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

import java.util.*;

public class Maps
{
    // window frame
    private JFrame frame;
    private JPanel contentPane;

    // list and position
    private TreeMap<Integer, Player> map;

    // labels
    private JLabel nameLabel;
    private JLabel numLabel;
    private JLabel positionLabel;
    private JLabel avgPtsLabel;
    private JLabel avgRbndsLabel;
    private JLabel avgAssistsLabel;

    // text fields
    private JTextField playerName;
    private JTextField playerNum;
    private JTextField playerPosition;
    private JTextField playerAvgPts;
    private JTextField playerAvgRbnds;
    private JTextField playerAvgAssists;

    // team view fields
    private JTextArea textArea;
    private JScrollPane scrollArea;

    public static void main (String[] args)
    {
        Maps GUITabs = new Maps();
        GUITabs.start();
    }

    public void start()
    {
        frame = new JFrame("L11_Tabs");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = (JPanel)frame.getContentPane();

        makeMenus();
        makeContent();

        frame.pack();
        frame.setVisible(true);
    }

    private void makeMenus()
    {
        JMenuBar menuBar;

        menuBar = new JMenuBar();
        frame.setJMenuBar(menuBar);

        // set up menus
        menuBar.add(makeFileMenu());
        menuBar.add(makeViewMenu());
        menuBar.add(makeHelpMenu());
    }
```

```
private JMenu makeFileMenu()
{
    JMenu menu;
    JMenuItem menuItem;

    // set up the File menu
    menu = new JMenu("File");
    menu.setMnemonic(KeyEvent.VK_F);

    // add Open menu item
    menuItem = new JMenuItem("Open...");
    menuItem.setMnemonic(KeyEvent.VK_O);
    menuItem.addActionListener(new OpenMenuItemListener());
    menuItem.setAccelerator(
            KeyStroke.getKeyStroke(KeyEvent.VK_O,
                                   Event.CTRL_MASK));
    menu.add(menuItem);

    // add Exit menu item
    menu.addSeparator();
    menuItem = new JMenuItem("Exit");
    menuItem.setMnemonic(KeyEvent.VK_X);
    menuItem.addActionListener(new ExitMenuItemListener());
    menuItem.setAccelerator(
            KeyStroke.getKeyStroke(KeyEvent.VK_Q,
                                   Event.CTRL_MASK));
    menu.add(menuItem);

    return menu;
}

private JMenu makeViewMenu()
{
    JMenu menu;
    JMenuItem menuItem;

    // set up the View menu
    menu = new JMenu("View");
    menu.setMnemonic(KeyEvent.VK_V);

    // add Next Player menu item
    menuItem = new JMenuItem("Next Player");
    menuItem.addActionListener(new NextMenuItemListener());
    menuItem.setAccelerator(
            KeyStroke.getKeyStroke(KeyEvent.VK_DOWN,
                                   Event.ALT_MASK));
    menu.add(menuItem);

    // add Previous Player menu item
    menuItem = new JMenuItem("Previous Player");
    menuItem.addActionListener(new PrevMenuItemListener());
    menuItem.setAccelerator(
            KeyStroke.getKeyStroke(KeyEvent.VK_UP,
                                    Event.ALT_MASK));
    menu.add(menuItem);

    // add Find menu item
    menu.addSeparator();
    menuItem = new JMenuItem("Find a Player");
    menuItem.setMnemonic(KeyEvent.VK_F);
    menuItem.addActionListener(new FindMenuItemListener());
    menuItem.setAccelerator(
            KeyStroke.getKeyStroke(KeyEvent.VK_F,
                                   Event.CTRL_MASK));
    menu.add(menuItem);

    return menu;
}

private JMenu makeHelpMenu()
{
```

```
    JMenu menu;
    JMenuItem menuItem;

    // set up the Help menu
    menu = new JMenu("Help");
    menu.setMnemonic(KeyEvent.VK_H);

    // add About menu item
    menuItem = new JMenuItem("About L11-Tabs");
    menuItem.setMnemonic(KeyEvent.VK_A);
    menuItem.addActionListener(new AboutMenuItemListener());
    menu.add(menuItem);

    return menu;
}

private void makeContent()
{
    contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y_AXIS));
    contentPane.setBorder(BorderFactory.createEmptyBorder(6,6,6,6));
    JTabbedPane tabby = new JTabbedPane();

    // player panel
    JPanel panel = new JPanel();
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
    panel.setBorder(BorderFactory.createEmptyBorder(6,6,6,6));

    // player name
    nameLabel = new JLabel("Player Name:");
    nameLabel.setFont(new Font("Trebuchet MS",Font.BOLD + Font.ITALIC,14));
    panel.add(nameLabel);
    playerName = new JTextField();
    playerName.setFont(new Font("Trebuchet MS",Font.PLAIN,14));
    playerName.setForeground(Color.BLUE);
    panel.add(playerName);

    // player number
    numLabel = new JLabel("Player Number:");
    numLabel.setFont(new Font("Trebuchet MS",Font.BOLD + Font.ITALIC,14));
    panel.add(numLabel);
    playerNum = new JTextField();
    playerNum.setFont(new Font("Trebuchet MS",Font.PLAIN,14));
    playerNum.setForeground(Color.BLUE);
    panel.add(playerNum);

    // player position
    positionLabel = new JLabel("Position:");
    positionLabel.setFont(new Font("Trebuchet MS",Font.BOLD + Font.ITALIC,14));
    panel.add(positionLabel);
    playerPosition = new JTextField();
    playerPosition.setFont(new Font("Trebuchet MS",Font.PLAIN,14));
    playerPosition.setForeground(Color.BLUE);
    panel.add(playerPosition);

    // average points
    avgPtsLabel = new JLabel("Average Points per Game:");
    avgPtsLabel.setFont(new Font("Trebuchet MS",Font.BOLD + Font.ITALIC,14));
    panel.add(avgPtsLabel);
    playerAvgPts = new JTextField();
    playerAvgPts.setFont(new Font("Trebuchet MS",Font.PLAIN,14));
    playerAvgPts.setForeground(Color.BLUE);
    panel.add(playerAvgPts);

    // average rebounds
    avgRbndsLabel = new JLabel("Average Rebounds per Game:");
    avgRbndsLabel.setFont(new Font("Trebuchet MS",Font.BOLD + Font.ITALIC,14));
    panel.add(avgRbndsLabel);
    playerAvgRbnds = new JTextField();
    playerAvgRbnds.setFont(new Font("Trebuchet MS",Font.PLAIN,14));
    playerAvgRbnds.setForeground(Color.BLUE);
    panel.add(playerAvgRbnds);
```

```
    // average assists
    avgAssistsLabel = new JLabel("Average Assists per Game:");
    avgAssistsLabel.setFont(new Font("Trebuchet MS",Font.BOLD + Font.ITALIC,14));
    panel.add(avgAssistsLabel);
    playerAvgAssists = new JTextField();
    playerAvgAssists.setFont(new Font("Trebuchet MS",Font.PLAIN,14));
    playerAvgAssists.setForeground(Color.BLUE);
    panel.add(playerAvgAssists);

    tabby.addTab("Player View", panel);
    tabby.setMnemonicAt(0, KeyEvent.VK_P);

    panel = new JPanel();
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
    panel.setBorder(BorderFactory.createEmptyBorder(6,6,6,6));
    textArea = new JTextArea(15,25);
    scrollArea = new JScrollPane(textArea);
    scrollArea.setVerticalScrollBarPolicy(
            ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
    scrollArea.setHorizontalScrollBarPolicy(
            ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
    panel.add(scrollArea);

    tabby.addTab("Team View", panel);
    tabby.setMnemonicAt(1, KeyEvent.VK_T);

    contentPane.add(tabby);
}

private class AboutMenuItemListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        JOptionPane.showMessageDialog(frame,
                "L11-Tabs\n\nVersion 1.0\nBuild B20080407-1511\n\n" +
                    "(c) Copyright Merrill Hall 2008\nAll rights reserved\n\n" +
                    "Visit /\nEducation To Go\n" +
                    "Intermediate Java Course",
                "About L11-Tabs",
                JOptionPane.INFORMATION_MESSAGE);
    }
}

private class ExitMenuItemListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }
}

private class OpenMenuItemListener implements ActionListener
{
    public void actionPerformed(ActionEvent ae)
    {
        JFileChooser fc = new JFileChooser();
        fc.showOpenDialog(frame);
        File playerFile = fc.getSelectedFile();
        if (playerFile == null)
            return;
        map = new TreeMap<Integer, Player>();
        try
        {
            Scanner scan = new Scanner(playerFile);
            while (scan.hasNext())
            {
                String name = scan.next() + " " + scan.next();
                int nbr = scan.nextInt();
                char position = scan.next().charAt(0);
                double avgPoints = scan.nextDouble();
                double avgRebounds = scan.nextDouble();
                double avgAssists = scan.nextDouble();
```

```
                map.put(new Integer(nbr),
                        new Player(name, nbr, position, avgPoints, avgRebounds, avgAssists));
            }

            scan.close();
        }
        catch(IOException e)
        {
            JOptionPane.showMessageDialog(frame,
                    "I/O error in file\n\n     " +
                            playerFile.getName() +
                            "\n\nThis program will close",
                    "I/O Error",
                    JOptionPane.ERROR_MESSAGE);
            System.exit(1);
        }

        findPlayer();

        for (Player p : map.values())
           {
               textArea.setText(textArea.getText() + p.toString() + "\n\n");
           }

    }
}

private class NextMenuItemListener implements ActionListener
{
    public void actionPerformed(ActionEvent ae)
    {
        if (map == null || map.size() == 0)
            return;

        Map.Entry<Integer, Player> entry = map.higherEntry(Integer.parseInt(playerNum.getText()));

        if (entry == null)
            JOptionPane.showMessageDialog(frame,
                    "There are no more players.\nYou have reached the end of the list.",
                    "End of List",
                    JOptionPane.WARNING_MESSAGE);
        else
            getPlayer(entry.getValue());
    }
}

private class PrevMenuItemListener implements ActionListener
{
    public void actionPerformed(ActionEvent ae)
    {
        if (map == null || map.size() == 0)
            return;

        Map.Entry<Integer, Player> entry = map.lowerEntry(Integer.parseInt(playerNum.getText()));

        if (entry == null)
            JOptionPane.showMessageDialog(frame,
                    "There are no previous players.\nYou are at the start of the list.",
                    "Start of List",
                    JOptionPane.WARNING_MESSAGE);
        else
            getPlayer(entry.getValue());
    }
}

private class FindMenuItemListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        findPlayer();
    }
}
```

```
    private void getPlayer(Player p)
    {
        playerName.setText(p.getName());
        playerNum.setText("" + p.getNum());
        playerPosition.setText(p.getPosition());
        playerAvgPts.setText("" + p.getAvgPoints());
        playerAvgRbnds.setText("" + p.getAvgRebounds());
        playerAvgAssists.setText("" + p.getAvgAssists());
    }

    private void findPlayer()
    {
        boolean isGoodNumber = false;
        Integer playerNum = new Integer(0);
        while (!isGoodNumber)
        {
            try
            {
                playerNum = new Integer(Integer.parseInt(JOptionPane.showInputDialog(frame,
                                                         "Enter a player number:",
                                                         "Player Entry",
                                                         JOptionPane.QUESTION_MESSAGE)));
                isGoodNumber = true;
            }
            catch(NumberFormatException nfe)
            {
                JOptionPane.showMessageDialog(frame,
                        "That wasn't a player number!",
                        "Player Number Error",
                        JOptionPane.ERROR_MESSAGE);
            }

            if(isGoodNumber)
            {
                Player p = map.get(playerNum);
                if (p == null)
                {
                    JOptionPane.showMessageDialog(frame,
                            "Player number " + playerNum.intValue() + " does not exist!",
                            "Player Number Error",
                            JOptionPane.ERROR_MESSAGE);
                    isGoodNumber = false;
                }
                else
                    getPlayer(p);
            }
        }
    }
}
```