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.
    R: Primeiramente, o padrão básico Static Factory Method serve para instanciar o objeto sem chamar diretamente o construtor. Dessa forma, esse método possibilita que exista um polimorfismo na construção do objeto, uma reutilização de objetos já instanciados, uma maior flexibilidade no tipo do objeto retornado. Por exemplo:

Complex c = Complex.createFromCartesian(double x, double y);

Complex c = Complex.createFromPolar(double norm, double angle);

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?

```
public class RandomIntGenerator {
      private final int min;
      private final int max;


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

```
	public static RandomIntGenerator byMin(int min) {
		return new RandomIntGenerator(min, Integer.MAX_VALUE);
	}

	public static RandomIntGenerator byMax(int max) {
		return new RandomIntGenerator(Integer.MIN_VALUE, max);
	}

	public int next(){
		return 0;
	}

}
```

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 withBar){

	}

	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.
R: Normalmente ao se usar ponteiros para objeto, o valor Null é usado para indicar ausência do objeto referenciado. No entanto, isso cria complicações por ser sempre necessário checar se o ponteiro é Null antes de chamar um método. Assim, o NullObject é uma implementação que respeita a interface do objeto, mas seus métodos não surtem nenhum efeito, podendo ser utilizado normalmente sem checagem.

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;
   }

}
```

Sem Padrão:

```
public static void main(String[] args) {
         RealCustomer customer = DB.getCustomer(11);
         if (customer)
               System.out.println(customer.getName());
         else
               System.out.println("Não Existe");
}
```

Com Padrão:

```
public class NullCustomer extends AbstractCustomer{

   public NullCustomer(String name) {
   }

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

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

```
public class RealCustomer extends AbstractCustomer {
	private String name;

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

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

	@Override
	public boolean isNil() {
		return false;
	}

}
```

```
public static void main(String[] args) {
		AbstractCustomer customer = DB.getCustomer(11);

		System.out.println(customer.getName());

}
```

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]

    R: O método Hook Method serve para deixar 'pendurado' a implementação de um método da superclasse. Dessa forma, torna-se necessário a implementação deste nas classes filhas. Por exemplo:

```
public class SuperClasse{
	run(){
	//set-up ambiente;
	metodoNaoImplementado();
	//destrói ambiente;
	}
	metodoNaoImplementado();
}
public class ClasseFilha extends SuperClasse{
	metodoNaoImplementado(){
	....}
```

```
}
```

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)!

Sem Padrão:

```
public abstract class Entity {
	public abstract void draw();
}
```

```
public class Player extends Entity{
	public void draw(){
		// struct whole player
		// set-up buffer
		// destroy buffer

	}
}
```

```
public class Monster extends Entity{
	public void draw(){
		// struct whole monster
		// set-up buffer
		// destroy buffer

	}
}
```

Com Padrão:

```
public abstract class Entity {
	public abstract void drawSprite();

	public void draw(){
		// set-up buffer;
		drawSprite();
		// destroy buffer;
	}
}
```

```
public class Player extends Entity{
	public void drawSprite(){
		// struct a player
	}
}
```

```
public class Monster extends Entity{
	public void drawSprite(){
		// struct a monster
	}
}
```

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.
    R:"No caso dos hook methods esse método está na mesma classe, podendo a implementação variar com a subclasse, e no caso das hook classes o método está em um objeto que compõe a classe, fazendo com que a implementação varie com a instância."

Hook Method:

```
public class RegistrationForm {

	public boolean validateName(String value) {
		// verifica se nao é vazio
		// verifica o máximo de caracteres
		// verifica se foram inseridos caracteres validos
		return true;
	}

	public boolean validateEmail(String value) {
		// verifica se o email existe
		// verifica se foi escrito um e-mail
		return true;
	}

	public boolean validate(String name, String email) {
		return validateName(name) && validateEmail(email);
	}

}
```

Hook Classe:

```
public class RegistrationForm {
	AbstractHook nameHook = new NameHook();
	AbstractHook emailHook = new EmailHook();

	public boolean validate(String name, String email){
		return
nameHook.validate(name)&&emailHook.validate(email);
	}

}
```

```
public abstract class AbstractHook {
	public abstract boolean validate(String value);

}
```

```
public class NameHook extends AbstractHook {
	public boolean validate(String value){
		// verifica se nao é vazio
```

```
            // verifica o máximo de caracteres
            // verifica se foram inseridos caracteres validos
            return true;
      }
}
```

```
public class EmailHook extends AbstractHook{
      public boolean validate(String value){
            // verifica se o email existe
            // verifica se foi escrito um e-mail
            return true;
      }
}
```

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