Null Object Design Pattern

Viviana Montiel
Cylia Boukhiba
Fabiana Montiel

Difficultés et préparation

Documentation en anglais

Traduction du contenu pour préparer l'exposé.

Découpage du contenu

Choisir les informations les plus pertinentes pour mieux expliquer le sujet.

Recherche

Documentation, compréhension, trouver des exemples faciles à comprendre et en C# ;)

Synchronisation à distance

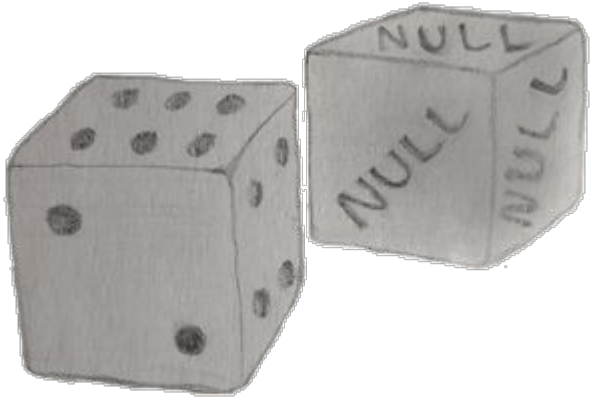
Mettre nos recherches ensemble dans un document pour ensuite préparer l'exposé. Google Doc et Google Presentation.

Consignes et feedback

Bien respecter la structure présentée par le prof ainsi que prendre en compte le feedback reçu de la dernière présentation.

Null Object

What is it?



The Null object pattern provides an object as a surrogate for the lack of an object of a given type.

The null object provides intelligent “do nothing” behavior, hiding the details from its collaborators.

Creational Patterns
Structural Patterns
Behavioral Patterns

What is this all about?

Intent	Problem	...and its drawbacks
We want to...	**But...**	**Refactoring...**
Encapsulate the absence of an object by providing a substitutable alternative that offers suitable default “do nothing” behavior.	How can the absence of an object — the presence of a null reference — be treated transparently?	Dozens of checks for null make your code longer and uglier.

Introductory problem

Problem

Since some methods return null instead of real objects, you have many checks for null in your code.

```
if (customer == null)
{
  plan = BillingPlan.Basic();
}
else
{
  plan = customer.GetPlan();
}
```

Solution

Refactoring... (some tips)

- Create a subclass that will perform the role of null object

- Find all places where the code may return null instead of a real object. Change the code so that it returns a null object.

Use the Null Object pattern when...

Case 1

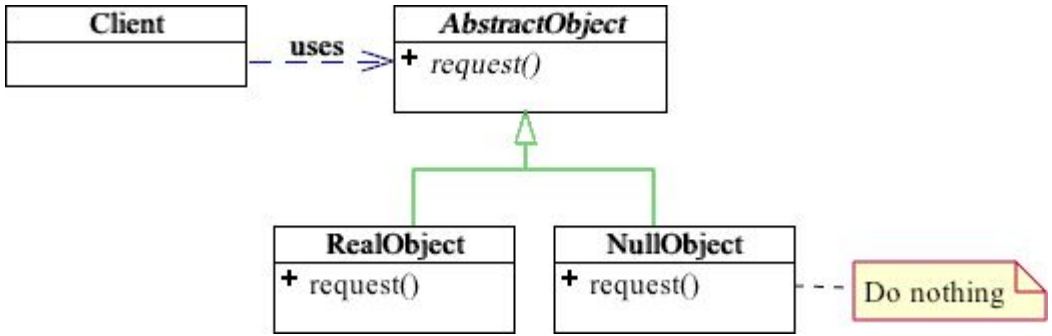
An object requires a collaborator. The Null Object Pattern does not introduce this collaboration, it makes use of a collaboration that already exists.

Case 2

Some collaborator instances should do nothing.

Case 3...

You want to abstract the handling of null away from the client.



Concrete example : *Man, Woman or Shadow*

```
interface Person{
    public void Breathe();
    public void Eat();
}

class Man : Person
{
    public override void Breathe(){
        Console.WriteLine("Oh");
    }

    public override void Eat(){
        Console.WriteLine("Mlask-mlask");
    }
}

class Woman : Person
{
    public override void Breathe(){
        Console.WriteLine("Ah");
    }

    public override void Eat(){
        Console.WriteLine("...");
    }
}
```

```
class Shadow : Person
{
    public override void Breathe()
    {

    }
    public override void Eat()
    {

    }
}
```

```
40  public class NullObjectTest
41  {
42      public static void Main()
43      {
44          Person foundPerson = Binoculars.LookForPerson(false);
45          foundPerson.Breathe();
46          foundPerson.Eat();
47          Person notFoundPerson = Binoculars.LookForPerson(true);
48          notFoundPerson.Breathe();
49          notFoundPerson.Eat();
50      }
51  }
52
53  class Binoculars
54  {
55      public static Person LookForPerson(bool isFog)
56      {
57          if(!isFog) return new Man();
58          return new Shadow();
59      }
60  }
```

Thank you ! :)

... we appreciate your time and attention !