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]
   1. [05min] [Recurso Primário 1] Definir o que é e para que serve o padrão básico Static Factory Method, nomes alternativos e estrutura.

R: É um modo de encapsular a criação de objetos, além de ser um construtor com nome "mais legível".  
Ela permite o uso de métodos diferentes com assinaturas iguais.

* 1. [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?

**RESPOSTAS:**

**public** static RandonIntGenerator bigger(**int** min) {

**return new** RandonIntGenerator(min, Integer.*MAX\_VALUE*);

}

**static public** RandonIntGenerator smaller(**int** max) {

**return new** RandonIntGenerator(Integer.*MIN\_VALUE,* max);

}

**static public** RandonIntGenerator between(**int** min, **int** max) {

**return new** RandonIntGenerator(min,max);

}

* 1. [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** **boolean** withBar;

**public** Foo(**boolean** withBar){

**this**.withBar = withBar;

}

**static** **public** Foo CreateWithBar(){

**return** **new** Foo(**true**);

}

**static** **public** Foo CreateWithOutBar(){

**return** **new** Foo(**false**);

}

}

* 1. [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

1. [Com outro colega do lado/da frente/de trás][Mudar de local, se for preciso]
   1. [05min] Definir o que é e para que serve o padrão básico Null Object, nomes alternativos e estrutura.

R: No caso de usar uma referência null para um objeto ou algo que nao existe, ela usa um objeto com metodos vazios, que nao fazem nada (mas pelo menos nao atrapalham ou explodem o código).

* 1. [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;

}

}

**RESPOSTA:**

**public** **abstract** **class** Customer {

**public** **abstract** String getName();

**public** **boolean** isNil() {

**return** **false**;

}

}

**public** **class** RealCustomer **extends** Customer{

**private** String name;

**public** RealCustomer(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

}

}

**public** **class** NullCustomer **extends** Customer{

**public** String getName() {

**return** "";

}

**public** **boolean** isNil() {

**return** **true**;

}

}

1. [Com outro colega do lado/da frente/de trás][Mudar de local, se for preciso]
   1. [05min] Definir o que é e para que serve o padrão básico Hook Method, nomes alternativos e estrutura. [Recursos Primários 3 e 4]

R: O método hook serve para você criar métodos concretos em uma classe abstrata e também métodos abstratos nela. Os métodos concretos fazem uso de um método abstrato. As classes que herdam essa classe abstrata fazem o uso de seus métodos concretos e implementam os métodos abstratos de forma específica (particular para cada classe). O que eu quero dizer é que uma nova subclasse pode extender o comportamento da superclasse.

* 1. [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)!

R:

**public** **abstract** **class** Canine {

**public** String BarkPrincipal(){

**return** "Auuuu" + BarkGancho();

}

**public** **abstract** String BarkGancho();

}

**public** **class** Dog **extends** Canine{

**public** String BarkGancho(){

**return** "uuuu";

}

}

**public** **class** Wolf **extends** Canine{

**public** String BarkGancho(){

**return** "uuuuuuuuuuuuuuuuuuuuuuuuuuu!!!";

}

}

1. [Com outro colega do lado/da frente/de trás][Mudar de local, se for preciso]
   1. [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.

R: O hook method utilizado será o do item b. dessa mesma questão.

Código do hook class:

**public** **class** CaninePrincipal {

**private** String latidoPrincipal;

**private** CanineGancho gancho;

**public** CaninePrincipal(String latido, CanineGancho gancho){

latidoPrincipal = latido;

**this**.gancho = gancho;

}

**public** String BarkPrincipal(){

**return** latidoPrincipal + gancho.BarkGancho();

}

}

**public** **abstract** **class** CanineGancho {

**public** **abstract** String BarkGancho();

}

**public** **class** Dog **extends** CanineGancho{

**public** String BarkGancho(){

**return** "uuuu";

}

}

**public** **class** Wolf **extends** CanineGancho{

**public** String BarkGancho(){

**return** "uuuuuuuuuuuuuuuuuuuuuuuuuuu!!!";

}

}

**Recursos Primários**

1. [Static Factory Method] <http://jlordiales.me/2012/12/26/static-factory-methods-vs-traditional-constructors/>

(former link: [http://jlordiales.wordpress.com](http://jlordiales.wordpress.com/2012/12/26/static-factory-methods-vs-traditional-constructors/))

1. [Null Object] https://sourcemaking.com/design\_patterns/null\_object
2. PDF com arquivo do link desativado <https://www.cs.oberlin.edu/~jwalker/nullObjPattern/> [TIDIA – Semana 09]
3. [Hook Methods 1] Hook Methods—Livro Guerra [TIDIA – Semana 09]
4. [Hook Methods 2] <http://c2.com/cgi/wiki?HookMethod>
5. [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 chamdo de “Special Case” no catalogo “EAA” do Fowler: <http://martinfowler.com/eaaCatalog/specialCase.html>