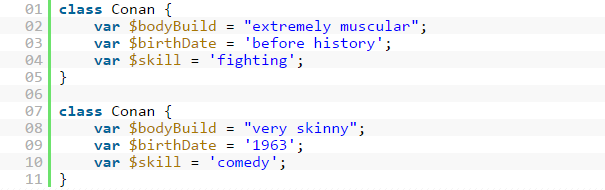
***NameSpace:***

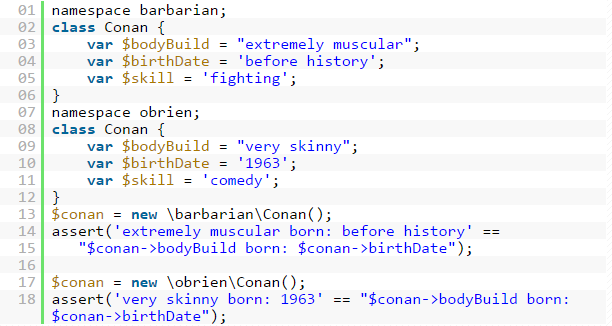
Un espacio de nombre proporciona un contexto para un nombre. Por ejemplo, las dos clases que se muestran en el listado 1, tienen una colisión de nombres.

**Listado 1. Dos clases con el mismo nombre ocasionan una colisión de nombres, sin nombres de espacio**



Para especificar un namespace, simplemente debemos añadir una declaración namespace en la primera línea del código, como se muestra en el listado 2.

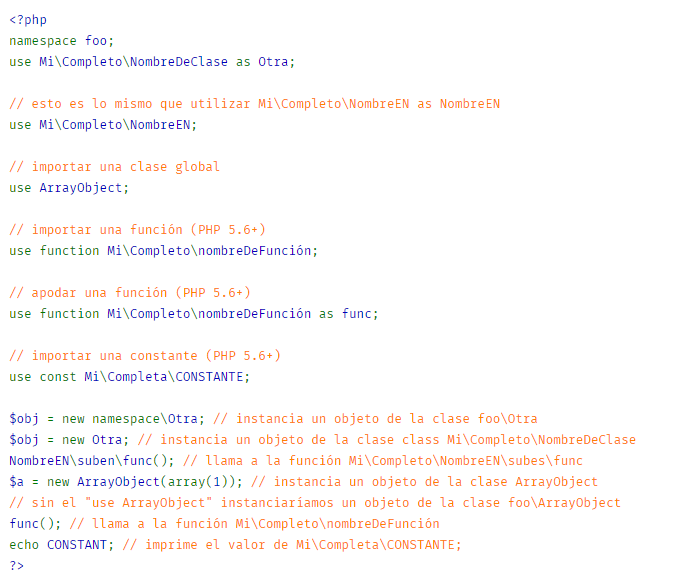
**Listado 2. Dos clases con el mismo nombre pero con namespaces resuelven la colisión**



***Use:***

Todas las versiones de PHP que tienen soporte para espacios de nombres admiten tres tipos de alias o importación: apodar un nombre de clase, apodar un nombre de interfaz, y apodar un nombre de espacio de nombres. PHP 5.6+ también permite apodar o importar nombres de funciones y constantes.

En PHP, la acción de apodar se lleva a cabo con el operador use. Aquí hay un ejemplo que muestra los cinco tipos de importación:



*Implements*

Para implementar una interfaz, se utiliza el operador implements. Todos los métodos en una interfaz deben ser implementados dentro de la clase; el no cumplir con esta regla resultará en un error fatal. Las clases pueden implementar más de una interfaz si se deseara, separándolas cada una por una coma.

**Nota:**

Antes de PHP 5.3.9, una clase no puede implementar dos interfaces que especifiquen un método con el mismo nombre, ya que podría causar ambigüedad. Las versiones más recientes de PHP permiten esto siempre y cuando los métodos duplicados tengan la misma firma.

**Nota:**

Las interfaces se pueden extender al igual que las clases utilizando el operador extends.

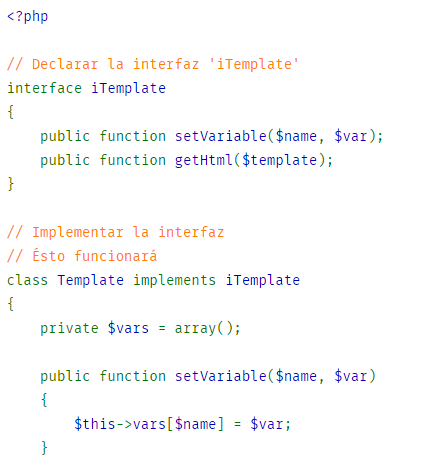
**Nota:**

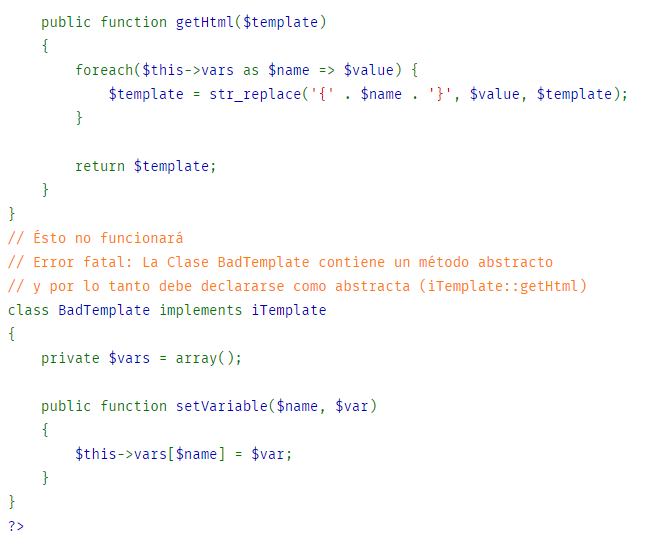
La clase que implemente una interfaz debe utilizar exactamente las mismas estructuras de métodos que fueron definidos en la interfaz. De no cumplir con esta regla, se generará un error fatal.

**Constantes**

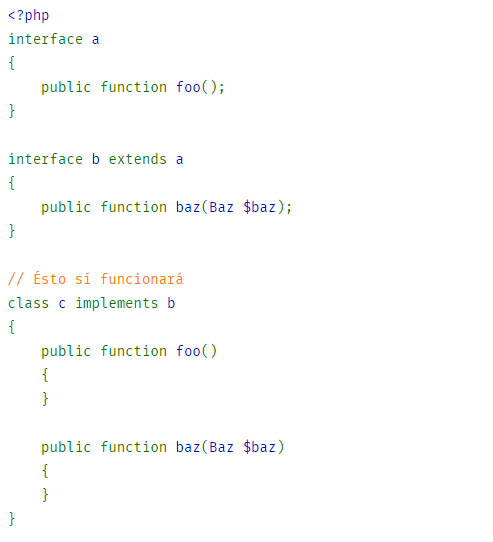
Es posible tener constantes dentro de las interfaces. Las constantes de interfaces funcionan como las constantes de clases excepto porque no pueden ser sobrescritas por una clase/interfaz que las herede.

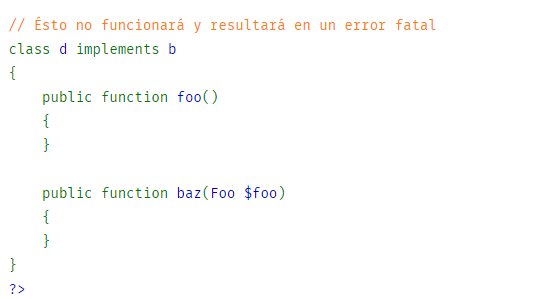
**Ejemplo #1 Ejemplo de interfaz**



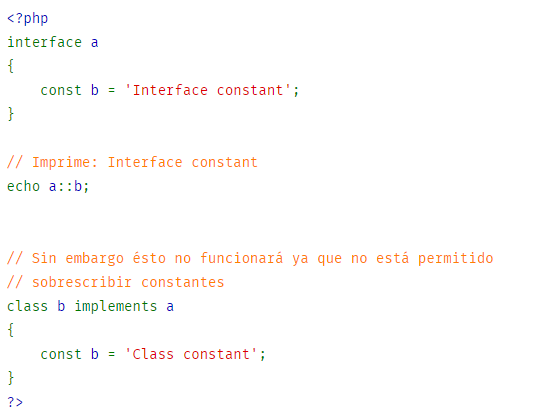


**Ejemplo #2 Interfaces extensibles**





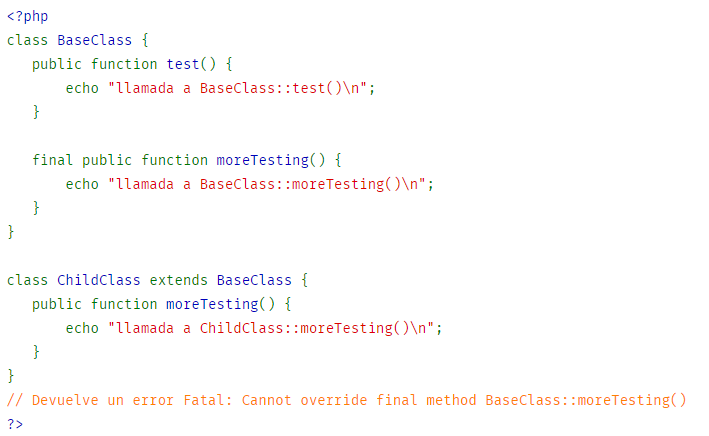
**Ejemplo #4 Interfaces con constantes**



***Final:***

PHP 5 introduce la nueva palabra clave final, que impide que las clases hijas sobrescriban un método, antecediendo su definición con final. Si la propia clase se define como final, entonces no se podrá heredar de ella.

**Ejemplo #1 Ejemplo de métodos Final**



**Ejemplo #2 Ejemplo de clase Final**

