Buenas Prácticas

¿Por qué usar constantes?

El uso de constantes en el código fuente es de vital importancia para evitar que en el mismo aparezcan "Números mágicos" que luego, a medida que pasa el tiempo y el sistema crece, pierden significado y generan confusión para el desarrollador que debe modificar cierta parte del código, corregir un error, etc.

Veamos el siguiente caso:

```
if( factura.estado == 4 || factura.estado == 28 ){ ...
```

Al leer lo anterior a simple vista, es imposible saber que significa que una factura esté en estado 4 o 28 sin analizar en detalle el código y las instrucciones luego del If.

Si el fragmento anterior lo escribimos de la siguiente manera:

```
Definir ESTADO_REGISTRADA = 4
Definir ESTADO_FACTURADA = 28

if( factura.estado == ESTADO_REGISTRADA || factura.estado == ESTADO_FACTURADA ){ ...
```

Ahora, al leerlo, se entiende claramente que se está preguntado si el estado de la factura es registrada o facturada.

Última modificación: Thursday, 6 de May de 2021, 16:36

◄ Announcements | Ir a... | Modificadores de Acceso ►

Buenas Prácticas

Modificadores de Acceso

Todos los tipos y miembros de tipo tienen un nivel de accesibilidad. El nivel de accesibilidad controla si se pueden usar desde otro código del ensamblado u otros ensamblados. Use los modificadores de acceso siguientes para especificar la accesibilidad de un tipo o miembro cuando lo declare:

- public: Puede obtener acceso al tipo o miembro cualquier otro código del mismo ensamblado o de otro ensamblado que haga referencia a éste.
- private: solamente el código de la misma `class` o `struct` puede acceder al tipo o miembro.
- protected: solamente el código de la misma `class`, o bien de una `class` derivada de esa `class`, puede acceder al tipo o miembro.
- internal: Puede obtener acceso al tipo o miembro cualquier código del mismo ensamblado, pero no de un ensamblado distinto.
- protected internal: cualquier código del ensamblado en el que se ha declarado, o desde una `class` derivada de otro ensamblado, puede acceder al tipo o miembro.
- private protected: el código de la misma `class`, o de un tipo derivado de esa `class`, puede acceder al tipo o miembro solo dentro de su ensamblado de declaración.

Última modificación: Monday, 10 de May de 2021, 14:56

◄ ¿Por qué usar constantes?

Ir a...

Accesibilidad de clases, registros y estructuras ►

Buenas Prácticas

Accesibilidad de clases, registros y estructuras

Las clases, los registros y las estructuras que se declaran directamente en un espacio de nombres (es decir, que no están anidadas en otras clases o estructuras) pueden ser `public` o `internal`. Si no se especifica ningún modificador de acceso, el valor predeterminado es `internal`.

Los miembros de estructura, incluidas las clases y las estructuras anidadas, se pueden declarar como `public`, `internal` o `private`. Los miembros de clase, incluidas las clases y las estructuras anidadas, pueden ser `public`, `protected internal`, `protected`, `internal`, `private protected` o `private`. Los miembros de clase y de estructura, incluidas las clases y estructuras anidadas, tienen acceso `private` de forma predeterminada. Los tipos anidados privados no son accesibles desde fuera del tipo contenedor.

Última modificación: Monday, 10 de May de 2021, 14:56

◄ Modificadores de Acceso

Ir a...