**WINDOW CLASS**

Windows are created based on predefined templates called "window classes."

A window class encapsulates the attributes and behaviors common to a group of windows.

It specifies the default appearance, layout, and message handling for its associated windows.

Think of a window class like ablueprint or a set of instructions for the computer on how to create a particular type of window.

It's like defining the style and behavior that a group of windows will share.

This way, when you want to create a new window with similar characteristics, you can refer to this blueprint, and the computer knows how to make it.



Imagine you're designing a house, and you have a blueprint that specifies certain features like the number of rooms, the size of windows, and the color of walls.

In this analogy, the blueprint is similar to a window class. It provides a template for creating windows with specific attributes and behaviors.

So, a window class is like a template that helps the computer know how to build and handle windows of a particular type. It's a way to organize and reuse characteristics for similar windows in a more efficient manner.

**Key Features of a Window Class:**

* Defines the default appearance of windows, such as the size, position, color, and font.
* Specifies the layout of the window, including the placement of controls and elements.
* Establishes the message handling rules for the window, determining how it responds to user input and system events.

**Example:**

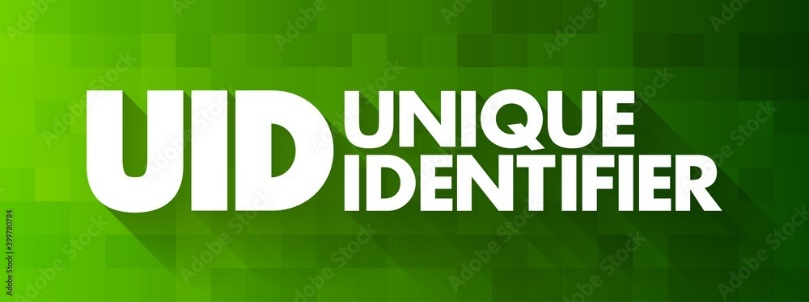
The "CWnd" window class in Microsoft Word defines the attributes and behaviors of the main application window. This class specifies the window's size, position, title bar, menu bar, toolbar, and main editing area.

When the user interacts with the window, such as clicking a menu item or pressing a toolbar button, the window's associated procedure receives messages corresponding to those actions. The procedure then interprets these messages and triggers the appropriate actions, such as opening menus or executing commands.

WINDOWS CLASS NAMES

A window class name is a unique identifier that distinguishes one window class from another.

It serves as a label for a particular type of window, allowing the operating system to associate the window class's attributes and behaviors with the newly created window.



**Key Characteristics of a Window Class Name:**

* Uniquely identifies a window class for the operating system.
* Acts as a reference to the window class's attributes and behaviors.
* Enables the creation of windows with consistent appearance and behavior based on the class definition.

**Example:**

The "CWnd" window class name is used to create the main application window in Microsoft Word. When the developer specifies the "CWnd" window class name during window creation, the operating system understands which set of attributes and behaviors to apply to the newly created window. This ensures that the window has the same appearance and behavior as other windows based on the "CWnd" class behave.

**WINDOW PROCEDURES**

Each window has an associated "window procedure," which is a function responsible for handling messages directed to that window.

A **window procedure** is a function that handles messages sent to a window.

It is responsible for interpreting and responding to these messages, which can range from user actions like clicking buttons to system events like window size changes.

It performs the necessary actions, and returns control to the operating system

The window procedure is the heart of a window's functionality, determining its responsiveness and behavior.



**CREATING A WINDOW**

To create a window, you need to register a window class first.

This involves defining the window class name and providing information about the window's attributes and behaviors using a WNDCLASS structure.

Once the window class is registered, you can create individual windows of that class using the CreateWindow function.

The CreateWindow function takes the window class name as one of its arguments.

**Simpler explanation:**

Imagine creating a window like putting together a puzzle.

First, you need to tell the computer what kind of window you want by registering a window class.

This involves giving it a name and describing its characteristics.

Once registered, you can use that class to create individual windows using a function called CreateWindow.

It's like telling the computer, "Hey, make me a window based on this class I've described."

**Relationship between Window Class Names and Procedures**

A window class is associated with a specific window procedure. When a window is created, the operating system associates the window class's procedure with the newly created window. This ensures that the window receives the appropriate message handling for its class.

By creating different window classes with varying attributes and procedures, you can create a wide range of windows tailored to specific needs.

**Example of Window Class Names and Procedures**

Consider a typical Windows application with a menu bar, a toolbar, and a main display area. Each of these elements could be represented by a separate window class.



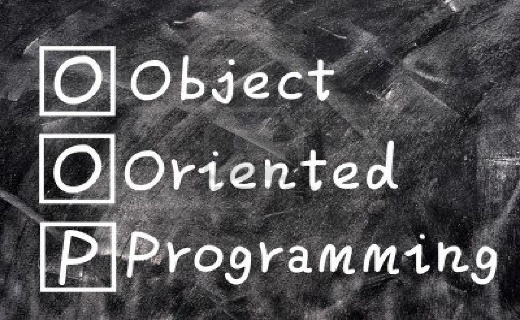
The menu bar window class would define the attributes and behaviors of a menu bar, such as its layout, font, and interaction with user input.

Similarly, the toolbar window class would define the attributes and behaviors of a toolbar, including its icons, tooltips, and behavior when clicked.

The main display area window class would define the attributes and behaviors of the main window, such as its size, background color, and ability to display graphics.

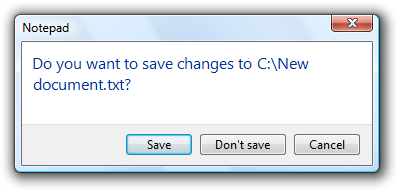
**Windows as Objects**

Windows programming is deeply rooted in object-oriented programming principles. The central object in this context is the "window," which represents a rectangular area on the screen that facilitates user interaction and displays graphical output. Windows can be categorized into two primary types:



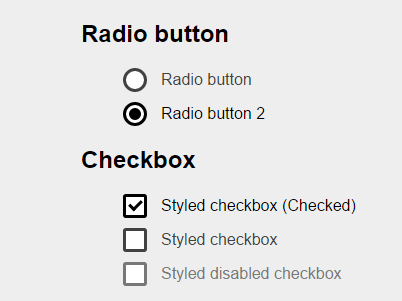
Application Windows: These are the main windows of applications, typically featuring a title bar, menu bar, toolbar, and main display area.

Dialog Boxes: These are specialized windows that pop up to provide additional information, request user input, or display messages. They may or may not have a title bar.



**Child Windows and Control Windows**: Embedded within these primary windows are smaller, interactive elements like push buttons, radio buttons, check boxes, list boxes, scroll bars, and text-entry fields.

These elements are collectively known as "child windows" or "control windows." They are considered child objects of their parent window and inherit its properties and behaviors.



**Message-Driven Communication**

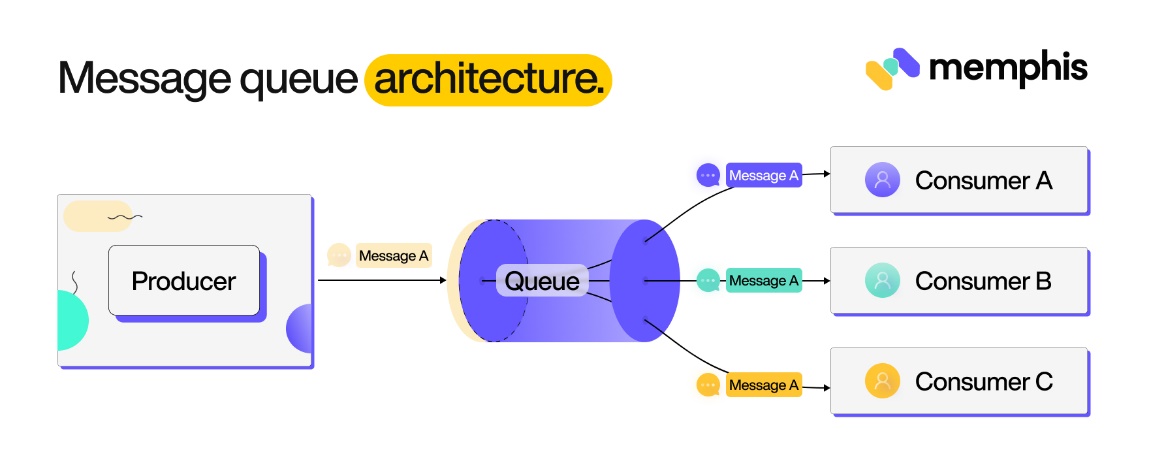
Windows programs communicate with the operating system and with each other through a mechanism called "messages."

These messages are structured data packets that convey information about user actions, system events, or requests for services.

**Message Queue and Message Loop**

When a Windows program starts, the operating system creates a "message queue" for that program. This queue stores incoming messages destined for the program's windows.

The program's main loop continuously retrieves messages from the queue and dispatches them to the appropriate window procedures for processing.



**The Architecture in Action**

To illustrate how these concepts work together, consider a simple scenario where a user resizes an application window:



* The user drags the window's border, sending a resize message to the operating system.
* The operating system directs the resize message to the program's message queue.
* The application's message loop retrieves the resize message from the queue.
* The message loop identifies the relevant window and its associated window procedure.
* The message loop calls the window procedure, passing the resize message as an argument.
* The window procedure processes the resize message, adjusting the window's layout and content accordingly.
* Control returns to the message loop, which continues processing other messages.

**Window Class Names**

Definition: A window class name is a unique identifier that defines a set of attributes and behaviors for a particular type of window. It serves as a template for creating windows with similar characteristics.

Example: In Microsoft Word, the window class name for the main application window is "CWnd." This window class defines the basic appearance and behavior of the main window, including its title bar, menu bar, toolbar, and main editing area.

Purpose: Window class names allow developers to create standardized windows with consistent attributes and behaviors. This simplifies development and ensures a cohesive user experience across different applications.

**Window Procedures**

Definition: A window procedure is a function that handles messages sent to a window. It is responsible for interpreting and responding to these messages, which can range from user actions like clicking buttons to system events like window size changes.

Example: In Microsoft Word, the window procedure for the main application window is responsible for processing messages such as menu selections, toolbar button clicks, and keyboard key presses. It handles these messages by performing the appropriate actions, such as opening menus, executing commands, and updating the document.

Purpose: Window procedures provide a mechanism for windows to respond to user input and system events. They are the heart of a window's functionality, determining its responsiveness and behavior.

**Creating a Window**

Steps:

Register the Window Class: This involves defining the window class name and providing information about the window's attributes and behaviors using a WNDCLASS structure.

Create the Window: Once the window class is registered, you can create individual windows of that class using the CreateWindow function. The CreateWindow function takes the window class name as one of its arguments.

Example: In Microsoft Word, the main application window is created by registering the "CWnd" window class and then calling the CreateWindow function. This creates the window with the specified attributes and behaviors, such as its size, position, and title bar.

Relationship between Window Class Names and Procedures

Association: A window class is associated with a specific window procedure. When a window is created, the operating system associates the window class's procedure with the newly created window.

Ensuring Appropriate Message Handling: This ensures that the window receives the appropriate message handling for its class. For instance, a button window will receive messages related to button clicks, while a menu window will receive messages related to menu selections.

Significance of Window Class Names and Procedures

Standardization and Customization: Window class names and procedures provide a mechanism for standardizing the appearance and behavior of windows while allowing for customization when necessary.

Flexibility in Window Design: By creating different window classes with varying attributes and procedures, developers can create a wide range of windows tailored to specific needs.

Example of Window Class Names and Procedures

Microsoft Word:

Window Class: "CWnd" - defines the main application window

Window Procedure: Handles messages for the main window, such as menu selections, toolbar button clicks, and keyboard key presses

Other Window Classes: "CButton" for buttons, "CMenu" for menus, and "CEdit" for text-entry fields

Conclusion

Window class names and procedures are fundamental concepts in Windows programming, providing a structured approach to creating and managing windows with diverse functionalities. They enable developers to customize the appearance and behavior of windows, creating a rich and interactive user experience.