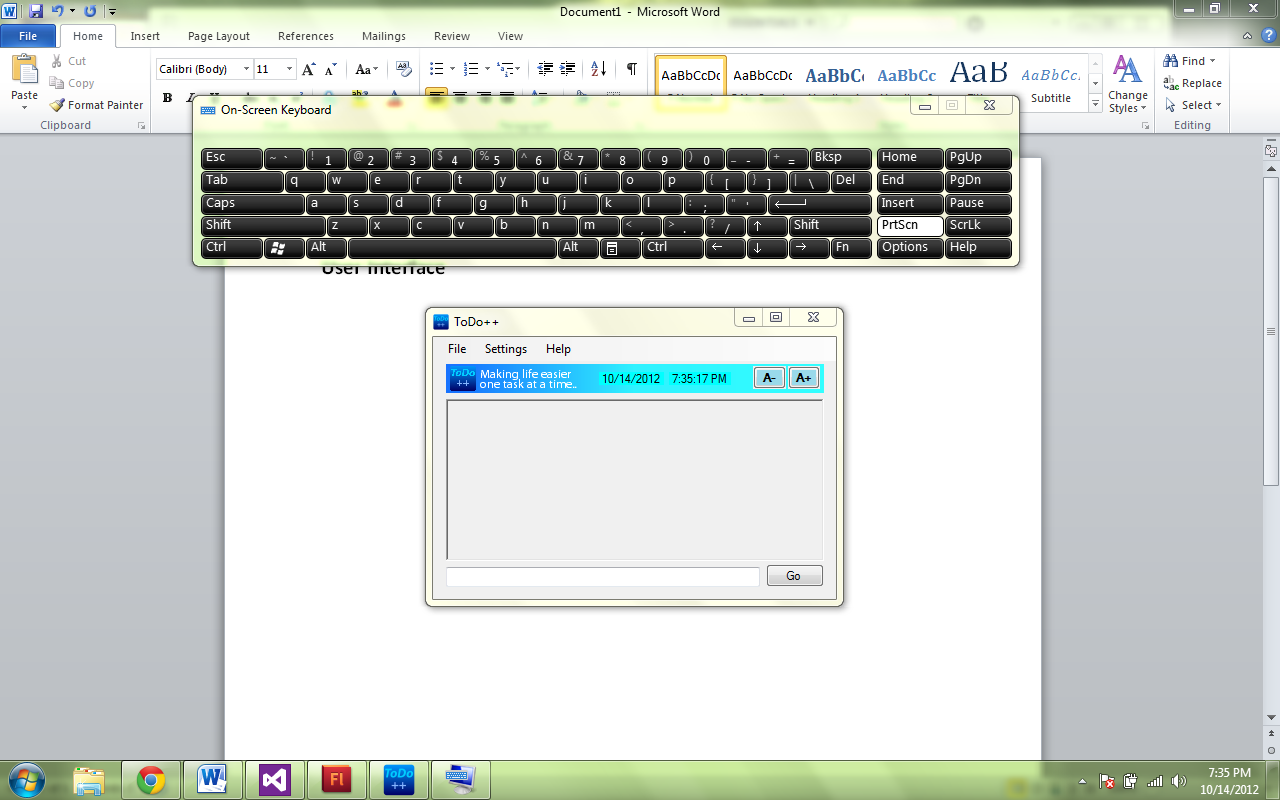
**Main User Interface**

Developer Guide for User Interface

**Controls/Interfaces**



**textInput**

Place where user inputs commands

**goButton**

ToDo++ executes the command. Enter may be hit too

**TimeDateControl** (TimeDateControl.cs)

Displays Date and Time

**increaseSizeButton**

Increase Size of Text

**decreaseSizeButton**

Decrease Size of Text

**OutputBox Control** (Menu.cs)

Requires an instance Settings to be passed in

**Menu Control** (Menu.cs)

Requires an instance of UI and Settings to be passed in

**Overview**

The User interface class is what the User see’s when the program is open. It is where all the magic begins. In order to function, this class contains an instance of Settings (see below) and Logic (see below). Settings contains all program settings that are loaded when ToDo++ is opened. Logic is where user input is passed in and processed. Listed below is a class diagram showing the relations between the objects.

**Main User Interface (UI.cs)**

What the User sees when opening ToDo++

Attributes

|  |  |
| --- | --- |
| Variables | Explanation |
| private Hotkeys.GlobalHotkey ghk; | Global Hotkey to Minimize to System Tray |
| Settings settings; | Settings stores all settings data, including Flexi-Commands |
| Logic logic; | Instance of Logic that handles Data structure and File Operations |

Constructor UI()

|  |  |
| --- | --- |
| Methods | Explanation |
| InitializeSystemTray(); | Loads Code to place App in System Tray |
| InitializeSettings (); | Loads initial Settings of App and applies the settings |
| InitializeMenu(); | Loads the menu strip |
| InitializeOutputBox(); | Loads Output Box |
| InitializeLogic(); | Creates instance of Logic to be used by Text Processing |

Win32Functions

|  |  |
| --- | --- |
| Methods | Explanation |
| private void InitializeSystemTray() | Registers the Hotkey (Alt+Q) to be used for Minimizing/Maximizing to/from System Tray |
| private void MinimiseMaximiseTray() | Minimize/Maximizes App from System Tray |
| private void RegisterInStartup(bool isChecked) | Sets whether ToDo++ is to be open at startup or not |

PrepareFunctions

|  |  |
| --- | --- |
| Settings | Explanation |
| private void Initializeettings () | Creates an Instance of Settings |
| private void MinimiseToTrayWhenChecked() | Minimizes App to System tray if true |
| private void RegisterLoadOnStartupWhenChecked() | Sets the Load on Startup Status |
| Menu | Explanation |
| private void InitializeMenu() | Prepare the Menu Bar. Pass an instance of settings into it so it can interact with it. Also loads the settings into menu |
| OutputBox | Explanation |
| private void InitializeOutputBox() | Prepare the Output Box. Pass an instance of settings into it so it can interact with it. Also load the settings into outputBox |
| Logic | Explanation |
| private void InitializeLogic() | Creates an instance of Logic |

TextInput/KeyboardCommands

|  |  |
| --- | --- |
| Methods | Explanation |
| private void ProcessText() | Passes an the user text to Logic, which processes it and returns an output to be displayed |
| protected override bool ProcessCmdKey(ref Message msg, Keys keyData) | New Keyboard shortcuts may be entered here. There is no need to use this function directly. Simply place all Keyboard shortcut keys here. Do not use ALT+Q as that is used for minimizing ToDo++ to the system tray. |

**Menu Control (Menu.cs)**

Menu Bar

|  |  |
| --- | --- |
| Methods | Explanation |
| public void SetSettings (Settings passedSettings) | Set the settingsof the Menu Control, so interaction is possible |
| public void LoadSettingsIntoMenu() | Currently loads the check status of Load On Startup and Get Minimized Status |

**OutputBox Control (Menu.cs)**

Output Box that shows User and ToDo++ commands

|  |  |
| --- | --- |
| Methods | Explanation |
| public void SetSettings(Settings passedSettings) | Set the settings of the OuputBox Control, so interaction is possible |
| public void SetOutputSize(int size) | Sets the Text Size of the OutputBox directly |
| public void DecreaseSizeOfOutput() | Decrease the Text Size by 1 unit, while modifying the settings |
| public void IncreaseSizeOfOutput() | Increase the Text Size by 1 unit, while modifying the settings |
| public void DisplayCommand(string userInput,string systemOutput) | This method displays both ToDo++ output and input with correct formatting |

**Settings User Interface**

Developer Guide for User Interface

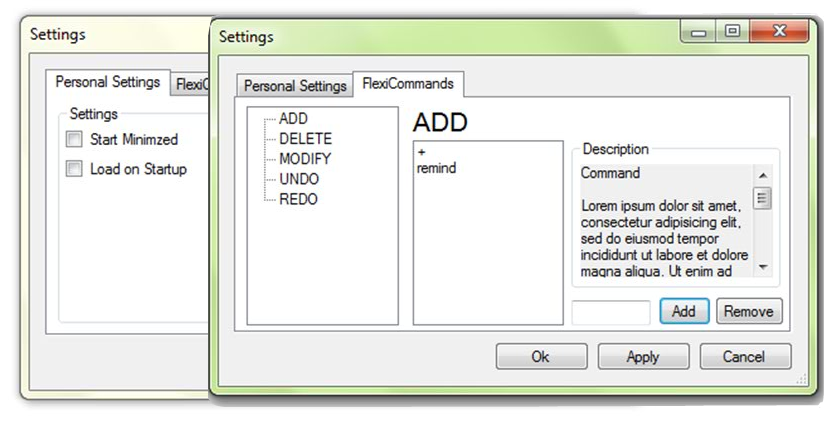
**Controls/Interfaces**

**commandPreview**

TextBox that shows the current command

**listOfCommands**

Lists all user Commands



**commandDescription**

Description of selected Command

**addUserCommandButton**

**removeUserCommandButton**

Add remove items from/to the list

**okayButton**

**applyButton**

**cancelButton**

**CommandTree**

This is where all Commands that are modifiable are added

**minimisedCheckbox**

**loadOnStartupCheckbox**

**Settings Control (SettingsUI.cs)**

Settings of ToDo++ are shown

Attributes

|  |  |
| --- | --- |
| Attribute | Explanation |
| private Settings settings; | Contains the instance of settings passed in |
| private CommandType currentCommand; | Name of Command that is currently selected |
| private Settings tempSettings; | Copy of settings. This is where FlexiCommands that are added stored and displayed in the SettingsUI. When Apply or Ok are hit, settings will be updated with commands from tempSettings |

Methods

|  |  |
| --- | --- |
| Methods | Explanation |
| public void LoadPersonalSettingsTab(); | Loads the Personal Settings Tab |
| public void LoadFlexiCommandTab(); | Loads the Flexi Commands Tab |

**Settings (Settings.cs)**

All Settings are managed and stored here

SettingsList is the class where all settings are stored. Settings contains an instance of SettingsList. Settings also has methods that can modify SettingsList safely. It consists of the following attributes/settings, and more settings can be added later on if needed be

public bool loadOnStartup;

public bool startMinimized;

public int textSize;

public List<string> customKeywords\_ADD;

public List<string> customKeywords\_DELETE;

public List<string> customKeywords\_MODIFY;

public List<string> customKeywords\_UNDO;

public List<string> customKeywords\_REDO;

Methods

|  |  |
| --- | --- |
| TextSize | Explanation |
| public void SetTextSize(int size)  public int GetTextSize() | Set Text size of OutputBox  Gets Text size of OutputBox |
| public void IncreaseTextSize()  public void DecreaseTextSize() | Methods to increase/decrease Text Size |
| CommandModifications | Explanation |
| public void AddCommand(string newCommand, CommandType commandType) | Add a new FlexiCommand. Requries the command and CommandType |
| public void RemoveCommand(string commandToRemove, CommandType commandType) | Remove the specified FlexiCommand |
| public List<string> GetCommandList(CommandType commandType) | Returns a list of all added/available user commands |
| public void AddCommandsToStringParser() | Pushes new set of FlexiCommands into the StringParser |
| public void OpenFile()  public void WriteToFile() | Open and Write to File the SettingsList |
| Clone/Copy Operations |  |
| public void CopyUpdatedCommandsFrom(Settings passedSettings) | Copies just the new commands from a copy of Settings |
| public Settings CloneObj() | Clones an instance of Settings (Deep Copy) |