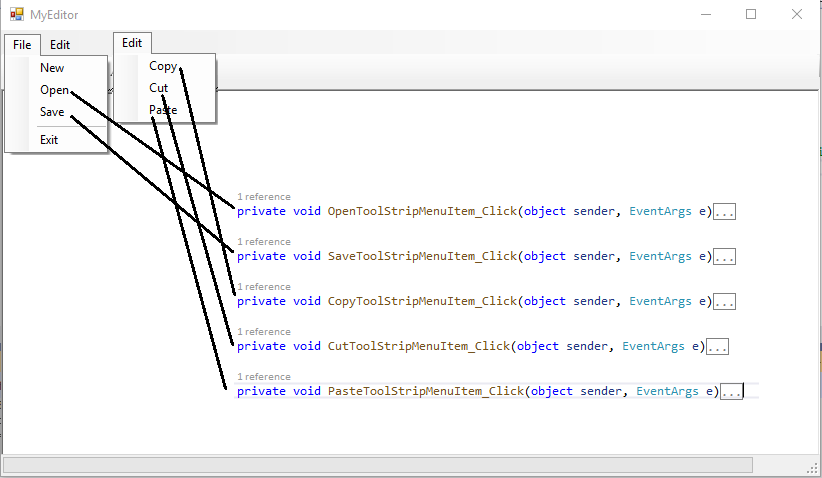
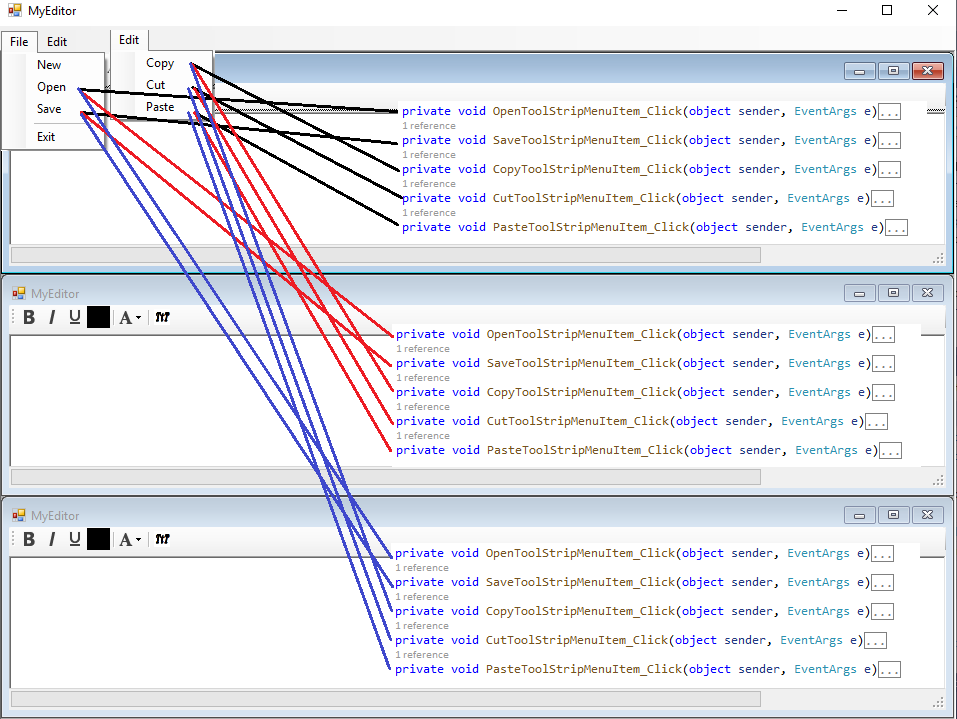
**Single Document Interface (SDI)**

With the Single Document Interface, one form is active at a time and the menu and toolbar items link directly to the event handlers belonging to that form. There is a one-to-one relationship between the controls and the event handlers. Any new forms that are created are not restricted to the "parent" form region.



**Multiple Document Interface (MDI)**

With the Multiple Document Interface, each child form is running as a separate application, but we want to link the menu options in the parent container to the event handlers in every child.



The child form constructor must receive a reference to the parent form to set MdiParent and to link to the parent event listeners:

public Form1( MyEditor myEditor)

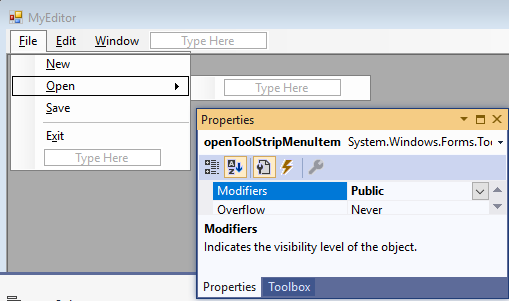
{

InitializeComponent();

// set the MdiParent of this form to the parent form so that MDI features are enabled

this.MdiParent = myEditor;

The parent menu item controls need to be made "public" via the "Modifiers" property:



So that when each child form is created, the constructor can add the OpenToolStripMenuItem\_Click event handler of the new form to the parent's menu item control:

public Form1( MyEditor myEditor)

{

InitializeComponent();

// set the MdiParent of this form to the parent form so that MDI features are enabled

this.MdiParent = myEditor;

// add our local form's event handlers to the parent form controls

myEditor.openToolStripMenuItem.Click += new System.EventHandler(this.OpenToolStripMenuItem\_Click);

myEditor.saveToolStripMenuItem.Click += new System.EventHandler(this.SaveToolStripMenuItem\_Click);

myEditor.copyToolStripMenuItem.Click += new System.EventHandler(this.CopyToolStripMenuItem\_Click);

myEditor.cutToolStripMenuItem.Click += new System.EventHandler(this.CutToolStripMenuItem\_Click);

myEditor.pasteToolStripMenuItem.Click += new System.EventHandler(this.PasteToolStripMenuItem\_Click);

// when this form closes, we need to remove the local event handlers from the parent's event listeners,

// because when this form closes, those handlers will no longer exist

this.FormClosing += new FormClosingEventHandler(this.Form1\_\_FormClosing);

When "Open" is selected, the OpenToolStripMenuItem\_Click event will trigger in every child form, so an additional check must be added to each event handler to ensure that the event is handled only by the active child form:

public void OpenToolStripMenuItem\_Click(object sender, EventArgs e)

{

// the MdiParent form keeps track of which child form is currently active

if( this.MdiParent.ActiveMdiChild != this )

{

return;

}

When the child form is closed, Form1\_\_FormClosing() is called which should remove the child event handlers from the parent event listeners:

private void Form1\_\_FormClosing( object sender, EventArgs e)

{

MyEditor myEditor = (MyEditor)this.MdiParent;

myEditor.openToolStripMenuItem.Click -= this.OpenToolStripMenuItem\_Click;

myEditor.saveToolStripMenuItem.Click -= this.SaveToolStripMenuItem\_Click;

myEditor.copyToolStripMenuItem.Click -= this.CopyToolStripMenuItem\_Click;

myEditor.cutToolStripMenuItem.Click -= this.CutToolStripMenuItem\_Click;

myEditor.pasteToolStripMenuItem.Click -= this.PasteToolStripMenuItem\_Click;

}