

# WebQuest

## Aula Semana 09

### Mais Sobre Padrões de Projeto Básicos:

#### Static Factory Method, Null Object,

#### Hook Methods e Hook Classes

#### Introdução

O objetivo deste WebQuest é consolidar o entendimento e implementação dos seguintes padrões básicos: Static Factory Method, Null Object, Hook Methods e Hook Classes.

Um padrão é básico se ele é usado isoladamente ou como parte de outros padrões de projeto do livro GoF [Recurso Secundário 1].

Recomendo comprar o livro do Prof. Guerra [Recurso Secundário 2].

#### Tarefa

Conhecer, ver exemplos e exercitar o uso dos padrões de projeto básicos Static Factory Method, Null Object, Hook Methods e Hook Classes.

#### Processo

1. [Com seu colega do lado/da frente/de trás]
  - a. [05min] [Recurso Primário 1] Definir o que é e para que serve o padrão básico Static Factory Method, nomes alternativos e estrutura.

Método para adequar o nome do construtor da classe.  
Serve para quem facilitar a criação de objetos específicos daquela classe para o usuário.

A estrutura do Static Factory Method é a chamada do construtor da classe passando os parâmetros corretos de acordo com o nome do método.

- b. [10min] Dada a classe `RandonIntGenerator`, que gera números aleatórios entre um mínimo e um máximo, implemente-a passo-a-passo:

```
public class RandonIntGenerator {  
  
    public int next() {...}  
  
    private final int min;  
    private final int max;  
}
```

Como os valores `min` e `max` são final, eles devem ser inicializados na declaração ou via construtor. Vamos inicializar por meio de um construtor!

```
public RandonIntGenerator(int min, int max) {  
    this.min = min;  
    this.max = max;  
}
```

Crie um novo construtor, supondo que o valor `min` é fornecido e o valor `max` é o maior valor inteiro do Java (`Integer.MAX_VALUE`)!

```
public RandonIntGenerator(int min) {  
    this.min = min;  
    this.max = Integer.MAX_VALUE;  
}
```

Crie um novo construtor, supondo que o valor `max` é fornecido e o valor `min` é o menor valor inteiro do Java (`Integer.MIN_VALUE`)!

```
public RandonIntGenerator(int max) {  
    this.min = Integer.MIN_VALUE;  
    this.max = max;  
}
```

Como resolver este problema?

- c. [05min] Melhore a legibilidade do código abaixo:

```
public class Foo{  
    public Foo(boolean withBar){  
        //...  
    }  
}
```

```
//...

// What exactly does this mean?
Foo foo = new Foo(true);
// You have to lookup the documentation to be sure.
// Even if you remember that the boolean has something to do with a
// Bar, you might not remember whether it specified withBar or
// withoutBar.
```

### Solução:

```
Public class Foo{

    private Foo(boolean bar){

        //....

    }

    public static Foo createWithBar(){

        return new Foo(True);

    }

    public static Foo createWithoutBar(){

        return new Foo(False);

    }

}
```

- d. **[Exercício para Casa]** Em [Recurso Primário 1], estende-se o gerador de inteiro do item b) para suportar inteiro, Double, Long e String. Mostrar uma implementação com static factory methods que resolva essa situação
2. **[Com outro colega do lado/da frente/de trás][Mudar de local, se for preciso]**
  - a. **[05min]** Definir o que é e para que serve o padrão básico Null Object, nomes alternativos e estrutura.

Null Object é uma classe que possui todos os métodos em geral não implementados. Dessa forma, ele não faz nada.

Ele serve para começar a implementação de uma classe da forma mais simples possível. E comparado ao valor especial "null", o Null Object pode implementar interfaces para não fazer nada. Assim, o Null Object pode ser substituído por objetos reais.

Null Object é chamado de Special Case por Martin Fowler.

Estrutura é que o Null Object é filho de uma classe abstrata.

- b. **[10min]** Dada a classe RealCustomer abaixo, projetar e implementar um exemplo de aplicação simples, mostrando o antes (sem o padrão) e o depois (com o padrão) quando alguns clientes reais existem no repositório de clientes e outros ainda não fazem parte dele! Simular tudo o que for necessário para exemplificar a necessidade do uso do Null Object, inclusive o repositório de clientes!

```
public class RealCustomer {  
    public RealCustomer(String name) {  
        this.name = name;  
    }  
    @Override  
    public String getName() {  
        return name;  
    }  
    @Override  
    public boolean isNil() {  
        return false;  
    }  
}
```

3. **[Com outro colega do lado/da frente/de trás][Mudar de local, se for preciso]**

- a. **[05min]** Definir o que é e para que serve o padrão básico Hook Method, nomes alternativos e estrutura. [Recursos Primários 3 e 4]

Método Hook é baseado na sobreescrita de métodos que são chamados dentro de um método que é um template, ou seja, possui códigos em comum com a lógica das suas classes filhas, porém uma parte dentro desse método é variável dependendo da classe filha.

Estrutura é um método template da classe pai que possui código em comum com as filhas e no meio desse método possui uma parte que varia dependendo da classe filha, e é essa parte que é colocado dentro de um método hook que é sobrescrito pelas classes filhas, e que é chamado no método template da classe pai.

Nome alternativo template method definido como um comportamento de design pattern que define o esqueleto de uma algoritmo de deixa algumas partes para as subclasses definirem por sobreescrita sem alterar a estrutura geral do algoritmo.

- b. [10min] Pesquisar no [Recursos Primários 3 e 4] ou em qualquer outra fonte e projetar e implementar um exemplo de aplicação simples, mostrando o antes (sem o padrão) e o depois (com o padrão)!
4. [Com outro colega do lado/da frente/de trás][Mudar de local, se for preciso]
  - a. [07min] Diferencie hook method de hook class, começando com um exemplo não operacional em Java que implementa um hook method e transforme-o em hook class.

O hook class é quando uma classe implementa um algoritmo e algumas partes do algoritmo é delegado para uma outra classe, por relação de agregação, que implementa essas partes de formas diferentes. Essa outra classe é a hook class. Portanto, a diferença entre o hook method e o hook class é que um utiliza a propriedade de herança para implementar o hook method e o outro utiliza a relação entre classes, por exemplo, agregação para implementar uma hook class.

### Recursos Primários

1. [Static Factory Method] <http://jlordiales.me/2012/12/26/static-factory-methods-vs-traditional-constructors/>  
(former link: <http://jlordiales.wordpress.com>)
2. [Null Object] [https://sourcemaking.com/design\\_patterns/null\\_object](https://sourcemaking.com/design_patterns/null_object)
3. PDF com arquivo do link desativado <https://www.cs.oberlin.edu/~jwalker/nullObjPattern/> [TIDIA - Semana 09]
4. [Hook Methods 1] Hook Methods—Livro Guerra [TIDIA - Semana 09]
5. [Hook Methods 2] <http://c2.com/cgi/wiki?HookMethod>
6. [Hook Classes] Hook Classes—Livro Guerra [TIDIA - Semana 09]

### Recursos Secundários

1. Gamma, Erich; Richard Helm, Ralph Johnson, and John Vlissides (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ["Gang of Four" or GoF]

2. Eduardo Guerra. Design Patterns com Java: Projeto Orientado a Objetos Guiado por Padrões. São Paulo: Casa do Código, 2013. [ISBN 978-85-66250-11-4][e-Book R\$ 29,90]
3. Null Object apresentado como refatoração:  
<http://www.refactoring.com/catalog/introduceNullObject.html>
4. Null Object é chamado de "Special Case" no catálogo "EAA" do Fowler:  
<http://martinfowler.com/eaCatalog/specialCase.html>