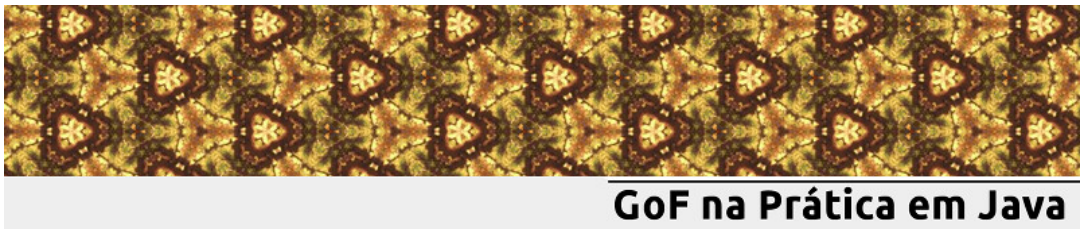
Singleton

Fernando Anselmo
GoF na Prática em Java

Função deste Padrão

Garantir que uma classe tenha apenas uma instância de objeto, e prover um ponto de acesso global a ela.



1 Ficha do Padrão

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

Conhecimentos : Classes Concretas.

Consequências : Acesso é totalmente controlado por uma instância. Espaço de nomes reduzido. Permite refinamento de operações e representação via especialização. Uma maior flexibilidade do que em operações de classes.

É usado quando : Deve haver exatamente uma única instância de uma classe, e esta deve estar a disposição de todos os clientes através de um ponto de acesso bem definido. Se deseja que a única instância possa ser estendida por herança, e os clientes serem capazes de utilizar essa instância estendida sem ter que modificar o código.

2 Problema

O cliente deseja criar vários objetos de uma determinada classe, porém todos esses objetos devem ser a mesma instância e não deve ser permitido ao cliente usar a construção padrão através da palavra chave new.

3 Prévia Estrutura de Classes

Classe de Conta Corrente que deve ser única:

Listagem 1: *Classe ContaCorrente*

```
1  class ContaCorrente {
2    private String numeroConta;
3
4    public void setNumeroConta(String numeroConta) {
5      this.numeroConta = numeroConta;
6    }
7    public String toString() {
8      return numeroConta;
9    }
10 }
```

4 Aplicação do Padrão

Nova classe de Conta Corrente com o padrão aplicado:

Listagem 2: *Classe ContaCorrente*

```
class ContaCorrente {
  private String numeroConta;
  private static ContaCorrente contaCorrente;

  public void setNumeroConta(String numeroConta) {
    this.numeroConta = numeroConta;
  }
  public String toString() {
    return numeroConta;
  }
  private ContaCorrente(){
  }
  public static synchronized ContaCorrente newInstance() {
    if (contaCorrente == null) {
      contaCorrente = new ContaCorrente();
    }
    return contaCorrente;
  }
}
```

Classe com um exemplo de uso pelo cliente:

Listagem 3: *Classe Cliente*

```
public class Cliente {
  public static void main(String[] args) {
    new Cliente().criarContas();
  }
  public void criarContas() {
    ContaCorrente ct1 = ContaCorrente.newInstance();
    ct1.setNumeroConta("123-45");
    System.out.println(ct1 + " - " + ct1.hashCode());
    ContaCorrente ct2 = ContaCorrente.newInstance();
    ct2.setNumeroConta("323-55");
    System.out.println(ct2 + " - " + ct2.hashCode());
  }
}
```

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.