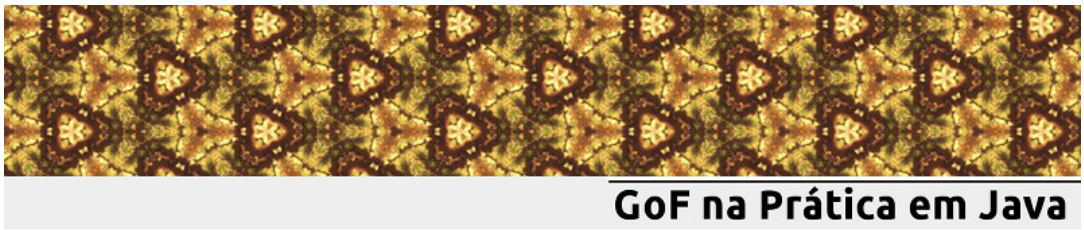
Prototype

Fernando Anselmo
GoF na Prática em Java

Função deste Padrão

Especifica os tipos de objetos a criar ao usar uma instância protótica, e cria os novos objetos através da cópia deste protótipo.



1 Ficha do Padrão

Tipo : Criacional, diz respeito ao processo de criação dos objetos.

Conhecimentos : Interface e Classes Concretas.

Consequências : Adiciona e remove produtos em tempo de execução. Especifica novos objetos pela variação de valores, e novos objetos pela variação de estruturas. Utiliza objetos compostos como protótipos. Reduz o número de subclasses (hierarquia) e Configura uma aplicação com classes dinamicamente.

É usado quando : O sistema deve ser independente de como os seus produtos são criados, compostos e representados. As classes a instanciar são especificadas em tempo de execução. Queremos evitar a construção de uma hierarquia de classes de fabricação paralela a uma hierarquia de classes de produtos por elas fabricados. Instâncias de uma classe podem ter uma de algumas poucas combinações de estados diferentes. Pode ser mais conveniente instalar um número correspondente de protótipos e cloná-los, ao invés de instanciar as classes no estado apropriado manualmente, toda vez que for necessário.

2 Problema

É necessário criar um novo objeto de um classe, este deve ser a cópia atualizada sem ser o mesmo objeto, porém o usuário não deve passar pelos processos de criação de clone e sim usar uma interface amigável para tal.

3 Prévia Estrutura de Classes

Classe a ser clonada:

Listagem 1: *Classe Ovelha*

```
1 class Ovelha implements Cloneable {
2   private String nome;
3
4   public Ovelha(String nome) {
```

```
    this.nome = nome;
  }
  public String toString() {
    return nome;
  }
  public Object clone() {
    Object object = null;
    try {
      object = super.clone();
    } catch (CloneNotSupportedException e) {
      System.out.println("Erro: " + e.getMessage());
    }
    return object;
  }
}
```

4 Aplicação do Padrão

Interface com a estrutura para a prototipação:

Listagem 2: *interface Prototype*

```
interface Prototype {
  public Ovelha duplicar(Ovelha ovelha);
}
```

Classe exemplo da implementação da prototipação:

Listagem 3: *Classe PrototypeImpl*

```
class PrototypeImpl implements Prototype {
  public Ovelha duplicar(Ovelha ovelha) {
    return (Ovelha)ovelha.clone();
  }
}
```

Classe com um exemplo de uso pelo cliente:

Listagem 4: *Classe Laboratorio*

```
public class Laboratorio {
  public static void main(String[] args) {
    new Laboratorio().executar();
  }
  public void executar() {
    Ovelha original = new Ovelha("Dolly");
    Ovelha clone = new PrototypeImpl().duplicar(original);
    System.out.println(original + " e seu clone " + clone);
    System.out.println((clone == original)?"Mesmo Objeto":"Objetos Diferentes");
  }
}
```

Referências

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides *Design Patterns. Elements of Reusable Object-Oriented Software 1 ed.* Estados Unidos, Addison-Wesley, 1995, ISBN 0-201-63361-2.