





Figure 4. A windows class hierarchy

The types of windows are:

Window – is allowed to draw itself on the Windows desktop.

Composite Window – contains other windows within its boundary.

Framed Window – can pop-up independently. It has a border that the user can click on with the mouse to move and resize.

Label – displays text to the user.

Textbox – allows the user to read and input text.

Button – the user presses the button to cause some action to occur.

CheckBox – allows the user to control a Boolean value.

OptionSelect – allows the user to select one option from a list.

Listbox – allows the user to select items from a list.

Canvas – allows drawing graphical shapes in a window.

Form – allows grouping a set of controls so that they can be positioned as one.

MenuBar – allows the user to select an item from a hierarchy of popup menus.

ToolBar – contains an array of buttons.

RadioButton – it looks similar to a group of checkboxes, but only one can be selected.

MainWindow – contains all the other Windows of the application.

DialogBox – a pop-up window for the user to input or see information.

FileDialog – a specialized pop-up window that allows the user to select a file.

When a windows program runs, all window instances are maintained in a composition hierarchy under a single window instance, called the “main window”. Every window except the main window has a “parent” window, which “owns” it. All windows except pop-up windows sit inside the borders of their parent. The window hierarchy allows for message passing, so that a function in a parent window can respond to an event in a child window. The hierarchy also allows for all windows to be destroyed when the main window is closed.

The term “control” is often used in place of “window,” since the user is able to control some operation or information in the program. In X-Windows, the term “Widget” is also used.

A typical object-oriented windows main function looks like:

```
int main()
{
    MainWindow *main_window = new MainWindow;
    MenuBar *menu = new MenuBar;
    main_window->Add(menu);
    // build up windows
    ...
    main_window -> Run();
    return 0;
}
```

In this program, the main window object is instantiated, and child windows are created with the main window as parent. These windows may then become parents to other windows. Once window objects have instantiated, positioning commands may have to be used to specify the positions of windows with respect to their siblings and their parent. Often this positioning is automatic, as with menu buttons. As well, event handlers must be added to windows to handle events coming from these windows. Finally the main window is popped up with its child windows and the main event handling loop is started through the Run() function. Typically the main event handling loop looks as follows:

```
void EventHandlingLoop()
{
    Event evt;
    Window win;
    for (;;)
    {
        WaitForEvent(&win, &evt);
        HandleEvent(win, evt);
    }
}
```

The `HandleEvent()` function knows which event handlers have been associated with each window and calls them as required.

The basic windows start up process may therefore be summarized as:

1. Create main window object,
2. Create child window objects,
3. Position child windows, if required,
4. Specify event handlers,
5. Show the main window and start the event handling loop.

Although the display of the hierarchy of windows is the most immediate striking aspect of windows programs, their true magic comes from the event handling.