ARQUITETURA E PROJETO DE SISTEMAS
Lista 1º Bimestre

Prof. Igor Avila Pereira

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Divisão de Computação

igor.pereira@riogrande.ifrs.edu.br

1. Strategy

1.1. Loja Virtual

Uma loja virtual possuiu alguns produtos a venda. Os produtos são livros, DVDs e brinquedos. Cada produto possui nome e preço. A mesma loja oferece promoções diferentes a cada mês. Uma promoção regular desconta cada produto em 10% mais um desconto extra varia de 5% a 10% dependendo do mês. Uma liquidação desconta 30% ao preço de cada produto. Há meses que não há promoção descrita. **Implemente o Padrão Strategy.**

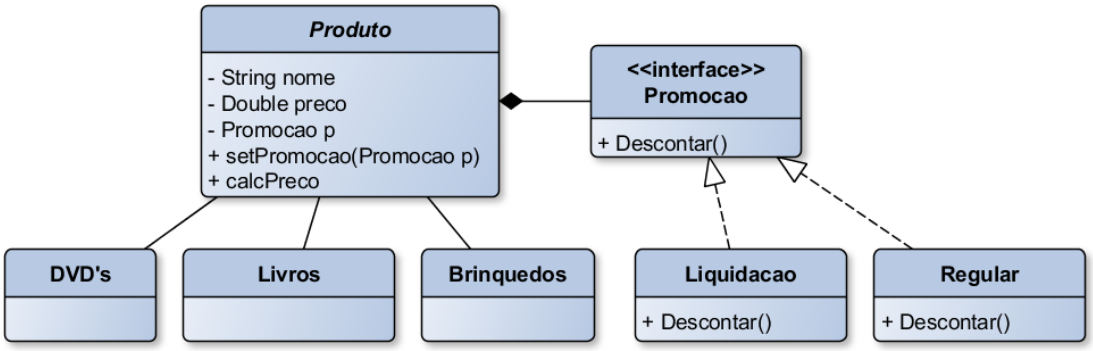

Figure 1. UML: Loja Virtual

1.2. Módulo de Pagamento

Pense em website de vendas online. Durante um processo de compra o usuário seleciona itens e determina o processo de pagamento que pode ser através do boleto, através do cartão de crédito e através do cartão de débito. **Implemente o padrão STRATEGY que determine as diversas formas de pagamento.**

2. Observer

2.1. Telefone

Como projetar um sistema que modele um telefone e todos os objetos que poderiam estar interessados quando ele toca? **Implemente o padrão Observer para o problema proposto.**

Os objetos interessados poderiam ser:

- Pessoas que estejam perto (na mesma sala)

- Uma secretária eletrônica
- Um FAX
- Até um dispositivo de escuta clandestina

Os objetos interessados podem mudar dinamicamente:

- Pessoas entram e saem da sala onde o telefone está.
- Secretárias eletrônicas, FAX, etc. podem ser adicionados ou removidos durante a execução do programa
- Novos dispositivos poderão ser inventados e adicionados em versões futuras do programa

2.2. Revista de Fofoca Online

Pense em uma site de uma revista de fofocas online. Nela há uma lista de assinantes. Quando uma nova notícia é cadastrada, todos os assinantes são notificados. Há vários tipos de assinantes: pessoa física, pessoa jurídica, idosos e etc. **Implemente o padrão Observer para o problema proposto.**

3. Active Record

3.1. Um pouco de teoria

Com Active Record, cada objeto do modelo tem a lógica necessária para ser persistido em um Banco de Dados relacional. Em outras palavras, Active Record encapsula a lógica para ser criar uma linha na tabela correspondente a respectiva classe. Isso é, cada objeto do modelo sabe como ser persistido.

Como consequência disso, o código fica mais leve e rápido para ser escrito para sistemas simples mas, em contrapartida, como existe uma grande dependência entre o modelo e o esquema relacional do Banco de Dados, qualquer mudança em um, implica mudança no outro. Isso torna mais difícil a manutenção do código.

3.2. Exercício - Problema

Dada a classe usuário, implemente os métodos *save* e *delete*. Construa também o SGBD

```
1  public class Usuario{
2      private int id;
3      private String login;
4      private String senha;
5      protected Connection con;
6
7      public Usuario(String login, String senha){
8          this.login = login;
9          this.senha = senha;
10     }
11     public String getLogin() {return login;}
12     public String getSenha() {return senha;}
13
14     public void save(){
15         // implemente aqui
```

```
16      }
17      public boolean delete(){
18          // implemente aqui
19      }
20  }
```

4. DAO

5. Um pouco de Teoria

Este padrão permite criar as classes de dados independentemente da fonte de dados ser um BD relacional, um arquivo texto, um arquivo XML, etc. Para isso, ele encapsula os mecanismos de acesso a dados e cria uma interface de cliente genérica para fazer o acesso aos dados permitindo que os mecanismos de acesso a dados sejam alterados independentemente do código que utiliza os dados.

Existem diversas implementações do padrão DAO mas em geral podemos relacionar algumas características desejáveis em uma implementação do padrão DAO:

- Todo o acesso aos dados deve ser feita através das classes DAO de forma a se ter o encapsulamento;
- Cada instância da DAO é responsável por um objeto de domínio;
- O DAO deve ser responsável pelas operações CRUD no domínio;
- O DAO não deve ser responsável por transações , sessões ou conexões que devem ser tratados fora do DAO;

5.1. Exercício - Problema

Implemente também a alternativa DAO para o exercício anterior.

6. Template Method

6.1. Um pouco de teoria

O Padrão Template Method é um padrão de projeto comportamental que define o esqueleto de um algoritmo na superclasse mas deixa as subclasses sobrescreverem etapas específicas do algoritmo sem modificar sua estrutura.

Um Template Method auxilia na definição de um algoritmo com partes do mesmo definidos por métodos abstratos. As subclasses devem se responsabilizar por estas partes abstratas, deste algoritmo, que serão implementadas, possivelmente de várias formas, ou seja, cada subclasse irá implementar à sua necessidade e oferecer um comportamento concreto construindo todo o algoritmo.

O Template Method fornece uma estrutura fixa, de um algoritmo, esta parte fixa deve estar presente na superclasse, sendo obrigatório uma *classeAbstrata* que possa conter um método concreto, pois em uma interface só é possível conter métodos abstratos que definem um comportamento, esta é a vantagem de ser uma Classe Abstrata porque também irá fornecer métodos abstratos às suas subclasses, que por sua vez herdam este método, por Herança (programação), e devem implementar os métodos abstratos fornecendo um comportamento concreto aos métodos que foram definidos como abstratos. Com isso certas partes do algoritmo serão preenchidos por implementações que irão variar, ou seja, implementar um algoritmo em um método, postergando a definição de alguns passos do algoritmo, para que outras classes possam redefini-los.

6.2. Exercício - Problema

Uma empresa criou uma classe responsável pelo cálculo do frete. Na medida que a empresa cresce normas formas de venda surgem e, consequentemente, novas modalidades para o cálculo do frete também começam a surgir. Entretanto, a empresa percebeu que o cálculo do frete, independente da modalidade, pode ser dividido em 3 etapas: (1) taxa do frete devido a modalidade e a distância; (2) valor do frete devido ao peso do produto e (3) o desconto devido aos cupons promocionais (caso haja algum). A única etapa que muda é a primeira pois para cada modalidade de frete (Sedex, convencional ou por Transportadora) temos uma regra de negócio diferente e um preço base diferente. Implemente o padrão Template Method para cada uma das modalidades de frete.

7. Anexos: UML

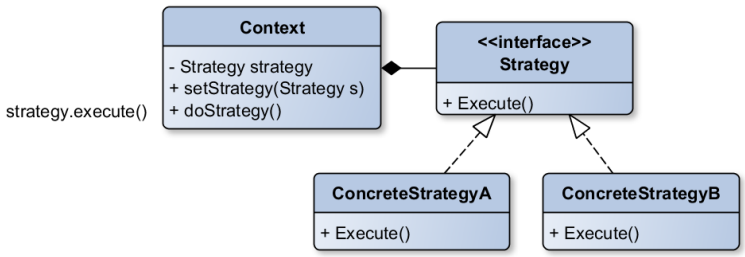

Figure 2. Padrão Strategy

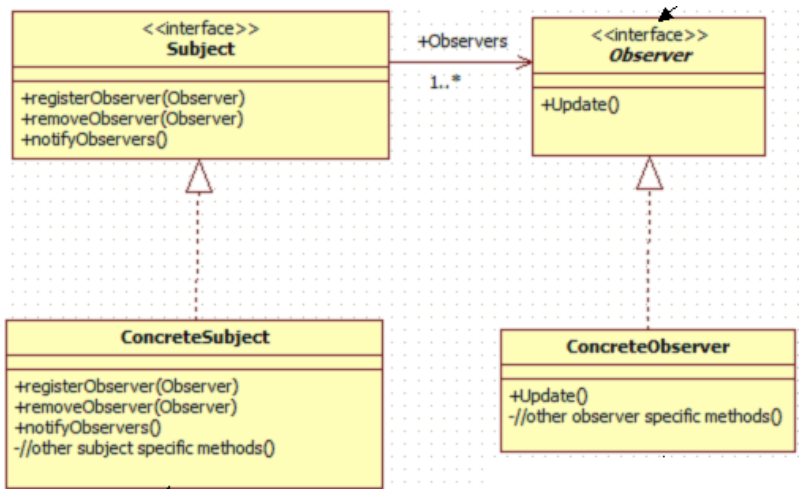

Figure 3. Padrão Observer

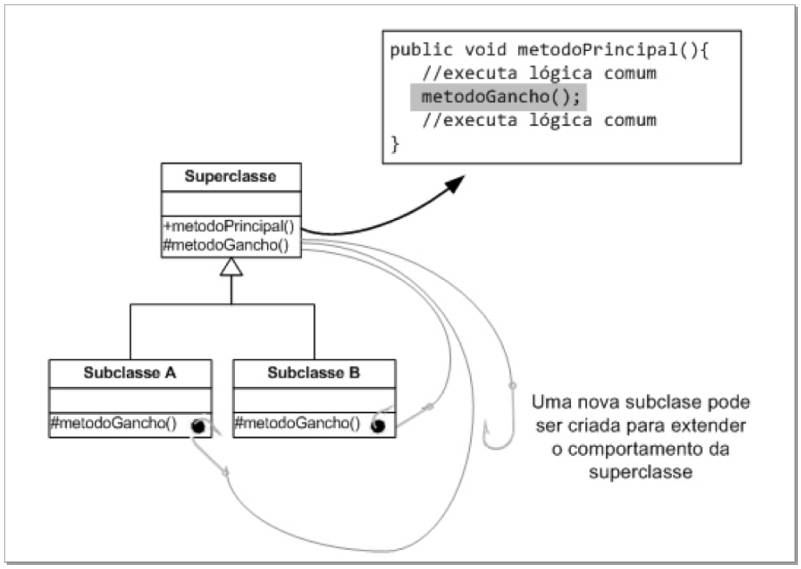

Figure 4. Padrão Template Method