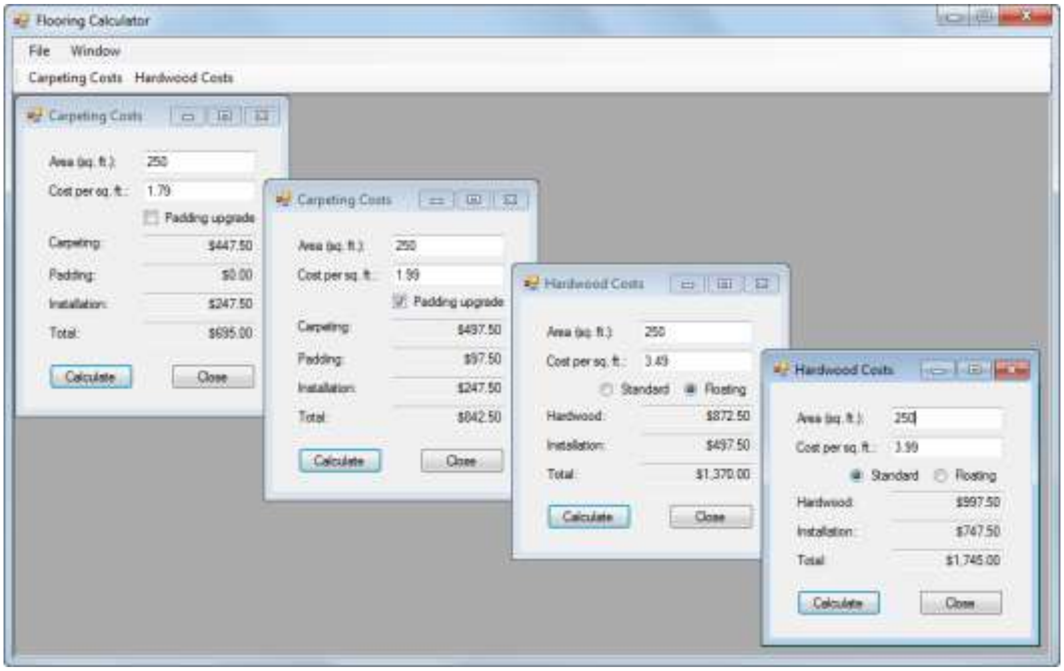
Create an MDI application

In this exercise, you will create an application with a multiple-document interface that consists of a parent form and two child forms.



Area (sq. ft.): 250
Cost per sq. ft.: 1.79
□ Padding upgrade
Carpeting: $447.50
Padding: $0.00
Installation: $247.50
Total: $695.00

Area (sq. ft.): 250
Cost per sq. ft.: 1.99
☑ Padding upgrade
Carpeting: $497.50
Padding: $97.50
Installation: $247.50
Total: $842.50

Area (sq. ft.): 250
Cost per sq. ft.: 3.49
○ Standard ● Floating
Hardwood: $872.50
Installation: $497.50
Total: $1,370.00

Area (sq. ft.): 250
Cost per sq. ft.: 3.99
● Standard ○ Floating
Hardwood: $997.50
Installation: $747.50
Total: $1,745.00

Open the project and add a parent form

1. Open the CalculateFlooringCosts project that is attached. Then, review the code for the Carpeting Costs and Hardwood Costs forms so you understand how they work.
2. Add a form to the project, and set the IsMdiContainer property of this form to true to identify it as the parent form. Resize the form so it's large enough to hold several child forms, and modify the program.cs file so this form is displayed when the application starts.

Add the File menu to the parent form

3. Add a MenuStrip control to the parent form. Then, use the Menu Designer to add a File menu to the parent form. This menu should include four menu items that display a Carpeting Costs form, display a Hardwood Costs form, close the active child form, and exit from the application. Give each menu item an appropriate name, include access keys for the menu and menu items, and include a separator bar between the Close and Exit items.
4. Add an event handler for the Click event of each item in the File menu. The event handlers for the New Carpeting Costs and New Hardwood Costs items should create a new instance of the appropriate form, set the parent form to the current form, and display the form. The event handler for the Close item should close the active child form, if there is one. And the event handler for the Exit item should exit from the application.
5. Test the application to be sure that the items in the File menu work as expected.

Add the Window menu to the parent form

6. Add a Window menu to the right of the File menu. Then, add three items to this menu that will let the user arrange the forms in a cascaded, vertical, or horizontal layout.
7. Give each item in the Window menu an appropriate name, and then add an event handler for the Click event of each item that arranges the child forms appropriately.
8. Set the MdiWindowListItem property of the menu strip so the Window menu will display a list of the open child forms.
9. Test the application to be sure that the items in the Window menu work as expected.

Add a toolbar to the parent form

10. Add a ToolStrip control to the parent form. Then, add two buttons to this toolbar with the text “Carpeting Costs” and “Hardwood Costs” on them. Be sure to assign appropriate names to these buttons.
11. Add an event handler for the Click event of each button that uses the Click event handler for the associated menu to display the form.
12. Test the application to be sure that the toolbar buttons work.

Add tooltips and context-sensitive help to the child forms

13. Add a ToolTip control to each child form. Then, add a tooltip for each text box, check box, and radio button on these forms. For the check box and radio buttons, you should display the cost associated with selecting that control.
14. Add a HelpProvider control to each child form. Then, add context-sensitive help for the Area text box on each form that explains how to calculate the area (length in feet x width in feet).
15. Add context-sensitive help for each form that describes what the form does.
16. Test the application to be sure that the tooltips and context-sensitive help are displayed.