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.
    [O static factory method permite à uma classe ter um construtor estático, de forma que assim é possível estanciar um objeto da classe de formas diferentes. Usualmente só era possível definir um construtor mas sendo estes estáticos vários são possíveis.
    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 static RandonIntGenerator beetween(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 static RandonIntGenerator higher_than(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 static RandonIntGenerator less_than(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 static Foo withBar( boolean x)
```

foo = Foo.withbar(true);

   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.
   [Um null object é como se fosse o símbolo nulo de uma operação de objetos. Ele é útil para se evitar escrever mais código para definir exceções.
   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  abstract class Customer {
  public String getName() {
    return name;
    }
  public boolean isNil() {
    return false;
    }

}

public class RealCustomer {
  public RealCustomer(String name) {
    this.name = name;
    }


}

public class NullCustomer {
  public NullCustomer() {
    this.name = “nao existe”;
    }
    @Override
  public boolean isNil() {
    return true;
    }

}
```

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]
[O hook method é um método A dentro de outro método B. A idéia é que B contenha um código chave que é passível de mudança quando a classe é herdada. Assim, ao invés da necessidade de se reescrever todo o método herdado A por meio de override quando só partes descritas por B são de interesse de mudança, reescreve-se somente o método B.

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)!

```
Public int method(Equacao eq){
        TextoDeCodigoA
        TextoDeCodigoB
}
```

Se torna:

```
Public int method(Equacao eq){
        TextoDeCodigoA
        public int hook_method(Equacao eq)();
}

public int hook_method(Equacao eq){
        TextoDeCodigoB
}
```

Assim caso ocorra herança, e somente o TextoDeCodigoB está sob desejo de mudança por meio de override, não haverá necessidade de escrever o TextoDeCodigoA

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.

Ao utilizarmos hook classes, se colocarmos os
métodos na mesma classe, o mesmo problema irá acontecer, porém se pusermos

em classes diferentes, cada implementação poderá ser con"gurada de forma inde-

pendente.

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
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 chamdo de “Special Case” no catalogo “EAA” do Fowler: http://martinfowler.com/eaaCatalog/specialCase.html