**Null Object**

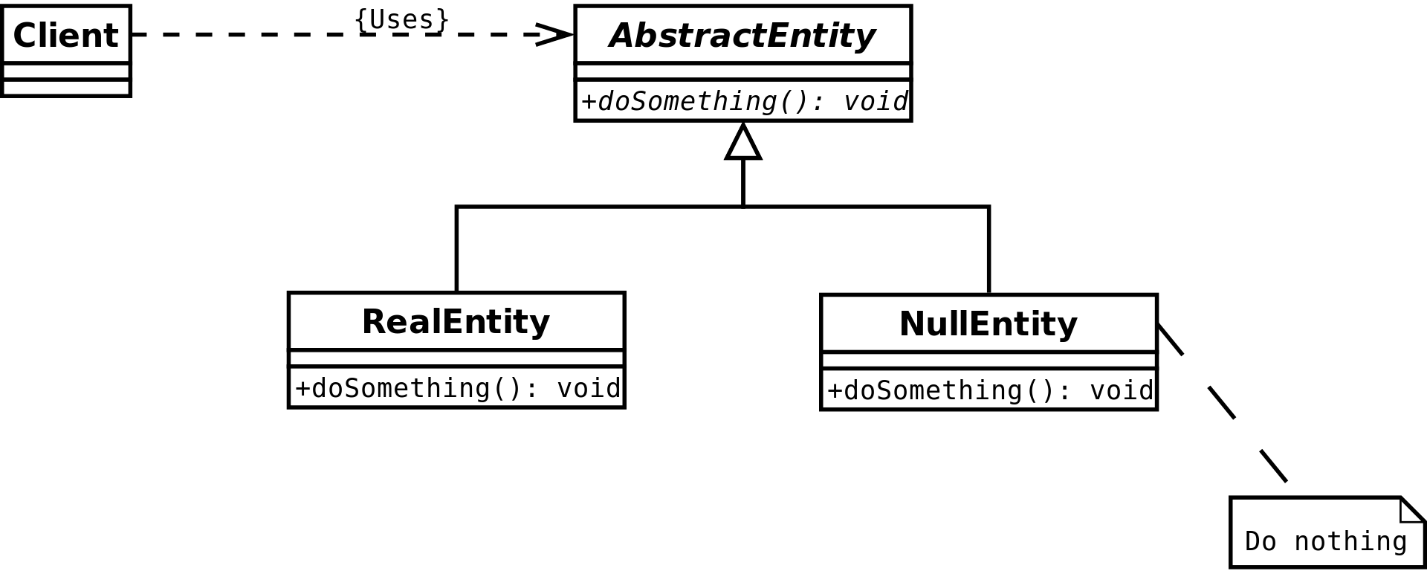
В объектно-ориентированном программировании , A нулевой объект является объектом, без ссылочного значения или с определенным нейтральным ( «NULL») поведением. Нулевой объект шаблон дизайна описывает использование таких объектов и их поведение (или его отсутствие).

В большинстве объектно-ориентированных языков программирования , таких как Java или C # , ссылки могут быть нулевым . Эти ссылки должны быть проверены , чтобы гарантировать , что они не равны нулю перед вызовом каких - либо методов , так как методы , как правило , не может быть вызван на нулевых ссылок.

Например, функция может получить список файлов в папке и выполнить некоторые действия по каждому. В случае пустой папки, один ответ может быть сгенерировано исключение или возвращает ссылку на нулевой, а не список. Таким образом, код, который ожидает, что список должен убедиться в том, что он на самом деле имеет один, прежде чем продолжить, что может осложнить конструкцию.

Возвращая нулевой объект (т.е. пустого списка), а, нет никакой необходимости, чтобы убедиться, что возвращаемое значение фактически список. Функция вызова может просто просматривать список, как обычно, эффективно ничего не делая. Это, однако, по-прежнему можно проверить, является ли возвращаемое значение является пустой объект (пустой список) и реагируют по-разному, если это необходимо.

Нулевая модель объекта также может быть использована, чтобы действовать в качестве заглушки для тестирования, если определенная функция, например, как база данных не доступна для тестирования.



Пример приложения:

Этот пример показывает объекты животного, отображающие звуки и NullAnimal экземпляр используется вместо C # нулевого ключевого слова. Нулевой объект обеспечивает согласованное поведение и предотвращает выполнения нулевого ссылочное исключения, которое будет иметь место, если C # нулевое ключевое слово использовало вместо этого.

**Пример кода:**

using System;

/// <summary>

/// Интерфейс Animal

/// <param name="MakeSound">Метод, позволяющий выводить звук животкого</param>

/// </summary>

interface IAnimal

{

void MakeSound();

}

/// <summary>

/// Cat - объект интерфейса IAnimal

/// <param name="MakeSound">Метод для работы объекта</param>

/// </summary>

/// <remarks>Выводит на консоль сообщение Животного</remarks>

class Cat : IAnimal

{

public void MakeSound()

{

Console.WriteLine("Meow!");

}

}

/// <summary>

/// Null Case: этот NullAnimal class будет сздавать экземпляр и использоваться как Null Object

/// <param name="MakeSound">Метод для работы Null объекта</param>

/// </summary>

class NullAnimal : IAnimal

{

public void MakeSound()

{

// do nothing

}

}

static class Program

{

/// <summary>

/// Точка входа для приложения.

/// </summary>

/// <param name="cat">Объект IAnimal для проверки класса Cat</param>

/// <param name="unknown">Объект IAnimal для проверки работы с Null Object</param>

/// <remarks>Переменные созданы для проверки приложения. Заместо того, чтобы использовать Null c#, используем NullObject (в случае этого приложения NullAnimal)

/// Использование такого паттерна проектирования не приведет у ошибке System.NullReferenceException, как в случае использования простого с# null</remarks>

static void Main()

{

IAnimal cat = new Cat();

cat.MakeSound();

IAnimal unknown = new NullAnimal(); //<< заменяет: IAnimal unknown = null;

unknown.MakeSound();

}

}