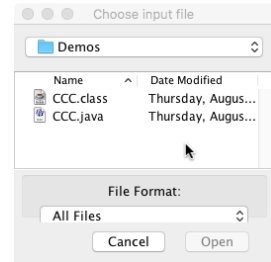


Getting a path using a JFileChooser

Class `javax.swing.JFileChooser` is used to open a dialog window that the user can use to navigate to a particular directory or file on their hard drive and select it for reading or writing. The example to the right is used to create a `java.nio.file.Path` object for a file to be read. The user navigates to the directory that contains the file, selects the file, and clicks button *Open*.

Your code gets to put the title at the top, in this case, *Choose input file*.



Getting a Path to an input file

Below is a method that uses a `JFileChooser` to create a `Path` to a file to be read. Given the `Path`, one can then create a `BufferedReader`, as usual, in order to read the file. We discuss each statement of the method body.

1. If `new JFileChooser()` is used, the dialog window will open to some default place on the hard drive, and a lot of navigating may be necessary to get to the desired directory. For example, on a Windows machine, it may be directory `My Documents`. If you know the probable directory that will be navigated to, the use `new JFileChooser(s)` where `s` is a path to that directory.
2. The second statement sets the title to be used in the dialog window.
3. In the third statement, the call `jd.showOpenDialog(null)` causes the dialog window to appear. This call terminates only when the user clicks *Cancel* or *Open*. The method call returns one of three integers: (1) `JFileChooser.APPROVE_OPTION`, (2) `JFileChooser.CANCEL_OPTION` (the user hit the cancel button), and (3) `JFileChooser.ERROR_OPTION` (an error occurred or the dialog window was closed. If it was not the first one, return null, as per the spec of `getInputPath`.
4. About the fourth statement. Class `JFileChooser` was written before Java version 7 and therefore uses the older class `java.io.File` to describe a path on the hard drive. Function `jd.getSelectedFile()` returns the path that the user selected as a `File` object. The call on `toPath()` returns an equivalent path in a `Path` object.

```
/** Use a JFileChooser to obtain a Path to a file to be read.
 * If user cancels the dialog window, return null.
 * Parameter s can be a path on the hard drive or null.
 * If s is not* null, start the JFileChooser at the path given by s. */
public static Path getInputPath(String s) {
    JFileChooser jd= s == null ? new JFileChooser() : new JFileChooser(s);

    jd.setDialogTitle("Choose input file");
    int returnVal= jd.showOpenDialog(null);

    if (returnVal != JFileChooser.APPROVE_OPTION) return null;

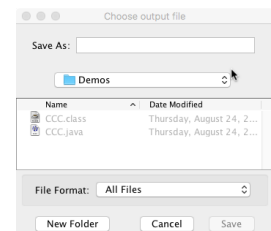
    return jd.getSelectedFile().toPath();
}
```

Class `JFileChooser` has some more useful features. For example, you can set it up so that the file to be read has to be a `.java` file, or a `.txt` file. Visit the API documentation for `JFileChooser` to learn more.

Getting a Path to an output file

To the right is an instance of a dialog window used to retrieve a path to a file to be written, i.e. an output file. Note the title: *Choose output file*. The user types in the name of the file to be written in the *Save as* field, navigates to the directory into which it is to be written, and clicks *Save*. Note that the user can create new folders (directories) if necessary.

The method to open the dialog window and then create and return the `Path` is quite similar to the method shown above. The main difference is that instead of calling `showOpenDialog` method `showSaveDialog` is called. It appears on the next page.



Getting a path using a JFileChooser

```
/** Use a JFileChooser to obtain a Path to a file to be written.
 * If the user cancels the dialog window (and thus does not give a
 * file name), return null.
 * Parameter s can be a path on the hard drive or null. If s is not
 * null, start the JFileChooser at the path given by s.*/
public static Path getOutputPath(String s) {
    JFileChooser jd= s == null ? new JFileChooser() : new JFileChooser(s);

    jd.setDialogTitle("Choose output file");
    int returnVal= jd.showSaveDialog(null);

    if (returnVal != JFileChooser.APPROVE_OPTION) return null;

    return jd.getSelectedFile().toPath();
}
```