Lucas Gonçalves

Guilherme Oliveira

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 funciona como uma espécie de encapsulamento na construção de objetos, criando um método

próprio para isso. Uma vantagem seria um método com nome diferente do construtor para criar-se o objeto. Outra
Vantagem seria fazer um setup dentro deste método static antes de chamar o construtor, podendo inclusive exceções serem tratadas dentro deste 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?

Pode-se fazer uso de vários Static Factory Method, com nomes diferentes, cada um deles chamaria o construtor com os parâmetros pertinentes.

```java
package aula09;

public class RandonIntGenerator {
	private RandonIntGenerator(int min, int max)
	{
		this._min=min;
		this._max=max;

	}
	public int get_min() {
		return _min;
	}

	public int get_max() {
		return _max;
	}

	private int _min;
	private int _max;

	public static RandonIntGenerator RandonIntGenerator_max_min(int max, int min)
	{
		return new RandonIntGenerator(min,max);


	}
	public static RandonIntGenerator RandonIntGenerator_min(int min)
	{
		return new RandonIntGenerator(min, Integer.MAX_VALUE);


	}
	public static RandonIntGenerator RandonIntGenerator_max(int max)
	{
		return new RandonIntGenerator( Integer.MIN_VALUE,max);


	}

	}
```

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

```java
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 withBar){
    //...
  }
public static Foo withBar(){
      return new Foo(true);
}
public static foo whithoutBar(){
      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.

Seria uma espécie de "novo" null criado pelo desenvolvidor. Uma vantagem seria obter métodos que nada fazem, muitas vezes uteis em testes de projetos.

   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 abstract String

        getName();

        public abstract boolean isNil();

        String name;
}

public class NilCustomer extends Customer {
```

```
        NilCustomer() {
                this.name = "";
        }

        @Override
        public String getName() {
                return "";
        }

        @Override
    public isNil(){
        return true;
    }
}

public class RealCustomer extends Customer {
        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]

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

Hook Method é um método criado com o intuito de dar override em apenas ma parte do método da classe Pai. Tal metodologia é útil quando deseja-se mudar apenas uma linha de um método muito grande na classe filha. Fazendo isso

evita-se a necessidade de se reescrever todo o método na classe filha.

d. [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)!

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

```
class Pai{
function(){
x = 2;
x = hook()*x;
return x;
}
int hook(){
return 2;
}

}

class F1 extends Pai{
      int hook(){

      return 3;
}
}
```

e. [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.

No hook method a diferença ocorre na implementação das classes :

```
//hook method
public class Customer{
   public String generateNome(){
      String a = "dsjafklçjaslkdjf";
      if(a=="boi") a = "dsjakl";
```

```
            a += getNomeHook();

            if(a.length()<3){
               a = "gru";
            }
            return a;


         }

         public String getNomeHook(){
            return "loucura";
         }
      }

      public class Filha extends Customer{
         public String getNomeHook(){
            return "filha";
         }
      }
```

No hook class a diferença ocorre no objeto da classe abstrata hook:

```
   //hook class
   public class Customer{
       public Customer(){
          n = new DefaultNome();
       }

       void setNome(Nome nome){
          n = nome;
       }

      public String generateNome(){
         String a = "dsjafklçjaslkdjf";
         if(a=="boi") a = "dsjakl";

         a += n.getNomeHook();
```

```
            if(a.length()<3){
               a = "gru";
            }
            return a;


         }

         Nome n;
      }

      public abstract class Nome{
         public abstract String getNomeHook();
      }

      public class DefaultNome extends Nome{
         public String getNomeHook(){
            return "loucura";
         }
      }

      public class OutroNome extends Nome{
         public String getNomeHook(){
            return "outroNome";
         }

}
```

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