CST8221 – Java Application Programming
Hybrid Activity #7
File Chooser Dialogs

Terminology

Just about any application you write needs to have a mechanism for opening and saving files A good file dialog box that shows files and directories and in the same time allows the user to navigate the file system is very difficult to write from scratch. Fortunately, Swing provides a ***JFileChooser*** class dialogs which allows you to display a file dialog box very similar to the one your native platform application offer to the user.

The Nature of Things

In the AWT, you can use the *FileDialog* class, but that is a heavyweight dialog that lacks the flexibility of the Swing components you have seen so far. The ***JFileChooser*** is Swing's answer to the AWT *FileDialog*. The Swing package also contains a helper dialog for choosing colors ***JColorChooser*** (a common task in the configuration area of applications). Note that the ***JFileChooser*** class is not a subclass of *JDialog* class. So instead of calling *setVisible(true)*, you have to call **showOpenDialog()** or **showSaveDialog()** to open or save a file correspondingly. The button for accepting the file is then automatically labeled Open or Save. You can also supply your own button label, but you have to use the more generic *showDialog()* method.
Here are the steps needed to create and show a file dialog box and retrieve what the user have chosen from the box:

1. Instantiate a *JFileChooser* object:

   ```
   fileChooser = new JFileChooser();
   ```

   Reusing the same file chooser object is a very good idea because the constructor can be quite slow especially with mapped network drives.

2. Change the default folder starting folder. For example, to use the current working directory (folder) you need to supply a File object like this

   ```
       fileChooser.setCurrentDirectory(new File("."));
   ```

3. To enable the user to select multiple files (not a very common case) you can use

   ```
       fileChooser.setAcceptAllFileFilterUsed(true);
   ```

   The default value for this property is false, which means that only a single selection is allowed.

4. If you want to restrict the display of files in the dialog to those of particular type (for example, .java), then you have to use a file filter. Swing provides a convenient implementation of the abstract class ***javax.swing.filechooser.FileFilter*** called ***avax.swing.filechooser.FileNameExtensionFilter***. For example, the lines bellow

   ```
   FileFilter filter = new FileNameExtensionFilter("Text files (.txt)","txt");
   fileChooser.addChoosableFileFilter(filter);
   fileChooser.setAcceptAllFileFilterUsed(false);
   ```

will restrict the dialog to displaying .java files only. The default option is to display all files, which is equivalent to: `fileChooser.setAcceptAllFileFilterUsed(true);`
Pay attention to the package because there is a class with the same name in AWT package.

5. By default, a user can select only files with a file chooser. If you want to allow the user to select directories, have to use the *setFileSelectionMode()* method with one of the following constants: *JFileChooser.FILES_ONLY* (the default), *JFileChooser.DIRECTORIES_ONLY* , and *JFileChooser.FILES_AND_ DIRECTORIES.*

6. When you want to show the dialog box, you must call *showOpenDialog()* or *showSaveDialog().* You must provide a parent component in these calls:

   ```
   returnOption = fileChooser.showOpenDialog(FileDialogDemo.this);
   ```

   or

   ```
   returnOption = fileChooser.showOpenDialog(null);
   ```

   If you provide *null*, the dialog will appear in the center of the screen. These calls return only when the user has approved, canceled, or dismissed the file dialog. The returned integer value is one of the following constants*: JFileChooser.APPROVE_OPTION*, *JFileChooser.CANCEL_OPTION, and JFileChooser.ERROR_OPTION.*

7. Finally, to get the selected file you call

   ```
       File file = fileChooser.getSelectedFile();
   ```

Once you get the file, you can use for whatever purposes of you application dictates.
The standard **JFileChooser** can be customized to great extend. For example, using the *accessory* property you can build a preview panel. For more details, please refer to the references below.
Swing provides one more standard chooser – ***JColorChooser*** which we will discuss later.

References
Textbook 1: Ch. 15.6
Java Swing, second edition.
Links:
https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html

The JavaFX API includes a similar component (control) with a similar name ***FileChooser***. It provides the same functionality but the appearance has been improved. JavaFX also includes a color chooser, but the name of the control has been changed to ***ColorPicker***. Visit the link below for more details and demo programs (one sample is included in the HA#7 code examples). When you examine the JavaFX ***FileChooser*** demo pay close attention to the use of ***Desktop*** class.

http://docs.oracle.com/javafx/2/ui_controls/file-chooser.htm

Code Examples

You will find the examples in ***CST8221_HA07_code_examples.zip***.

Exercise

Compile, and run the code example **FileDialogDemo.java**. You can use the same file to open it with the example application and save it under different name. Once you see how the example works, explore very carefully the code. Try to make the suggested modifications. Pay attention to the fact that the save dialog will overwrite an existing file without any warning. Try to modify the program so that it displays a dialog box asking the user for a confirmation.
Visit the reference links provided above. You will find more demos exploring the File Chooser dialog.
Compile, and run the code example **FileDialogDemoFX.java**. Examine the code. Pay attention to the ***Desktop*** class.

Questions

Q1. Can you filter the type of the files displayed in the file chooser dialog box?
Q2. Can you select multiple files?
Q3. Can you select a directory?
Q4. Does JavaFX provide similar components? If yes, do they provide the same functionality?

Submission

No submission is required for this activity.

Marks

No marks are allocated for this activity, but remember that understanding how *the demo example works* could be **crucial** for your mid-term test success.

And do not forget that:

"It is good to have a choice." Anonymous

but also never forget that*:*

"But to make the right choice you have to be well informed." Another Anonymous