
            rVolunteerFile.seek(i * aVolunteer.size());
            aVolunteer.read(rVolunteerFile);

            if
(last_name.compareToIgnoreCase(aVolunteer.getLastName()) == 0) {
                innerPanelRight.setVisible(false);
                final VolunteerDisplayPanel vdPanel = new
VolunteerDisplayPanel(
                    aVolunteer);
                innerPanelRight = vdPanel;
                innerPanelRight.setVisible(true);
                add(innerPanelRight,
BorderLayout.CENTER);

                JButton clearButton =
vdPanel.clearButton;
                clearButton.addActionListener(new
ActionListener() {
                    public void
actionPerformed(ActionEvent event) {
                        vdPanel.clearFields();
                    }
                });

                JButton editButton = vdPanel.editButton;
                editButton.addActionListener(new
ActionListener() {
                    public void
actionPerformed(ActionEvent event) {
                        editVolunteerFile(vdPanel);
                    }
                });

                setVisible(true);
            }
        }
    } catch (IOException ioException) {
        JOptionPane.showMessageDialog(this, "Error reading
from file",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

```

/**
 * open a database for project
 *
 */
public void openProjectDatabase() {
    // file opened/created by using JFile Chooser
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);

    int result = fileChooser.showSaveDialog(this);

    // if cancel button is clicked:
    if (result == JFileChooser.CANCEL_OPTION) {
        return;
    }

    projectFile = fileChooser.getSelectedFile();

    // display error message if invalid
    if (projectFile == null ||
projectFile.getName().equals("")) {
        JOptionPane.showMessageDialog(this, "Invalid File
Name",
                                "Invalid File Name",
JOptionPane.ERROR_MESSAGE);
        return;
    }
}

public static void main(String args[]) {
    FrontUI guiTest = new FrontUI();

    guiTest.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}

/*=====
Program:  Volunteerist.NameList                Author:  Victor Feng
Date:    March 9, 2007                        Teacher: Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1
=====*/

/**
 * This class constructs a list that stores, sorts, and processes names
 of an array of names. Its purpose is to construct the two JList
 (Volunteer and Project) in the main Program
 *
 */

import java.util.*;

public class NameList {

```

```

/**
 * Constructs an empty list.
 */
public NameList() {
    first = null;
    currentSize = 0;
}

/**
 * Returns the first element in the linked list.
 *
 * @return the first element in the linked list
 */
public Object getFirst() {
    if (first == null)
        throw new NoSuchElementException();
    return first.data;
}

/**
 * Removes the first element in the linked list.
 *
 * @return the removed element
 */
public Object removeFirst() {
    if (first == null)
        throw new NoSuchElementException();
    Object element = first.data;
    first = first.next;
    currentSize--;
    return element;
}

/**
 * Adds an element to the front of the linked list.
 *
 * @param element
 *         the element to add
 */
public void addFirst(String element) {
    Node newNode = new Node();
    newNode.data = element;
    newNode.next = first;
    first = newNode;
    currentSize++;
}

/**
 * add an element at the end of the list
 *
 * @param element
 *         the element that will be added at the end of the
list
 */
public void addLast(String element) {
    Node next = first;
    if (next == null) {

```

```

        Node newNode = new Node();
        newNode.data = element;
        newNode.next = null;
        first = newNode;
        currentSize++;
        return;
    }
    while (next.next != null) {
        next = next.next;
    }
    Node newNode = new Node();
    newNode.data = element;
    newNode.next = null;
    next.next = newNode;
    currentSize++;
}

/**
 * remove an element at the end of the list
 */
public void removeLast() {
    if (first == null)
        throw new NoSuchElementException();
    Node next = first;
    Node temp = first;
    while (next.next != null) {
        temp = next;
        next = next.next;
    }
    currentSize--;
    temp.next = null;
}

/**
 * get the length of the NameList
 *
 * @return the length of the NameList
 */
public int getLength() {
    return currentSize;
}

/**
 * clear the entire name list
 *
 */
public void clear() {
    first = null;
    currentSize = 0;
}

/**
 * sort the name list using recursive merge sort
 */
public void sort() {
    if (currentSize <= 1)
        return;
}

```

```

        NameList firstList = new NameList();
        NameList secondList = new NameList();
        StringListIterator iterator = listIterator();

        int temp = currentSize / 2;
        for (int i = 1; i <= temp; i++) {
            firstList.addLast(iterator.next());
        }
        while (iterator.hasNext()) {
            secondList.addLast(iterator.next());
        }
        secondList.addLast(iterator.getValue());

        firstList.sort();
        secondList.sort();

        merge(firstList, secondList);
    }

    /**
     * subclass method of sort(). It merges two sorted lists into a
     big list and
     * store it in the original list
     *
     * @param firstL
     *         the first sorted list
     * @param secondL
     *         the second sorted list
     */
    private void merge(NameList firstL, NameList secondL) {

        NameList temp = new NameList();
        StringListIterator firstIterator = firstL.listIterator();
        StringListIterator secondIterator = secondL.listIterator();

        while (firstIterator.isPointingNull()
            && secondIterator.isPointingNull()) {
            if
(firstIterator.getValue().compareTo(secondIterator.getValue()) < 0) {
                temp.addLast(firstIterator.getValue());
                firstIterator.next();
            } else {
                temp.addLast(secondIterator.getValue());
                secondIterator.next();
            }
        }

        while (firstIterator.isPointingNull()) {
            temp.addLast(firstIterator.next());
        }

        while (secondIterator.isPointingNull()) {
            temp.addLast(secondIterator.next());
        }
        // transfer contents in temp into first

```

```

        clear();
        StringListIterator tempIterator = temp.listIterator();
        while (tempIterator.isPointingNull()) {
            addLast(tempIterator.getValue());
            tempIterator.next();
        }
    }

    /**
     * Return an array of strings containing the content in the list
     *
     * @return an array of strings containing the content in the list
     */
    public String[] getArray() {
        sort();
        String[] aString = new String[currentSize];
        StringListIterator iterator = listIterator();
        int counter = 0; // counter for the while loop;
        while (iterator.isPointingNull()) {
            aString[counter] = iterator.getValue();
            iterator.next();
            counter++;
        }

        return aString;
    }

    /**
     * Returns an iterator for iterating through this list.
     *
     * @return an iterator for iterating through this list
     */
    public StringListIterator listIterator() {
        return new NameListIterator();
    }

    private Node first;
    private int currentSize;

    private class Node {
        public String data;
        public Node next;
    }

    private class NameListIterator implements StringListIterator {
        /**
         * Constructs an iterator that points to the front of the
         linked list.
         */
        public NameListIterator() {
            position = first;
            previous = null;
        }
    }
}

```

```

        * return the String value stored in the memory where the
iterator is
        * pointing
        *
        * @return the String value stored in the memory where the
iterator is
        *         pointing
        */
public String getValue() {
    if (position == null)
        return "";
    return position.data;
}

/**
 * test whether the pointer is pointing at null
 *
 * @return boolean true for not pointing at null and false
for pointing
        *         at null
        */
public boolean isPointingNull() {
    return position != null;
}

/**
 * Moves the iterator past the next element.
 *
 * @return the traversed element
 */
public String next() {
    previous = position; // Remember for remove

    if (position == null)
        throw new NoSuchElementException();
    else
        position = position.next;

    return previous.data;
}

/**
 * Tests if there is an element after the iterator
position.
 *
 * @return true if there is an element after the iterator
position
        */
public boolean hasNext() {
    if (position == null)
        return previous != null;
    else
        return position.next != null;
}

/**

```

```

the iterator
    * Adds an element before the iterator position and moves
    * past the inserted element.
    *
    * @param element
    *       the element to add
    */
public void add(String element) {
    if (position == null && previous == null) {
        addFirst(element);
        position = first;
    } else {
        Node newNode = new Node();
        newNode.data = element;
        newNode.next = position.next;
        position.next = newNode;
        position = newNode;
    }
    previous = position;
    currentSize++;
}

/**
 * Removes the last traversed element. This method may only
 * be called
 * after a call to the next() method.
 */
public void remove() {
    if (previous == position)
        throw new IllegalStateException();

    if (position == first) {
        removeFirst();
    } else {
        previous.next = position.next;
    }
    position = previous;
    currentSize--;
}

/**
 * Sets the last traversed element to a different value.
 *
 * @param element
 *       the element to set
 */
public void set(String element) {
    if (position == null)
        throw new NoSuchElementException();
    position.data = element;
}

private Node position;
private Node previous;
}
}

```

```

/*=====
Program:  Volunteerist.ProjectInputPanel          Author:
Victor Feng
Date:    March 9, 2007                          Teacher:  Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1
=====*/

/**
 * This constructs a panel that collects information about a project
 *
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class ProjectInputPanel extends JPanel {
    protected JTextField fields[]; // the text fields
    protected final static String names[] = { "Name:", "Date
(YY/MM/DD):",
        "Address", "Phone number:", "supervisor", "Email
Address:",
        "Hours:", "description:" };
    protected JLabel labels[]; // labels of the input text field
    protected JPanel checkBoxPanel, bigCenterPanel, innerPanelCenter,
        innerPanelSouth, innerPanelSouthEast, innerPanelWest,
fieldPanel[],
        subFieldPanel[];
    protected JButton saveButton, clearButton;
    protected JCheckBox[] skills;

    /**
     * set up the GUI for VolunteerInputPanel
     */
    public ProjectInputPanel() {

        // set up a variable for the length of all the text
criteria of input
        int numberOfInput = names.length;

        // set up the overall layout
        setLayout(new BorderLayout());

        // set up labels and text fields
        labels = new JLabel[numberOfInput];
        fields = new JTextField[numberOfInput];
        for (int i = 0; i < labels.length; i++)
            labels[i] = new JLabel(names[i]);

```

```

// set up field panel and its sub panel
fieldPanel = new JPanel[fields.length];
for (int i = 0; i < fieldPanel.length; i++) {
    fieldPanel[i] = new JPanel();
    fieldPanel[i].setLayout(new BorderLayout());
}

subFieldPanel = new JPanel[fieldPanel.length];
for (int i = 0; i < subFieldPanel.length; i++) {
    subFieldPanel[i] = new JPanel();
    subFieldPanel[i].setLayout(new BorderLayout());
}

// set up and specify the length of each input field
fields[0] = new JTextField(30);
fields[1] = new JTextField(8);
fields[2] = new JTextField(30);
fields[3] = new JTextField(10);
fields[4] = new JTextField(25);
fields[5] = new JTextField(30);
fields[6] = new JTextField(3);
fields[7] = new JTextField(50);

// set up the west inner panel and specify the layout
innerPanelWest = new JPanel();
innerPanelWest.setLayout(new GridLayout(labels.length + 1,
1));

// attach labels into the panel
innerPanelWest.add(new JLabel("New Volunteer Profile"));
for (int i = 0; i < numberOfInput; i++) {
    innerPanelWest.add(labels[i]);
}

// set up the center inner panel and specify the layout
innerPanelCenter = new JPanel();
innerPanelCenter.setLayout(new GridLayout(numberOfInput +
1, 1));

// attach labels and fields into the panel
for (int i = 0; i < numberOfInput; i++) {
    subFieldPanel[i].add(fields[i], BorderLayout.WEST);
    fieldPanel[i].add(subFieldPanel[i],
BorderLayout.SOUTH);
}
innerPanelCenter.add(new JLabel(""));
for (int i = 0; i < numberOfInput; i++) {
    innerPanelCenter.add(fieldPanel[i]);
}

// set up the check box panel
checkBoxPanel = new JPanel();
skills = new JCheckBox[Volunteer.SKILL_SET.length];
for (int i = 0; i < skills.length; i++) {
    skills[i] = new JCheckBox(Volunteer.SKILL_SET[i]);
    checkBoxPanel.add(skills[i]);
}

```

```

        checkBoxPanel.setBorder(new TitledBorder(new
EtchedBorder(), "Skills"));

        // set up the south inner panel and the east panel inside
this panel.
        // Then specify the layout
        innerPanelSouth = new JPanel();
        innerPanelSouth.setLayout(new BorderLayout());
        innerPanelSouthEast = new JPanel();
        innerPanelSouthEast.setLayout(new GridLayout(1, 2));

        // set up the ultimate center panel
        bigCenterPanel = new JPanel();
        bigCenterPanel.setLayout(new BorderLayout());
        bigCenterPanel.add(innerPanelWest, BorderLayout.WEST);
        bigCenterPanel.add(innerPanelCenter, BorderLayout.CENTER);
        bigCenterPanel.add(checkBoxPanel, BorderLayout.SOUTH);

        // set up the "Save" and "Clear" buttons
        saveButton = new JButton("Save");
        clearButton = new JButton("Clear");

        // attach the buttons into the south panel
        innerPanelSouthEast.add(saveButton);
        innerPanelSouthEast.add(clearButton);

        // attach this panel and skill panel into the south panel
        innerPanelSouth.add(innerPanelSouthEast,
BorderLayout.EAST);

        // attach all sub-panels into the VolunteerInputPanel
        add(bigCenterPanel, BorderLayout.CENTER);
        add(innerPanelSouth, BorderLayout.SOUTH);

        // Validate layout
        validate();
    }

    /**
     * get the saveButton
     *
     * @return saveButton
     */
    public JButton getSaveButton() {
        return saveButton;
    }

    /**
     * get the clearButton
     *
     * @return clearButton
     */
    public JButton getClearButton() {
        return clearButton;
    }

    /**

```

```

    * return the fields array
    *
    * @return fields[]
    */
    public JTextField[] getFields() {
        return fields;
    }

    /**
     * clear the content of text fields
     */
    public void clearFields() {
        for (int i = 0; i < fields.length; i++) {
            fields[i].setText("");
        }
        for (int i = 0; i < skills.length; i++) {
            skills[i].setSelected(false);
        }
    }

    /**
     * get array of Strings with current text field contents
     *
     * @return array of strings with current text field contents
     */
    public String[] getFieldValues() {
        String values[] = new String[fields.length];

        for (int i = 0; i < values.length; i++) {
            values[i] = fields[i].getText();
        }

        return values;
    }

    /**
     * check if the values in text field are in the acceptable format
     *
     * @return if the text in the all fields are acceptable, return
    "PASS";
     * otherwise the method return the error message
     */
    public String isFormatAcceptable() {

        // test for the hours input
        try {
            double tester =
Double.parseDouble(fields[6].getText());
            if (tester < 0)
                return "NO NEGATIVE HOURS ACCEPTED";
        } catch (NumberFormatException e) {
            return "ERROR IN THE NUMBER FORMAT FOR HOURS";
        }

        return "PASS";
    }
}

```

```

/*=====
Program:  Volunteerist.RandomAccessVolunteer_Project Author:  Victor
Feng
Date:    March 9, 2007          Teacher:  Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1
=====*/

import java.io.IOException;
import java.io.RandomAccessFile;

/**
 * subclass of Volunteer_Project for random access file handling
 *
 */
public class RandomAccessVolunteer_Project extends Volunteer_Project {
    /**
     * set up no argument-constructor
     */
    public RandomAccessVolunteer_Project() {
        super();
    }

    /**
     * read a record from specified RandomAccessFile
     */
    public void read(RandomAccessFile file) throws IOException {
        setName(padName(file));
        setDate(padDate(file));
        setAddress(padAddress(file));
        setPhoneNumber(padPhoneNumber(file));
        setSupervisor(padSupervisor(file));
        setEmail(padEmailAddress(file));
        setHours(file.readDouble());
        setDescription(padDescription(file));
        setSkillSet(padSkillSet(file));
    }

    /**
     * set birthday into correct length
     */
    private String padDate(RandomAccessFile file) throws IOException
    {
        char name[] = new char[8], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }
    }
}

```

```

        return new String(name).replace('\0', ' ');
    }

    /**
     * set name into proper length
     */
    private String padName(RandomAccessFile file) throws IOException
    {
        char name[] = new char[30], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set address into correct length
     */
    private String padAddress(RandomAccessFile file) throws
IOException {
        char name[] = new char[50], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set phone number into correct length
     */
    private String padPhoneNumber(RandomAccessFile file) throws
IOException {
        char name[] = new char[12], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set supervisor into correct length
     */
    private String padSupervisor(RandomAccessFile file) throws
IOException {

```

```

        char name[] = new char[20], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set email address into correct length
     */
    private String padEmailAddress(RandomAccessFile file) throws
IOException {
        char name[] = new char[30], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set description of the project into correct length
     */
    private String padDescription(RandomAccessFile file) throws
IOException {
        char name[] = new char[50], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set the proper format for skill set
     */
    private boolean[] padSkillSet(RandomAccessFile file) throws
IOException {
        boolean skilling[] = new
boolean[Volunteer.SKILL_SET.length], temp;

        for (int count = 0; count < skilling.length; count++) {
            temp = file.readBoolean();
            skilling[count] = temp;
        }

        return skilling;
    }

```

```

}

/**
 * write a record to specified RandomAccessFile
 */
public void write(RandomAccessFile file) throws IOException {
    writeName(file, getName());
    writeDate(file, getDate());
    writeAddress(file, getAddress());
    writePhoneNumber(file, getPhoneNumber());
    writeSupervisor(file, getSupervisor());
    writeEmail(file, getEmail());
    file.writeDouble(getHours());
    writeDescription(file, getDescription());
    writeSkillSet(file, getSkillSet());
}

/**
 * Write a name to file; maximum of 30 characters
 */
private void writeName(RandomAccessFile file, String name)
    throws IOException {
    StringBuffer buffer = null;

    if (name != null)
        buffer = new StringBuffer(name);
    else
        buffer = new StringBuffer(30);

    buffer.setLength(30);
    file.writeChars(buffer.toString());
}

/**
 * Write a date to file; maximum of 8 characters
 */
private void writeDate(RandomAccessFile file, String Adate)
    throws IOException {
    StringBuffer buffer = null;

    if (Adate != null)
        buffer = new StringBuffer(Adate);
    else
        buffer = new StringBuffer(8);

    buffer.setLength(8);
    file.writeChars(buffer.toString());
}

/**
 * Write an address to file; maximum of 50 characters
 */
private void writeAddress(RandomAccessFile file, String
anAddress)

```

```

        throws IOException {
StringBuffer buffer = null;

    if (anAddress != null)
        buffer = new StringBuffer(anAddress);
    else
        buffer = new StringBuffer(50);

    buffer.setLength(50);
    file.writeChars(buffer.toString());
}

/**
 * Write a phone number to file; maximum of 12 characters
 */
private void writePhoneNumber(RandomAccessFile file, String
aNumber)
    throws IOException {
StringBuffer buffer = null;

    if (aNumber != null)
        buffer = new StringBuffer(aNumber);
    else
        buffer = new StringBuffer(12);

    buffer.setLength(12);
    file.writeChars(buffer.toString());
}

/**
 * Write a supervisor to file; maximum of 20 characters
 */
private void writeSupervisor(RandomAccessFile file, String
aSupervisor)
    throws IOException {
StringBuffer buffer = null;

    if (aSupervisor != null)
        buffer = new StringBuffer(aSupervisor);
    else
        buffer = new StringBuffer(20);

    buffer.setLength(20);
    file.writeChars(buffer.toString());
}

/**
 * Write an email address to file; maximum of 30 characters
 */
private void writeEmail(RandomAccessFile file, String anEmail)
    throws IOException {
StringBuffer buffer = null;

    if (anEmail != null)
        buffer = new StringBuffer(anEmail);

```

```

        else
            buffer = new StringBuffer(30);

        buffer.setLength(30);
        file.writeChars(buffer.toString());
    }

    /**
     * Write an email address to file; maximum of 50 characters
     */
    private void writeDescription(RandomAccessFile file, String
aDescription)
        throws IOException {
        StringBuffer buffer = null;

        if (aDescription != null)
            buffer = new StringBuffer(aDescription);
        else
            buffer = new StringBuffer(50);

        buffer.setLength(50);
        file.writeChars(buffer.toString());
    }

    /**
     * Write a skill set to file in correct format;
     *
     * @precondition the skill set is in the correct length
     */
    private void writeSkillSet(RandomAccessFile file, boolean[]
aSkillSet)
        throws IOException {
        for (int count = 0; count < aSkillSet.length; count++) {
            file.writeBoolean(aSkillSet[count]);
        }
    }

    /**
     * size of a record of information
     */
    public static int size() {
        return 410;
    }
}

/*=====
Program: Volunteerist.RandomAccessVolunteer   Author: Victor Feng
Date:    March 9, 2007                       Teacher: Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
*/

```

IDE: Eclipse 3.1

=====*/

```
/**
 * subclass of Volunteer for random access file handling
 *
 */
import java.io.*;

public class RandomAccessVolunteer extends Volunteer {
    /**
     * set up no argument-constructor
     */
    public RandomAccessVolunteer() {
        super();
    }

    /**
     * read a record from specified RandomAccessFile
     */
    public void read(RandomAccessFile file) throws IOException {
        setLastName(padName(file));
        setFirstName(padName(file));
        setAge(file.readInt());
        setBirthday(padBirthday(file));
        setAddress(padAddress(file));
        setSex(file.readChar());
        setEmail(padEmailAddress(file));
        setPhoneNumber(padPhoneNumber(file));
        setHours(file.readDouble());
        setSkillSet(padSkillSet(file));
    }

    /**
     * set name into proper length
     */
    private String padName(RandomAccessFile file) throws IOException
    {
        char name[] = new char[15], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set birthday into correct length
     */
    private String padBirthday(RandomAccessFile file) throws
IOException {
        char name[] = new char[8], temp;
```

```

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set address into correct length
     */
    private String padAddress(RandomAccessFile file) throws
IOException {
        char name[] = new char[50], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set email address into correct length
     */
    private String padEmailAddress(RandomAccessFile file) throws
IOException {
        char name[] = new char[30], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }

    /**
     * set phone number into correct length
     */
    private String padPhoneNumber(RandomAccessFile file) throws
IOException {
        char name[] = new char[12], temp;

        for (int count = 0; count < name.length; count++) {
            temp = file.readChar();
            name[count] = temp;
        }

        return new String(name).replace('\0', ' ');
    }
}

```

```

    /**
     * set the proper format for skill set
     */
    private boolean[] padSkillSet(RandomAccessFile file) throws
IOException {
        boolean skilling[] = new
boolean[Volunteer.SKILL_SET.length], temp;

        for (int count = 0; count < skilling.length; count++) {
            temp = file.readBoolean();
            skilling[count] = temp;
        }

        return skilling;
    }

    /**
     * write a record to specified RandomAccessFile
     */
    public void write(RandomAccessFile file) throws IOException {
        writeName(file, getLastName());
        writeName(file, getFirstName());
        file.writeInt(getAge());
        writeBirthday(file, getBirthday());
        writeAddress(file, getAddress());
        file.writeChar(getSex());
        writeEmail(file, getEmail());
        writePhoneNumber(file, getPhoneNumber());
        file.writeDouble(getHours());
        writeSkillSet(file, getSkillSet());
    }

    /**
     * Write a name to file; maximum of 15 characters
     */
    private void writeName(RandomAccessFile file, String name)
        throws IOException {
        StringBuffer buffer = null;

        if (name != null)
            buffer = new StringBuffer(name);
        else
            buffer = new StringBuffer(15);

        buffer.setLength(15);
        file.writeChars(buffer.toString());
    }

    /**
     * Write a birthday to file; maximum of 8 characters
     */
    private void writeBirthday(RandomAccessFile file, String
aBirthday)
        throws IOException {
        StringBuffer buffer = null;

```

```

        if (aBirthday != null)
            buffer = new StringBuffer(aBirthday);
        else
            buffer = new StringBuffer(8);

        buffer.setLength(8);
        file.writeChars(buffer.toString());
    }

    /**
     * Write an address to file; maximum of 50 characters
     */
    private void writeAddress(RandomAccessFile file, String
anAddress)
        throws IOException {
        StringBuffer buffer = null;

        if (anAddress != null)
            buffer = new StringBuffer(anAddress);
        else
            buffer = new StringBuffer(50);

        buffer.setLength(50);
        file.writeChars(buffer.toString());
    }

    /**
     * Write an email address to file; maximum of 30 characters
     */
    private void writeEmail(RandomAccessFile file, String anEmail)
        throws IOException {
        StringBuffer buffer = null;

        if (anEmail != null)
            buffer = new StringBuffer(anEmail);
        else
            buffer = new StringBuffer(30);

        buffer.setLength(30);
        file.writeChars(buffer.toString());
    }

    /**
     * Write a phone number to file; maximum of 12 characters
     */
    private void writePhoneNumber(RandomAccessFile file, String
aNumber)
        throws IOException {
        StringBuffer buffer = null;

        if (aNumber != null)
            buffer = new StringBuffer(aNumber);
        else

```

```

        buffer = new StringBuffer(12);

        buffer.setLength(12);
        file.writeChars(buffer.toString());

    }

    /**
     * Write a skill set to file in correct format;
     *
     * @precondition the skill set is in the correct length
     */
    private void writeSkillSet(RandomAccessFile file, boolean[]
aSkillSet)
        throws IOException {
        for (int count = 0; count < aSkillSet.length; count++) {
            file.writeBoolean(aSkillSet[count]);
        }
    }

    /**
     * size of a record of information
     */
    public static int size() {
        return 280;
    }
}

/*=====
Program: Volunteerist.StringListIterator      Author: Victor Feng
Date:    March 9, 2007                        Teacher: Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1
=====*/

/**
 * A list iterator allows access of a position in a NameList
 */
public interface StringListIterator {
    /**
     * Moves the iterator past the next element.
     *
     * @return the traversed element
     */
    String next();

    /**
     * Tests if there is an element after the iterator position.
     *
     * @return true if there is an element after the iterator
position

```

```

        */
        boolean hasNext();

        /**
         * return the String value stored in the memory where the
iterator is
         * pointing
         *
         * @return the String value stored in the memory where the
iterator is
         *         pointing
         */
        public String getValue();

        /**
         * Adds an element before the iterator position and moves the
iterator past
         * the inserted element.
         *
         * @param element
         *         the element to add
         */
        void add(String element);

        /**
         * test whether the pointer is pointing at null
         *
         * @return boolean true for not pointing at null and false for
pointing at
         *         null
         */
        public boolean isPointingNull();

        /**
         * Removes the last traversed element. This method may only be
called after
         * a call to the next() method.
         */
        void remove();

        /**
         * Sets the last traversed element to a different value.
         *
         * @param element
         *         the element to set
         */
        void set(String element);
}

/*=====
Program:  Volunteerist.Volunteer_Project      Author:  Victor Feng
Date:    March 9, 2007                       Teacher:  Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
*/

```

System: Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1

=====*/

```
/**
 * a class that contains and processes all necessary information on a
 volunteering project
 *
 */
```

```
import java.io.Serializable;
```

```
public class Volunteer_Project implements Serializable {
    private double hours; // the volunteering hours involved in this
 project
```

```
    private String name;
    private String date; // the date of the project
    private String address;
    private String phoneNumber;
    private String supervisor; // names of the supervisor
    private String email; // the email address which may be used to
 contact
```

```
    // the work station
    private String description; // a short description regarding the
 nature of
```

```
    // this project
    private boolean[] skillSet;
```

```
/**
 * set the name of the station
 *
 * @param aName
 *         name of the station
 */
```

```
public void setName(String aName) {
    name = aName;
    skillSet = new boolean[Volunteer.SKILL_SET.length];
    for (int i = 0; i < skillSet.length; i++)
        skillSet[i] = false;
}
```

```
/**
 * get the name of the station
 *
 */
```

```
public String getName() {
    return name;
}
```

```
/**
 * set the volunteering hours of the project
 *
 * @param anHours
 *         the volunteering hours of the project
 */
```

```
public void setHours(double anHours) {
    hours = anHours;
}
```

```

}

/**
 * get the volunteering hours of the station
 *
 */
public double getHours() {
    return hours;
}

/**
 * set the date of the project
 *
 * @param aDate
 *         the date of the project
 */
public void setDate(String aDate) {
    date = aDate;
}

/**
 * get the date of the station
 *
 */
public String getDate() {
    return date;
}

/**
 * set the address of the project
 *
 * @param anAddress
 *         birthday of the project
 */
public void setAddress(String anAddress) {
    address = anAddress;
}

/**
 * get the address of the project
 *
 */
public String getAddress() {
    return address;
}

/**
 * set the phone number of the project
 *
 * @param aNumber
 *         the phone number of the project
 */
public void setPhoneNumber(String aNumber) {
    phoneNumber = aNumber;
}

/**

```

```

    * get the phone number of the project
    *
    */
    public String getPhoneNumber() {
        return phoneNumber;
    }

    /**
     * set the email address of the project
     *
     * @param anEmail
     *         the email address of the project
     */
    public void setEmail(String anEmail) {
        email = anEmail;
    }

    /**
     * get the email of the project
     *
     */
    public String getEmail() {
        return email;
    }

    /**
     * set the name of supervisors of the project
     *
     * @param aSupervisor
     *         the name of supervisors in the project
     */
    public void setSupervisor(String aSupervisor) {
        supervisor = aSupervisor;
    }

    /**
     * get the names of supervisors in the project
     *
     */
    public String getSupervisor() {
        return supervisor;
    }

    /**
     * set the description of the project
     *
     * @param aDescription
     *         the description of the project
     */
    public void setDescription(String aDescription) {
        description = aDescription;
    }

    /**
     * get the description of the project
     *
     */

```

```

    public String getDescription() {
        return description;
    }

    /**
     * set the skill set that the volunteer possesses
     *
     * @param aSkillSet
     *         the skill set that will be set for the volunteer
     * @precondition the length of the boolean array parameter is
equal to that
     *         of the SKILL_SET
     */
    public void setSkillSet(boolean[] aSkillSet) {
        for (int i = 0; i < Volunteer.SKILL_SET.length; i++) {
            skillSet[i] = aSkillSet[i];
        }
    }

    /**
     * get the skill sets that the volunteer possesses
     *
     * @return skill set in boolean form
     */
    public boolean[] getSkillSet() {
        return skillSet;
    }
}

/*=====
Program:  Volunteerist.Volunteer          Author:  Victor Feng
Date:    March 9, 2007                    Teacher:  Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1
=====*/

/**
 * a class that contains and processes necessary information on
volunteers
 *
 */

import java.io.Serializable;

public class Volunteer implements Serializable {
    private String lastName;
    private String firstName;
    private int age;
    private String birthday;
    private String address;
    private char sex;
}

```

```

    private double hours; // the volunteer hours that the volunteer
completed
    private String email; // email address of the volunteer
    private String phoneNumber; // contact phone number of the
volunteer
    public static final String[] SKILL_SET = { "Driving", "Watching
Children",
        "Working in Alcoholic Environment", "Lifting 20kg or
up",
        "Music Skill", "Art Skill" };
    private boolean[] skillSet;

    /**
     * construct a volunteer profile
     */
    public Volunteer() {
        hours = 0;
        skillSet = new boolean[SKILL_SET.length];
        for (int i = 0; i < skillSet.length; i++)
            skillSet[i] = false;
    }

    /**
     * set the first name of the volunteer
     *
     * @param aFirstName
     *         first name of the volunteer
     */
    public void setFirstName(String aFirstName) {
        firstName = aFirstName;
    }

    /**
     * get the first name of the volunteer
     *
     */
    public String getFirstName() {
        return firstName;
    }

    /**
     * set the last name of the volunteer
     *
     * @param aLastName
     *         last name of the volunteer
     */
    public void setLastName(String aLastName) {
        lastName = aLastName;
    }

    /**
     * get the last name of the volunteer
     *
     */
    public String getLastName() {
        return lastName;
    }
}

```

```

/**
 * set the age of the volunteer
 *
 * @param anAge
 *         age of the volunteer
 */
public void setAge(int anAge) {
    age = anAge;
}

/**
 * get the age of the volunteer
 *
 */
public int getAge() {
    return age;
}

/**
 * set the birthday of the volunteer
 *
 * @param aBirthday
 *         birthday of the volunteer
 */
public void setBirthday(String aBirthday) {
    birthday = aBirthday;
}

/**
 * get the birthday of the volunteer
 *
 */
public String getBirthday() {
    return birthday;
}

/**
 * set the address of the volunteer
 *
 * @param anAddress
 *         birthday of the volunteer
 */
public void setAddress(String anAddress) {
    address = anAddress;
}

/**
 * get the address of the volunteer
 *
 */
public String getAddress() {
    return address;
}

/**
 * set the sex of the volunteer

```

```

*
* @param aSex
*         the sex of the volunteer
*/
public void setSex(char aSex) {
    sex = aSex;
}

/**
 * get the sex of the volunteer
 *
 */
public char getSex() {
    return sex;
}

/**
 * set the email address of the volunteer
 *
 * @param anEmail
 *         the email address of the volunteer
 */
public void setEmail(String anEmail) {
    email = anEmail;
}

/**
 * get the email of the volunteer
 *
 */
public String getEmail() {
    return email;
}

/**
 * set the phone number of the volunteer
 *
 * @param aNumber
 *         the phone number of the volunteer
 */
public void setPhoneNumber(String aNumber) {
    phoneNumber = aNumber;
}

/**
 * get the phone number of the volunteer
 *
 */
public String getPhoneNumber() {
    return phoneNumber;
}

/**
 * set the volunteering hours of the volunteer
 *
 * @param aNumber
 *         the volunteering hours of the volunteer

```

```

    */
    public void setHours(double aNumber) {
        hours = aNumber;
    }

    /**
     * add volunteering hours into his/her profile
     *
     * @param anHours
     *         the volunteering hours that will be added
     */
    public void addHours(double anHours) {
        hours += anHours;
    }

    /**
     * get the volunteering hours of the volunteer
     *
     */
    public double getHours() {
        return hours;
    }

    /**
     * set the skill set that the volunteer possesses
     *
     * @param aSkillSet
     *         the skill set that will be set for the volunteer
     * @precondition the length of the boolean array parameter is
equal to that
     *         of the SKILL_SET
     */
    public void setSkillSet(boolean[] aSkillSet) {
        for (int i = 0; i < skillSet.length; i++) {
            skillSet[i] = aSkillSet[i];
        }
    }

    /**
     * get the skill sets that the volunteer possesses
     *
     * @return skill set in boolean form
     */
    public boolean[] getSkillSet() {
        return skillSet;
    }
}

/*=====
Program: Volunteerist.VolunteerDisplayPanel   Author: Victor Feng
Date:    March 9, 2007                       Teacher: Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
*/

```

Target Operating System: Java Virtual Machine
System: Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1

=====*/

```
/**
 * This constructs a panel that displays information of a volunteer
 *
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;
import java.io.*;

public class VolunteerDisplayPanel extends JPanel {
    protected JTextField fields[]; // the text fields
    protected final static String names[] = { "Last Name:", "First
Name:",
        "Age:", "Birthday (YY/MM/DD):", "Address:", "Sex
(M/F):",
        "Email Address:", "Hours", "Phone Number:" };
    protected JLabel labels[]; // labels of the input text field
    protected JPanel checkBoxPanel, bigCenterPanel, innerPanelCenter,
innerPanelSouth, innerPanelSouthEast, innerPanelWest,
fieldPanel[],
        subFieldPanel[];
    protected JButton editButton, eraseButton, clearButton;
    protected JCheckBox[] skills;

    /**
     * set up the GUI for VolunteerInputPanel
     */
    public VolunteerDisplayPanel(RandomAccessVolunteer volunteer) {

        // set up a variable for the length of all the criteria of
input
        int numberOfInput = names.length;

        // set up the overall layout
        setLayout(new BorderLayout());

        // set up labels and text fields
        labels = new JLabel[numberOfInput];
        fields = new JTextField[numberOfInput];
        for (int i = 0; i < labels.length; i++)
            labels[i] = new JLabel(names[i]);

        // set up field panel and its sub panel
        fieldPanel = new JPanel[fields.length];
        for (int i = 0; i < fieldPanel.length; i++) {
            fieldPanel[i] = new JPanel();
            fieldPanel[i].setLayout(new BorderLayout());
        }
    }
}
```

```

subFieldPanel = new JPanel[fieldPanel.length];
for (int i = 0; i < subFieldPanel.length; i++) {
    subFieldPanel[i] = new JPanel();
    subFieldPanel[i].setLayout(new BorderLayout());
}

// set up and specify the length of each input field
fields[0] = new JTextField(volunteer.getLastName(), 15);
fields[1] = new JTextField(volunteer.getFirstName(), 15);
fields[2] = new JTextField(volunteer.getAge() + "", 2);
fields[3] = new JTextField(volunteer.getBirthDay(), 8);
fields[4] = new JTextField(volunteer.getAddress(), 40);
fields[5] = new JTextField(volunteer.getSex() + "", 1);
fields[6] = new JTextField(volunteer.getEmail(), 40);
fields[7] = new JTextField(volunteer.getHours() + "", 3);
fields[8] = new JTextField(volunteer.getPhoneNumber(), 10);

// set up the west inner panel and specify the layout
innerPanelWest = new JPanel();
innerPanelWest.setLayout(new GridLayout(labels.length + 1,
1));

// attach labels into the panel
innerPanelWest.add(new JLabel("New Volunteer Profile"));
for (int i = 0; i < numberOfInput; i++) {
    innerPanelWest.add(labels[i]);
}

// set up the center inner panel and specify the layout
innerPanelCenter = new JPanel();
innerPanelCenter.setLayout(new GridLayout(numberOfInput +
1, 1));

// attach labels and fields into the panel
for (int i = 0; i < numberOfInput; i++) {
    subFieldPanel[i].add(fields[i], BorderLayout.WEST);
    fieldPanel[i].add(subFieldPanel[i],
BorderLayout.SOUTH);
}
innerPanelCenter.add(new JLabel(""));
for (int i = 0; i < numberOfInput; i++) {
    innerPanelCenter.add(fieldPanel[i]);
}

// set up the check box panel
checkBoxPanel = new JPanel();
skills = new JCheckBox[Volunteer.SKILL_SET.length];
for (int i = 0; i < skills.length; i++) {
    skills[i] = new JCheckBox(Volunteer.SKILL_SET[i]);
    if (volunteer.getSkillSet()[i])
        skills[i].setSelected(true);
    checkBoxPanel.add(skills[i]);
}
checkBoxPanel.setBorder(new TitledBorder(new
EtchedBorder(), "Skills"));

// set up the ultimate center panel

```

```

        bigCenterPanel = new JPanel();
        bigCenterPanel.setLayout(new BorderLayout());
        bigCenterPanel.add(innerPanelWest, BorderLayout.WEST);
        bigCenterPanel.add(innerPanelCenter, BorderLayout.CENTER);
        bigCenterPanel.add(checkBoxPanel, BorderLayout.SOUTH);

        // set up the south inner panel and the east panel inside
this panel.
        // Then specify the layout
        innerPanelSouth = new JPanel();
        innerPanelSouth.setLayout(new BorderLayout());
        innerPanelSouthEast = new JPanel();
        innerPanelSouthEast.setLayout(new GridLayout(1, 2));

        // set up the "Edit", "Save" and "Clear" buttons
        editButton = new JButton("Edit");
        eraseButton = new JButton("Erase");
        clearButton = new JButton("Clear");

        // attach the buttons into the south panel
        innerPanelSouthEast.add(editButton);
        innerPanelSouthEast.add(eraseButton);
        innerPanelSouthEast.add(clearButton);

        // attach this panel into the south panel
        innerPanelSouth.add(innerPanelSouthEast,
BorderLayout.EAST);

        // attach all sub-panels into the VolunteerInputPanel
        add(bigCenterPanel, BorderLayout.CENTER);
        add(innerPanelSouth, BorderLayout.SOUTH);

        // Validate layout
        validate();
    }

    /**
     * get the eraseButton
     *
     * @return eraseButton
     */
    public JButton getEraseButton() {
        return eraseButton;
    }

    /**
     * get the clearButton
     *
     * @return clearButton
     */
    public JButton getClearButton() {
        return clearButton;
    }

    /**
     * return the fields array
     *

```

```

    * @return fields[]
    */
    public JTextField[] getFields() {
        return fields;
    }

    /**
     * clear the content of text fields
     */
    public void clearFields() {
        for (int i = 0; i < fields.length; i++) {
            fields[i].setText("");
        }
        for (int i = 0; i < skills.length; i++) {
            skills[i].setSelected(false);
        }
    }

    /**
     * get array of Strings with current text field contents
     *
     * @return array of strings with current text field contents
     */
    public String[] getFieldValues() {
        String values[] = new String[fields.length];

        for (int i = 0; i < values.length; i++) {
            values[i] = fields[i].getText();
        }

        return values;
    }

    /**
     * check if the values in text field are in the acceptable format
     *
     * @return if the text in the all fields are acceptable, return
    "PASS";
     * otherwise the method return the error message
     */
    public String isFormatAcceptable() {

        // test for the age input
        try {
            int tester = Integer.parseInt(fields[2].getText(),
10);

            if (tester < 0)
                return "NO NEGATIVE AGE ACCEPTED";
        } catch (NumberFormatException e) {
            return "ERROR IN THE NUMBER FORMAT FOR AGE";
        }

        // check for the sex input
        if (fields[5].getText().length() != 1)
            return "PLEASE INSERT ONLY ONE LETTER FOR THE SEX
INPUT";
    }

```

```

        if (!(fields[5].getText().equalsIgnoreCase("M") ||
fields[5].getText()
                .equalsIgnoreCase("F")))
            return "WRONG INPUT FOR VOLUNTEER'S SEX";

        return "PASS";
    }
}

/*=====
Program: Volunteerist.VolunteerDisplayPanel   Author: Victor Feng
Date:    March 9, 2007                       Teacher: Gerry Donaldson
School:  Sir Winston Churchill High School, Calgary, Alberta, Canada
Language: Java J2SE 5.0
Target Operating System: Java Virtual Machine
System:  Pentium IV 2.5 GHz running under Windows XP
IDE: Eclipse 3.1
=====*/

/**
 * This constructs a panel that collects information about a volunteer
 *
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;

public class VolunteerInputPanel extends JPanel {
    protected JTextField fields[]; //the text fields
    protected final static String names[] = { "Last Name:", "First
Name:",
        "Age:", "Birthday (YY/MM/DD):", "Address:", "Sex
(M/F):",
        "Email Address:", "Phone Number:" };
    protected JLabel labels[]; //labels of the input text field
    protected JPanel checkBoxPanel, bigCenterPanel, innerPanelCenter,
innerPanelSouth, innerPanelSouthEast, innerPanelWest,
fieldPanel[],
        subFieldPanel[];
    protected JButton saveButton, clearButton;
    protected JCheckBox[] skills;

    /**
     * set up the GUI for VolunteerInputPanel
     */
    public VolunteerInputPanel() {

        //set up a variable for the length of all the criteria of
input
        int numberOfInput = names.length;

```

```

//set up the overall layout
setLayout(new BorderLayout());

//set up labels and text fields
labels = new JLabel[numberOfInput];
fields = new JTextField[numberOfInput];
for (int i = 0; i < labels.length; i++)
    labels[i] = new JLabel(names[i]);

//set up field panel and its sub panel
fieldPanel = new JPanel[fields.length];
for (int i = 0; i < fieldPanel.length; i++) {
    fieldPanel[i] = new JPanel();
    fieldPanel[i].setLayout(new BorderLayout());
}

subFieldPanel = new JPanel[fieldPanel.length];
for (int i = 0; i < subFieldPanel.length; i++) {
    subFieldPanel[i] = new JPanel();
    subFieldPanel[i].setLayout(new BorderLayout());
}

//set up and specify the length of each input field
fields[0] = new JTextField(15);
fields[1] = new JTextField(15);
fields[2] = new JTextField(2);
fields[3] = new JTextField(8);
fields[4] = new JTextField(40);
fields[5] = new JTextField(1);
fields[6] = new JTextField(40);
fields[7] = new JTextField(10);

//set up the west inner panel and specify the layout
innerPanelWest = new JPanel();
innerPanelWest.setLayout(new GridLayout(labels.length + 1,
1));

//attach labels into the panel
innerPanelWest.add(new JLabel("New Volunteer Profile"));
for (int i = 0; i < numberOfInput; i++) {
    innerPanelWest.add(labels[i]);
}

//set up the center inner panel and specify the layout
innerPanelCenter = new JPanel();
innerPanelCenter.setLayout(new GridLayout(numberOfInput +
1, 1));

//attach labels and fields into the panel
for (int i = 0; i < numberOfInput; i++) {
    subFieldPanel[i].add(fields[i], BorderLayout.WEST);
    fieldPanel[i].add(subFieldPanel[i],
BorderLayout.SOUTH);
}
innerPanelCenter.add(new JLabel(""));
for (int i = 0; i < numberOfInput; i++) {

```

```

        innerPanelCenter.add(fieldPanel[i]);
    }

    //set up the check box panel
    checkBoxPanel = new JPanel();
    skills = new JCheckBox[Volunteer.SKILL_SET.length];
    for (int i = 0; i < skills.length; i++) {
        skills[i] = new JCheckBox(Volunteer.SKILL_SET[i]);
        checkBoxPanel.add(skills[i]);
    }
    checkBoxPanel.setBorder(new TitledBorder(new
EtchedBorder(), "Skills"));

    //set up the ultimate center panel
    bigCenterPanel = new JPanel();
    bigCenterPanel.setLayout(new BorderLayout());
    bigCenterPanel.add(innerPanelWest, BorderLayout.WEST);
    bigCenterPanel.add(innerPanelCenter, BorderLayout.CENTER);
    bigCenterPanel.add(checkBoxPanel, BorderLayout.SOUTH);

    //set up the south inner panel and the east panel inside
this panel. Then specify the layout
    innerPanelSouth = new JPanel();
    innerPanelSouth.setLayout(new BorderLayout());
    innerPanelSouthEast = new JPanel();
    innerPanelSouthEast.setLayout(new GridLayout(1, 2));

    //set up the "Save" and "Clear" buttons
    saveButton = new JButton("Save");
    clearButton = new JButton("Clear");

    //attach the buttons into the south panel
    innerPanelSouthEast.add(saveButton);
    innerPanelSouthEast.add(clearButton);

    //attach this panel into the south panel
    innerPanelSouth.add(innerPanelSouthEast,
BorderLayout.EAST);

    //attach all sub-panels into the VolunteerInputPanel
    add(bigCenterPanel, BorderLayout.CENTER);
    add(innerPanelSouth, BorderLayout.SOUTH);

    //Validate layout
    validate();
}

/**
 * get the saveButton
 * @return saveButton
 */
public JButton getSaveButton() {
    return saveButton;
}

/**
 * get the clearButton

```

```

    * @return clearButton
    */
    public JButton getClearButton() {
        return clearButton;
    }

    /**
     * return the fields array
     * @return fields[]
     */
    public JTextField[] getFields() {
        return fields;
    }

    /**
     * clear the content of text fields
     */
    public void clearFields() {
        for (int i = 0; i < fields.length; i++) {
            fields[i].setText("");
        }
        for (int i = 0; i < skills.length; i++) {
            skills[i].setSelected(false);
        }
    }

    /**
     * get array of Strings with current text field contents
     * @return array of strings with current text field contents
     */
    public String[] getFieldValues() {
        String values[] = new String[fields.length];

        for (int i = 0; i < values.length; i++) {
            values[i] = fields[i].getText();
        }

        return values;
    }

    /**
     * check if the values in text field are in the acceptable format
     * @return if the text in the all fields are acceptable, return
    "PASS"; otherwise the method return the error message
     */
    public String isFormatAcceptable() {

        //test for the age input
        try {
            int tester = Integer.parseInt(fields[2].getText(),
10);

            if (tester < 0)
                return "NO NEGATIVE AGE ACCEPTED";
        } catch (NumberFormatException e) {
            return "ERROR IN THE NUMBER FORMAT FOR AGE";
        }
    }

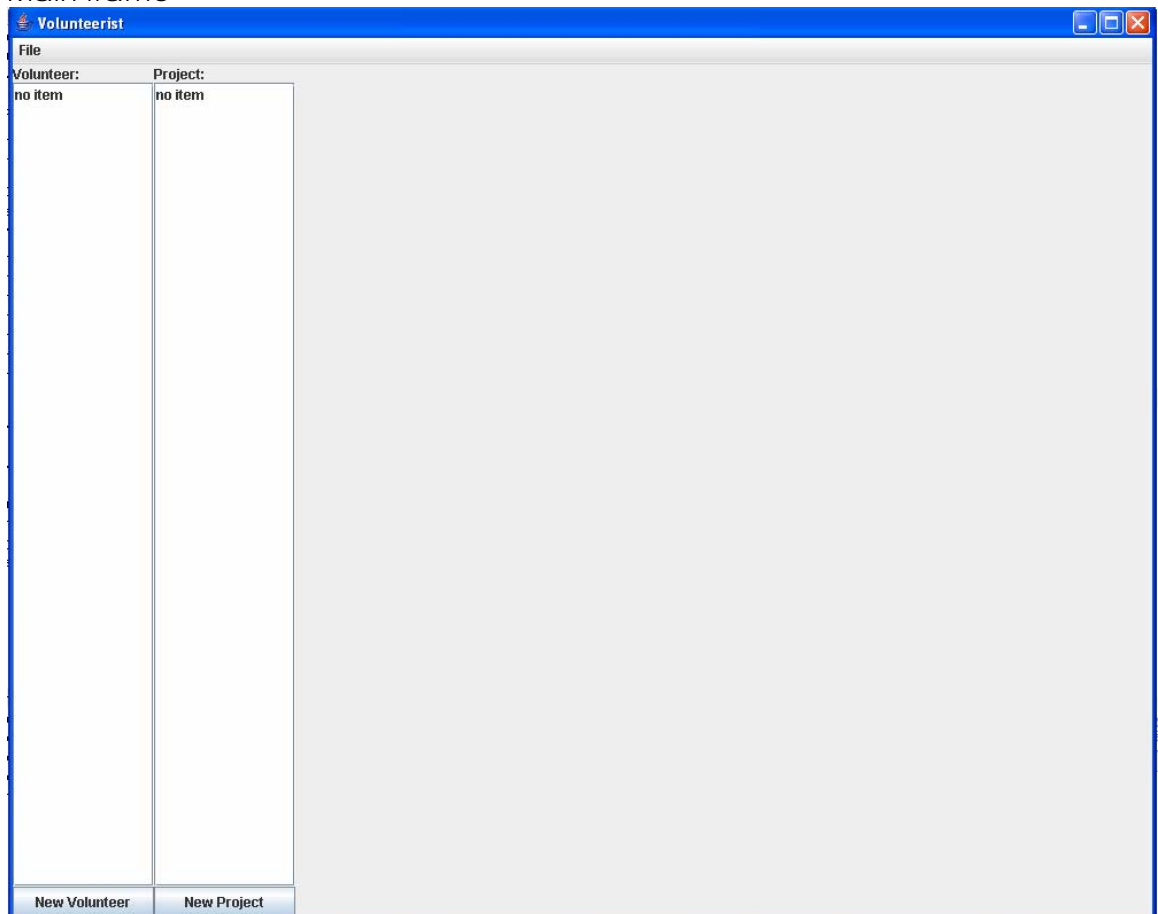
```

```
        //check for the sex input
        if (fields[5].getText().length() != 1)
            return "PLEASE INSERT ONLY ONE LETTER FOR THE SEX
INPUT";
        if (!(fields[5].getText().equalsIgnoreCase("M") ||
fields[5].getText()
            .equalsIgnoreCase("F")))
            return "WRONG INPUT FOR VOLUNTEER'S SEX";

        return "PASS";
    }
}
```

C2: Usability

- Main frame

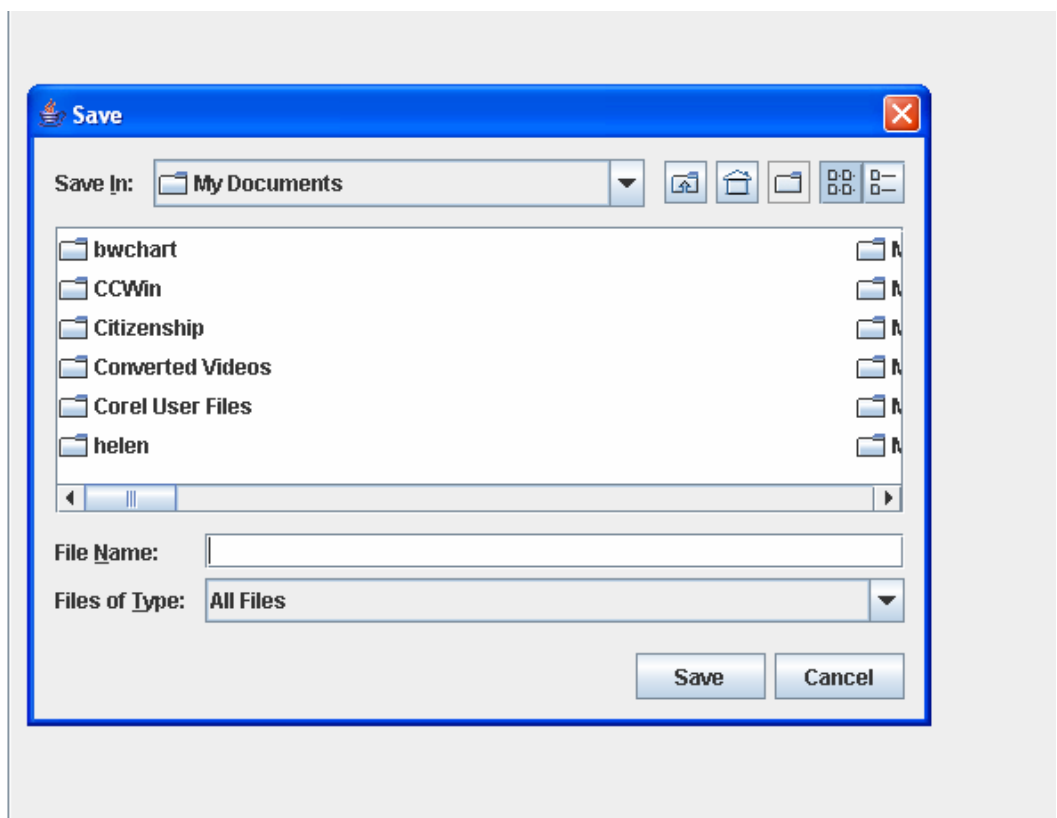


The two lists: "Volunteer" and "Project", appears at the right and has no items in it. This is because no database has been opened/created.

- Open File

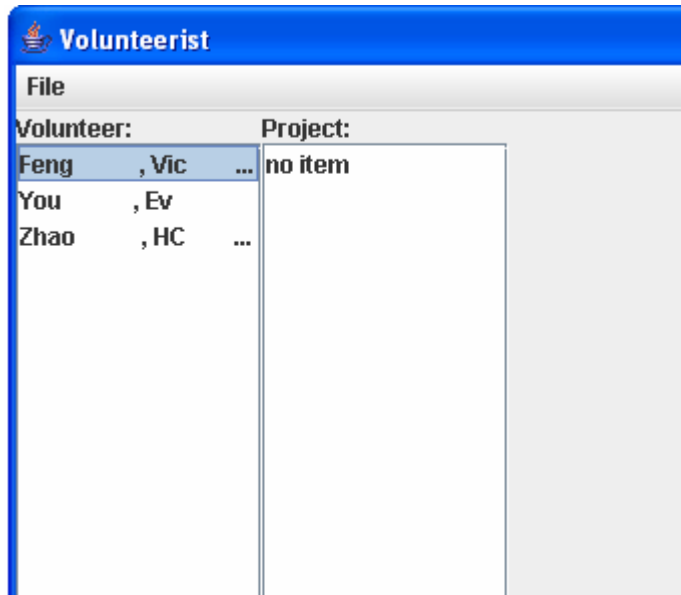


The Menu at the top provides access to opening the file



A JFileChooser is used to locate the file.

- When file is opened, the content is automatically shown in JList



The content are always organized in alphabetical order.

- When creating new file, click "New Volunteer" or "New Project" and a chart will appear:

The screenshot shows the 'Volunteerist' application window with the 'New Volunteer Profile' form open. The form contains several input fields and checkboxes. The 'Volunteer:' column in the table is highlighted, and the 'Project:' column contains 'no item'. The form fields are: Last Name, First Name, Age, Birthday (YYMMDD), Address, Sex (M/F), Email Address, and Phone Number. At the bottom, there is a 'Skills' section with checkboxes for Driving, Watching Children, Working in Alcoholic Environment, Lifting 20kg or up, Music Skill, and Art Skill. The 'New Volunteer' and 'New Project' buttons are visible at the bottom left, and 'Save' and 'Clear' buttons are at the bottom right.

New Volunteer Profile

Last Name:

First Name:

Age:

Birthday (YYMMDD):

Address:

Sex (M/F):

Email Address:

Phone Number:

Skills

Driving Watching Children Working in Alcoholic Environment Lifting 20kg or up Music Skill Art Skill

New Volunteer New Project Save Clear

Fill out the chart and click "Save", the record will be saved into the file.

- Click the items inside the JList, the information in the record will appear:

The screenshot shows a window titled "Volunteerist" with a menu bar containing "File". Below the menu bar, there are two columns: "Volunteer:" and "Project:". The "Volunteer:" column contains a list of names: "Feng ,Vic ...", "You ,Ev", and "Zhao ,HC ...". The "Project:" column contains "no item". To the right of these columns is a form titled "New Volunteer Profile". The form contains the following fields and options:

- Last Name:
- First Name:
- Age:
- Birthday (YY/MM/DD):
- Address:
- Sex (M/F):
- Email Address:
- Phone Number:
- Skills: Driving Watching Children Working in Alcoholic Environment Lifting 20kg or up Music Skill Art Skill

At the bottom of the window, there are two buttons: "New Volunteer" and "New Project". At the bottom right, there are two buttons: "Save" and "Clear".

- Error Handling: invalid data will not be processed; user will be given a warning message and user is given the chance to retype the data

Volunteerist

File

Volunteer:	Project:
Feng , Vic ...	no item
Jiang , Andy ...	
You , Ev ...	
Zhao , HC ...	

New Volunteer Profile

Last Name:

First Name:

Age:

Birthday (YY/MM/DD):

Address:

Sex (MF):

Email Address:

Phone Number:

Skills

Driving Watching Children Working in Alcoholic Environment Lifting 20kg or up Music Skill Art Skill

ERROR

ERROR IN THE NUMBER FORMAT FOR AGE

OK

