```
 1: /*
 2:
 3: Jared Dyreson
 4: CWID: 889546529
 5: AboutPage.java ->
 6:
 7: */
 8:
 9: import javax.swing.*;
10: import java.text.MessageFormat;
11: import java.awt.*;
12: import javax.swing.JFrame;
13: import java.awt.event.*;
14: import java.lang.Math;
15:
16: import java.io.*;
17: import java.util.*;
18: import java.util.List;
19: import java.util.ArrayList;
20: import java.util.Arrays;
21:
22: public class AboutPage extends JFrame implements ActionListener{
23:         // Auto generated with caffine and ureadahead.service
24:
25:         private final int FRAME_HEIGHT = 424, FRAME_WIDTH = 373;
26:         private String about_message = "Zip Viewer\n1.0.0\nA zip manager written in pure Java.\nThis is a direc
27:         private JLabel center_label = new JLabel(about_message);
28:
29:         private JButton close = new JButton("Close");
30:         private JButton credits = new JButton("Credits");
31:
32:         private JPanel sub_panel = new JPanel();
33:
34:         public AboutPage(){
35:                 super("About Zip Viewer");
36:
37:                 this.setSize(FRAME_HEIGHT, FRAME_WIDTH);
38:                 this.setLocationRelativeTo(null);
39:                 this.setLayout(new FlowLayout());
40:                 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
41:                 //this.add(sub_panel, BorderLayout.SOUTH);
42:
43:                 close.addActionListener(this);
44:                 credits.addActionListener(this);
45:
46:                 sub_panel.setLayout(new FlowLayout());
47:                 this.add(center_label, BorderLayout.CENTER);
48:                 this.add(credits, BorderLayout.WEST);
```

```
49:                 this.add(close, BorderLayout.EAST);
50:
51:                 //this.add(sub_panel, BorderLayout.SOUTH);
52:
53:
54:         }
55:         @Override
56:         public void actionPerformed(ActionEvent event){
57:                 Object source = event.getSource();
58:
59:                 if(source == close){
60:                         this.dispose();
61:                 }
62:                 else{
63:                         System.out.println("Credits...my mom");
64:                 }
65:
66:         }
67: }
```

```
 1: /*
 2:
 3: Jared Dyreson
 4: CWID: 889546529
 5: CreateNewArchive.java ->
 6:
 7: */
 8:
 9: import javax.swing.*;
10: import java.text.MessageFormat;
11: import java.awt.*;
12: import javax.swing.JFrame;
13: import java.awt.event.*;
14: import java.lang.Math;
15:
16: import java.io.*;
17: import java.util.*;
18: import java.util.List;
19: import java.util.ArrayList;
20: import java.util.Arrays;
21:
22:
23: public class CreateNewArchive extends JFrame implements ActionListener{
24:         // Auto generated with caffine and pppd-dns.service
25:
26:         private final int FRAME_HEIGHT = 522, FRAME_WIDTH = 193;
27:         JButton create = new JButton("Create");
28:         JButton close = new JButton("Close");
29:
30:         JLabel location_label = new JLabel("Location");
31:         JLabel archive_name_label = new JLabel("Archive Name:");
32:
33:         JTextField archive_name_field = new JTextField(20);
34:
35:         JPanel bottom_elements = new JPanel();
36:         JPanel center_elements = new JPanel();
37:
38:         ZipBackend zipper = new ZipBackend();
39:
40:         public CreateNewArchive(){
41:                 super("Create New Archive");
42:
43:                 this.setSize(FRAME_HEIGHT, FRAME_WIDTH);
44:                 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45:                 this.setLocationRelativeTo(null);
46:
47:                 create.addActionListener(this);
48:                 close.addActionListener(this);
```

```
49:                     bottom_elements.setLayout(new FlowLayout());
50:                     bottom_elements.add(create);
51:                     bottom_elements.add(close);
52:
53:                     center_elements.setLayout(new FlowLayout());
54:                     center_elements.add(location_label);
55:                     center_elements.add(archive_name_field);
56:
57:                     this.add(bottom_elements, BorderLayout.SOUTH);
58:                     this.add(center_elements, BorderLayout.CENTER);
59:             }
60:             @Override
61:             public void actionPerformed(ActionEvent event){
62:                     Object action_performed = event.getSource();
63:
64:                     if(action_performed == close){
65:                             this.dispose();
66:                     }
67:                     else if(action_performed == create){
68:                             String path_to_empty = archive_name_field.getText();
69:                             try{
70:
71:                                     zipper.create_empty_archive(path_to_empty+".zip");
72:                             }
73:                             catch(Exception error){
74:                                     System.out.println(error);
75:                             }
76:                     }
77:             }
78: }
```

```
 1: /*
 2:
 3: Jared Dyreson
 4: CWID: 889546529
 5: Driver.java ->
 6:
 7: */
 8:
 9: import java.io.File;
10: import javax.swing.JFileChooser;
11: import javax.swing.filechooser.FileSystemView;
12:
13: public class Driver {
14:         public static void main(String[] args){
15:                 // Auto generated with caffine and NetworkManager-dispatcher.service
16:                 ZipWindow zipper = new ZipWindow();
17:                 zipper.setVisible(true);
18:         }
19: }
```

```
 1: /*
 2:
 3: Jared Dyreson
 4: CWID: 889546529
 5: FileHandler.java ->
 6:
 7: */
 8: import java.io.IOException;
 9: import java.nio.file.Files;
10: import java.nio.file.Path;
11: import java.nio.file.Paths;
12:
13: import java.io.File;
14: import java.text.SimpleDateFormat;
15: import java.text.MessageFormat;
16:
17: // CONTENTS
18: // Get specific information about files in the zip buffer
19: // file name (relative and absolute)
20: // modification date
21: // mime type (what kind of file is it)
22: // current file size
23: // return the following information into a string that can
24: // be used in a JLabel object
25:
26:
27: public class FileHandler {
28:         // Auto generated with caffine and networking.service
29:
30:         private String path = "";
31:         private File file;
32:
33:         public FileHandler(String path_to_file){
34:                 this.path = path_to_file;
35:                 this.file = new File(this.path);
36:         }
37:
38:         public FileHandler(File file_path_object){
39:                 this.file = file_path_object;
40:                 this.path = this.get_path();
41:         }
42:
43:         public String get_path(){
44:                 return this.file.getName();
45:         }
46:         public String get_absolute_path(){
47:                 if(this.file.exists()){ return this.file.getAbsolutePath(); }
48:                 return "";
```

```
49:         }
50:         public String get_last_modified(){
51:                 SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");
52:                 return formatter.format(this.file.lastModified());
53:         }
54:         public String get_mime_type() throws IOException{
55:                 if(this.file.isDirectory()){ return "dir"; }
56:                 Path source = Paths.get(this.path);
57:                 return Files.probeContentType(source);
58:         }
59:
60:         public String get_file_size(){
61:                 return String.valueOf(this.file.length());
62:         }
63:
64:         public String row_information() throws IOException{
65:                 return MessageFormat.format("{0}\t\t\t\t{1}\t\t{2}\t\t\t\t{3}", this.get_path(), this.get_file_
66:         }
67:
68:         public static void main(String args[]){
69:                 File[] files_in_wd = new File(".").listFiles();
70:                 for(int i = 0; i < files_in_wd.length; ++i){
71:                         FileHandler f = new FileHandler(files_in_wd[i]);
72:                         try{
73:                                 System.out.println(f.row_information());
74:                         }
75:                         catch(Exception error){
76:                                 System.out.println(error);
77:                         }
78:                 }
79:
80:
81:         }
82: }
```

```
 1: /*
 2:
 3: Jared Dyreson
 4: CWID: 889546529
 5: ZipBackend.java ->
 6:
 7: */
 8:
 9: import java.io.*;
10: import java.util.*;
11: import java.util.zip.*;
12:
13: public class ZipBackend {
14:         // Auto generated with caffine and accounts-daemon.service
15:
16:         private ZipOutputStream zip_stream_for_class;
17:         private String path = "";
18:
19:         //public ZipBackend(String zip_path){
20:                 //this.path = zip_path;
21:         //}
22:
23:         public void zip_contents(List<String> file_contents, String path_to_file) throws IOException{
24:                 FileOutputStream file_stream = new FileOutputStream(path_to_file);
25:                 ZipOutputStream zip_out = new ZipOutputStream(file_stream);
26:                 File destination_zip = new File(path_to_file);
27:                 //if(destination_zip.exists()){
28:                         //System.out.println("File already exists");
29:                         //return;
30:                 //}
31:
32:                 for(String path : file_contents){
33:                         File zip_this_file = new File(path);
34:                         FileInputStream input_stream = new FileInputStream(zip_this_file);
35:                         ZipEntry zip_entry = new ZipEntry(zip_this_file.getName());
36:                         zip_out.putNextEntry(zip_entry);
37:
38:                         byte[] bytes = new byte[1024];
39:                         int len;
40:                         while((len = input_stream.read(bytes)) >= 0){
41:                                 zip_out.write(bytes, 0, len);
42:                         }
43:                         input_stream.close();
44:                 }
45:                 zip_out.close();
46:                 file_stream.close();
47:         }
48:
```

```
49:         public void create_empty_archive(String path_to_file) throws IOException{
50:                 File zip_file = new File(path_to_file);
51:                 FileOutputStream stream = new FileOutputStream(zip_file);
52:                 BufferedOutputStream buffered = new BufferedOutputStream(stream);
53:                 ZipOutputStream zip_stream = new ZipOutputStream(stream);
54:
55:                 //ZipEntry fake_entry = new ZipEntry("fake_meta");
56:                 //zip_stream.putNextEntry(fake_entry);
57:                 //zip_stream.closeEntry();
58:                 zip_stream.close();
59:         }
60:
61:         //public void dump_contents(){
62:
63:         //}
64:
65:         //public void list_contents(String path_to_archive){
66:                 //// this will only print with depth of 1
67:
68:
69:         //}
70: }
```

```
 1: /*
 2:
 3: Jared Dyreson
 4: CWID: 889546529
 5: ZipWindow.java ->
 6:
 7: */
 8:
 9: import javax.swing.JFileChooser;
10: import javax.swing.filechooser.FileSystemView;
11:
12: import java.awt.*;
13: import java.awt.event.*;
14: import java.lang.Math;
15: import java.text.MessageFormat;
16: import javax.swing.*;
17: import javax.swing.JFrame;
18:
19: import java.io.*;
20: import java.util.*;
21: import java.util.ArrayList;
22: import java.util.Arrays;
23: import java.util.List;
24:
25: // TODO
26: // contents of the zip needs to be displayed in the middle of the screen
27: // Name           Size               Type                    Modified
28: // ------------------------------------------------------------------
29: // [filename]     [file size]        [mime type]             [last modified]
30: //
31: // better format the buttons
32: // add functionality to the drop down options
33: //      mapping the about page and creating a new empty archive
34: // create the functionality for appending to the archive
35: // create the functionality for dumping the archive
36:
37: public class ZipWindow extends JFrame implements ActionListener{
38:         // Auto generated with caffine and cups.service
39:         final int FRAME_WIDTH = 959, FRAME_HEIGHT = 1052;
40:
41:         JPanel file_manipulation_buttons = new JPanel();
42:         JPanel menu_panel = new JPanel();
43:
44:         JButton extract_button = new JButton("Extract");
45:         JButton add_files = new JButton("+");
46:
47:         JButton next = new JButton("<--");
48:         JButton previous = new JButton("-->");
```

```
49:         JButton home_button = new JButton("Root folder");
50:         JButton about = new JButton("about");
51:         JButton list_contents = new JButton("list contents");
52:
53:         JLabel location_label = new JLabel("Location:");
54:
55:         JTextField path_in_archive = new JTextField();
56:
57:         JToolBar toolbar = new JToolBar();
58:
59:         String[] menu_options = {"File", "Edit", "View", "Help"};
60:         String[] file_menu_options = {"New Archive", "Open", "Extract Files", "Close"};
61:         List<String> file_manifest = new ArrayList<>();
62:         List<JPanel> contents_of_nth_depth = new ArrayList<>();
63:
64:         JMenuBar menu_bar = new JMenuBar();
65:         ZipBackend zipper = new ZipBackend();
66:
67:         public ZipWindow(){
68:                 super("Zip Viewer");
69:
70:                 setSize(FRAME_WIDTH, FRAME_HEIGHT);
71:                 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
72:                 setLocationRelativeTo(null);
73:                 add_files.addActionListener(this);
74:                 extract_button.addActionListener(this);
75:                 about.addActionListener(this);
76:                 list_contents.addActionListener(this);
77:
78:                 toolbar.setRollover(true);
79:                 for(int i = 0; i < menu_options.length; ++i){
80:                         menu_bar.add(new JMenu(menu_options[i]));
81:                 }
82:
83:                 toolbar.add(menu_bar);
84:
85:                 file_manipulation_buttons.setLayout(new FlowLayout());
86:                 file_manipulation_buttons.add(extract_button);
87:                 file_manipulation_buttons.add(add_files);
88:                 file_manipulation_buttons.add(about);
89:                 file_manipulation_buttons.add(list_contents);
90:
91:                 menu_panel.setLayout(new BorderLayout(2, 1));
92:                 menu_panel.add(toolbar, BorderLayout.WEST);
93:                 menu_panel.add(file_manipulation_buttons, BorderLayout.EAST);
94:                 this.add(menu_panel, BorderLayout.NORTH);
95:
96:         }
```

```
 97:
 98:         public String get_file(){
 99:                 String file_path = "";
100:                 JFileChooser jfc = new JFileChooser(FileSystemView.getFileSystemView().getHomeDirectory());
101:                 int returnValue = jfc.showOpenDialog(null);
102:                 if (returnValue == JFileChooser.APPROVE_OPTION) {
103:                         File selectedFile = jfc.getSelectedFile();
104:                         file_path = selectedFile.getAbsolutePath();
105:                 }
106:                 return file_path;
107:
108:         }
109:         @Override
110:         public void actionPerformed(ActionEvent event){
111:                 Object source = event.getSource();
112:                 if(source == add_files){
113:                         String path_to_file = get_file();
114:                         file_manifest.add(path_to_file);
115:                 }
116:                 else if(source == extract_button){
117:                         //String zip_destination_name =
118:                         try{
119:                                 zipper.zip_contents(file_manifest, "example.zip");
120:                         }
121:                         catch(Exception error){
122:                                 System.out.println(error);
123:                         }
124:                 }
125:                 else if(source == about){
126:                         AboutPage about_page = new AboutPage();
127:                         about_page.setVisible(true);
128:                 }
129:                 else if(source == list_contents){
130:                         for(int i = 0; i < file_manifest.size(); ++i){
131:                                 try{
132:                                         FileHandler file = new FileHandler(file_manifest.get(i));
133:                                         JPanel file_panel = new JPanel();
134:                                         file_panel.setLayout(new FlowLayout());
135:                                         file_panel.add(new JLabel(file.get_last_modified()));
136:                                         file_panel.add(new JLabel(file.get_mime_type()));
137:                                         contents_of_nth_depth.add(file_panel);
138:                                         /*menu_panel.add(file_panel, BorderLayout.CENTER);*/
139:                                         /*this.add(menu_panel, BorderLayout.CENTER);*/
140:                                 }
141:                                 catch(Exception error){
142:                                         System.out.println(error);
143:                                 }
144:
```

```
145:                            }
146:                    }
147:            }
148:
149: }
```