package edu.duke;

import java.io.File;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.SwingUtilities;

import javax.swing.filechooser.FileFilter;

/\*\*

\* This utility class creates a thread safe file dialog box for loading and

\* saving files.

\*/

class FileSelector {

// result of selection

private static File[] ourFiles;

// BUGBUG: I think this is the right behavior, remembers where user left it last

private static JFileChooser ourChooser = new JFileChooser();

static {

ourChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

ourChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));

}

/\*\*

\* Pops up a dialog box to select only one file.

\*

\* @return

\*/

public static File selectFile () {

// guaranteed to have one element, though it may be null

return selectFiles(null, false, true)[0];

}

/\*\*

\* Pops up a dialog box to select only one file with given extensions.

\*/

public static File selectFile (String[] extensionAccepted) {

// guaranteed to have one element, though it may be null

return selectFiles(extensionAccepted, false, true)[0];

}

/\*\*

\* Pops up a dialog box to select multiple files.

\*/

public static File[] selectFiles () {

return selectFiles(null, true, true);

}

/\*\*

\* Pops up a dialog box to select multiple files with given extensions.

\* @return

\*/

public static File[] selectFiles (String[] extensionAccepted) {

return selectFiles(extensionAccepted, true, true);

}

/\*\*

\* Pops up a dialog box to save file with any extension.

\*/

public static File saveFile () {

// guaranteed to have one element, though it may be null

return selectFiles(null, false, false)[0];

}

/\*\*

\* Pops up a dialog box to save file with given extensions.

\*/

public static File saveFile (String[] extensionAccepted) {

// guaranteed to have one element, though it may be null

return selectFiles(extensionAccepted, false, false)[0];

}

// BUGBUG: one general function, but lots of booleans :(

private static File[] selectFiles (String[] extensionAccepted,

final boolean allowMultiple,

final boolean openForRead) {

ourChooser.setMultiSelectionEnabled(allowMultiple);

ourChooser.setFileFilter(new ChooserFilter(extensionAccepted));

try {

ourFiles = null;

SwingUtilities.invokeAndWait(new Runnable() {

@Override

public void run () {

int result = 0;

if (openForRead) {

result = ourChooser.showOpenDialog(null);

}

else {

result = ourChooser.showSaveDialog(null);

}

if (result == JFileChooser.CANCEL\_OPTION) {

ourFiles = new File[] { null };

} else {

try {

if (allowMultiple) {

ourFiles = ourChooser.getSelectedFiles();

} else {

ourFiles = new File[] { ourChooser.getSelectedFile() };

}

} catch (Exception e) {

JOptionPane.showMessageDialog(null, e.toString());

}

}

}

});

return ourFiles;

} catch (Exception e) {

// it is still an exception, just not one required to be handled

throw new RuntimeException(e);

}

}

// This class implements a filter for image file names.

static class ChooserFilter extends FileFilter {

private String myExtensions;

public ChooserFilter (String[] extensionsAccepted) {

if (extensionsAccepted != null) {

myExtensions = String.format("(?i).\*\\.(%s)", String.join("|", extensionsAccepted));

}

}

@Override

public boolean accept (File f) {

if (myExtensions != null) {

return f.getName().matches(myExtensions) || f.isDirectory();

} else {

return true;

}

}

@Override

public String getDescription () {

// BUGBUG: useful?

return "Files";

}

}

}