DigitalHouse>
Coding School

Certified Tech Developer
The Ultimate Degree

Boas Práticas Singleton

Uma abordagem bastante antiga onde temos a classe com um construtor privado e exportamos um membro estático público para dar acesso à instância exclusiva. O exemplo abaixo demonstra como isso é feito em Java:

```
1  public class GerenciadorDeJanelas {
2    public static final GerenciadorDeJanelas INSTANCE = new GerenciadorDeJanelas();
3
4    private GerenciadorDeJanelas() {
5    }
6  }
```

O construtor privado é chamado apenas uma única vez para inicializar o campo final estático **INSTANCE**. Como não temos um construtor público ou protegido, temos a garantia de que existirá apenas uma instância para a classe GerenciadorDeJanelas.

Outra abordagem para implementar o Singleton é através de um método de fabricação estático.

```java
public class GerenciadorDeJanelas {
  private static final GerenciadorDeJanelas INSTANCE = new GerenciadorDeJanelas();

  private GerenciadorDeJanelas() {
  }

  public static GerenciadorDeJanelas getInstance() {
    return INSTANCE;
  }
}
```

Temos no exemplo acima que todas as chamadas a GerenciadorDeJanelas.getInstance() retornam a mesma referência de objeto. A grande vantagem desta abordagem é que através de um campo público, torna-se claro que esta classe é um Singleton. Outra vantagem dessa abordagem é se quisermos retornar uma instância diferente para cada chamada, ou seja, mudar completamente o comportamento. Nesse tipo de implementação, alterar esse comportamento seria bastante simples.

```
public interface Cuidador{

  public String guarda();


}
```

```
public interface Labrador{

  public String latir();


}
```

```
public class Doberman implements Cuidador, Labrador{

  public String guarda(){
        return “Estou atento guardando a casa”;
  }



  public String latir(){
       return “Guau! Guau!”;
  }


}
```

```
public class Lobo implements Labrador{


  public void String latir(){
       return “guau! Os lobos também latem”;
  }


}
```

```
/*Dada uma referência labrador do tipo Labrador (Labrador labrador)*/

    labrador = new Doberman(); //labrador é agora do tipo Doberman()
    System.out.println(labrador.latir()); //Polimorfismo
```

```
/*Dada uma referência labrador do tipo Labrador (Labrador labrador)*/

    labrador = new Lobo(); //labrador é agora do tipo Lobo()

    System.out.println(labrador.latir()); //Polimorfismo
```