

# Extract an interface refactoring

01/26/2018 • 2 minutes to read • 

## In this article

[Extract an interface refactoring](#)

[See also](#)

This refactoring applies to:

- C#
- Visual Basic

**What:** Lets you create an interface using existing members from a class, struct, or interface.

**When:** You have members in a class, struct, or interface that could be inherited by other classes, structs, or interfaces.

**Why:** Interfaces are great constructs for object-oriented designs. Imagine having classes for various animals (Dog, Cat, Bird) which might all have common methods, such as Eat, Drink, Sleep. Using an interface like IAnimal would allow Dog, Cat, and Bird to have a common "signature" for these methods.

## Extract an interface refactoring

1. Place your cursor in the class name.

- C#:

```
class Dog
{
    public void Eat()
    {
    }

    public void Drink(int value)
    {
    }

    public int Sleep()
    {
        return 0;
    }
}
```

- Visual Basic:

```
Class Dog
    Public Sub Eat()

    End Sub

    Public Sub Drink(value As Integer)

    End Sub

    Public Function Sleep() As Integer
        Return 0
    End Function
End Class
```

2. Next, do one of the following actions:

- **Keyboard**

- Press **Ctrl+R**, then **Ctrl+I**. (Your keyboard shortcut may be different based on which profile you've selected.)
- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Extract Interface** from the Preview window popup.

- **Mouse**

- Select **Edit > Refactor > Extract Interface**.
- Right-click the name of the class, select the **Quick Actions and Refactorings** menu and select **Extract Interface** from the Preview window popup.

3. In the **Extract Interface** dialog box that pops up, enter the information asked:

Extract Interface

New interface name:

IAnimal

Generated name:

IAnimal

Select destination

☐ Add to current file

☒ New file name: IAnimal.cs

Select public members to form interface

|                                     |  |            |
|-------------------------------------|--|------------|
| <input checked="" type="checkbox"/> |  | Drink(int) |
| <input checked="" type="checkbox"/> |  | Eat()      |
| <input checked="" type="checkbox"/> |  | Sleep()    |

Select All

Deselect All



| Field  | Description  |
|--|--|
| <b>New interface name</b>                      | The name of the interface to be created. The name will default to <i>IClassName</i> , where <i>ClassName</i> is the name of the class you selected above.  |
| <b>New file name</b>                           | The name of the generated file that will contain the interface. As with the interface name, this name will default to <i>IClassName</i> , where <i>ClassName</i> is the name of the class you selected above. You can also select the option to <b>Add to current file</b> . |
| <b>Select public members to form interface</b> | The items to extract into the interface. You may select as many as you wish.   |

4. Choose **OK**.

The interface is created in the file of the name specified. Additionally, the class you selected implements that interface.

- C#:

```
class Dog : IAnimal
{
    public void Eat()
    {
    }

    public void Drink(int value)
    {
    }

    public int Sleep()
    {
        return 0;
    }
}

interface IAnimal
{
    void Drink(int value);
    void Eat();
    int Sleep();
}
```

- Visual Basic:

```
Class Dog
    Implements IAnimal

    Public Sub Eat() Implements IAnimal.Eat
    End Sub

    Public Sub Drink(value As Integer) Implements IAnimal.Drink
    End Sub

    Public Function Sleep() As Integer Implements IAnimal.Sleep
        Return 0
    End Function
End Class
```

```
Interface IAnimal
    Sub Drink(value As Integer)
    Sub Eat()
    Function Sleep() As Integer
End Interface
```

## See also

- [Refactoring](#)
- [Tips for .NET Developers](#)

---

Is this page helpful?

 Yes  No

---