Contents

[HelpingClass.Java getStringInput(String title,int width,int height):String 2](#_Toc131346395)

[getSelecteFolderPathFromDirectory() :String 2](#_Toc131346396)

[Reson: 2](#_Toc131346397)

[getFileNameInput(String title,String Message): String 2](#_Toc131346398)

[Reason 3](#_Toc131346399)

[XMLAcceccer File.Java: 3](#_Toc131346400)

[LoadNewFile() :String 3](#_Toc131346401)

[Reason: 3](#_Toc131346402)

[saveNewFile(String file data,String fileName) : void 3](#_Toc131346403)

[Reason: 3](#_Toc131346404)

[XMLFIleSelector: 3](#_Toc131346405)

[selectXmlFile(): String 3](#_Toc131346406)

[Reason: 4](#_Toc131346407)

[checkIfPresent(String filename):boolean 4](#_Toc131346408)

[Reason: 4](#_Toc131346409)

[CopyInCurrentDirectory(String sourceFilePath,String fileName): boolean 4](#_Toc131346410)

[Reason: 5](#_Toc131346411)

## HelpingClass.Java getStringInput(String title,int width,int height):String

This function is creating a popup dialog window using Java Swing components that allow the user to enter a string input. The window includes a JTextArea where the user can enter text and two buttons, "Save" and "Cancel".

The "Save" button saves the user's input and closes the dialog window while the "Cancel" button simply closes the dialog window without saving any input.

The function takes in three arguments: a title for the dialog window, width, and height.

After the user enters text and clicks "Save", the function returns the user's input as a String. If the user does not enter any text and clicks "Save", the function returns null.  
Reason   
This is the get the XML from the new slide

## getSelecteFolderPathFromDirectory() :String

This function creates a file chooser dialog using Java Swing components that allows the user to select a folder from the directory.

The function sets the default directory to the current directory and sets the dialog title to "Select a folder". It also restricts file selection to only directories and disables the option to accept all file filters.

After the user selects a folder and clicks "Open", the function returns the absolute path of the selected folder as a String. If the user cancels the selection or closes the dialog window without selecting any folder, the function returns null.

This function is useful for applications that require the user to select a folder for input or output purposes, such as file management systems or data processing applications.

### Reson:

This allows users to select the folder in the local disk

## getFileNameInput(String title,String Message): String

This function creates an input dialog using Java Swing components that allows the user to enter a file name as a string input. The function takes two arguments: a title for the input dialog window and a message to display to the user. The title appears in the dialog window's title bar, while the message appears as a label in the input dialog window. After the user enters a file name and clicks "OK", the function returns the entered file name as a String. If the user cancels the input dialog or leaves the input field empty, the function returns null. This function is useful for applications that require the user to input a file name for saving, opening, or processing files, such as text editors or image-processing software.

### Reason

This is for the online text In this context it is used to get the name of the file

# XMLAcceccer File.Java:

## LoadNewFile() :String

This function loads new XML data from the user, prompts the user to enter a file name to save the data as an XML file, saves the data to the specified file, and returns the name of the created file as a String. It uses other functions from a "HelpingClass" to prompt the user for input and save the data to a file. If the user cancels any input dialogs or does not enter data, the function returns null. This function is useful for applications that require loading and saving XML data, such as data processing or visualization application.

### Reason:

This is for the Load new file.

## saveNewFile(String file data,String fileName) : void

This function saves the provided file data to a file with the specified file name. It takes in two parameters: fileData as a String, and fileName as a String representing the name of the file to be created.

The function creates a new File object with the specified file name, then creates a FileOutputStream to write the file data to the new file. It then converts the file data String to a byte array and writes it to the output stream using the write() method.

After the file data has been written, the function closes the output stream and prints a message indicating whether the file was created successfully or not.

This function is useful for applications that need to save data to a file, such as data processing or file management applications.

### Reason:

This function allows the user to save the new file.

# XMLFIleSelector:

## selectXmlFile(): String

This function displays a file chooser dialog and allows the user to select an XML file. If the user selects a file and clicks "OK", the function returns the name of the selected file as a String.

The function first creates a JFileChooser object and displays it using the showOpenDialog() method. If the user selects a file and clicks "OK", the function retrieves the absolute path of the selected file using the getAbsolutePath() method and checks if the file is already present in the current directory using a checkIfPresent() method.

If the selected file is not already present in the current directory, the function calls a CopyInCurrentDirectory() method to copy the file to the current directory.

Finally, the function returns the name of the selected file as a String. If the user cancels the file chooser dialog, the function returns null.

This function is useful for applications that require the user to select an XML file, such as data processing or visualization applications.

Reason:  
this allows users to select the XML file from the local disk that they want to show in Jabberpoint.

## checkIfPresent(String filename):boolean

This function checks whether a file with a given filename already exists in the current working directory.

First, the function gets the current working directory using the method. It then creates a file path by joining the filename to the current working directory using the method to ensure platform-independent file paths.

Next, the function checks if the file exists using the method. If the file exists, it displays a message to the user using a dialog and returns. Otherwise, it simply returns.

Overall, this function can be useful in preventing the user from accidentally overwriting an existing file when saving data.

Reason:  
For check, the XML that they select to show is not present in the current directory on which they are working.

## CopyInCurrentDirectory(String sourceFilePath,String fileName): boolean

This function copies a file from a source location to the current directory, with a specified file name. It takes in the source file path and the desired file name as input parameters.

The function first creates a File object for the source file and another for the destination file with the given name. Then, it creates input and output streams to read from and write to the files, respectively. It reads the data from the input stream in chunks of 1024 bytes and writes it to the output stream until there is no more data to read.

After successfully copying the file, the function returns true. If an exception occurs during the copying process, the function catches the exception and returns false.

This function can be used in applications that need to copy files from one location to another, such as file managers or backup utilities.

Reason:  
This is for copying the XML file to the current directory if they want to open the XML then that XML is first copied into the current directory then Java load it otherwise it will show an error