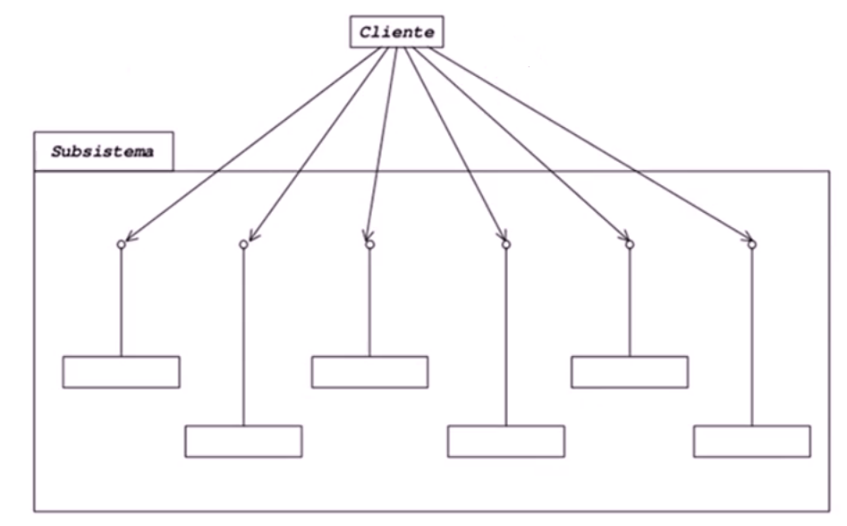
**Roteiro 06**

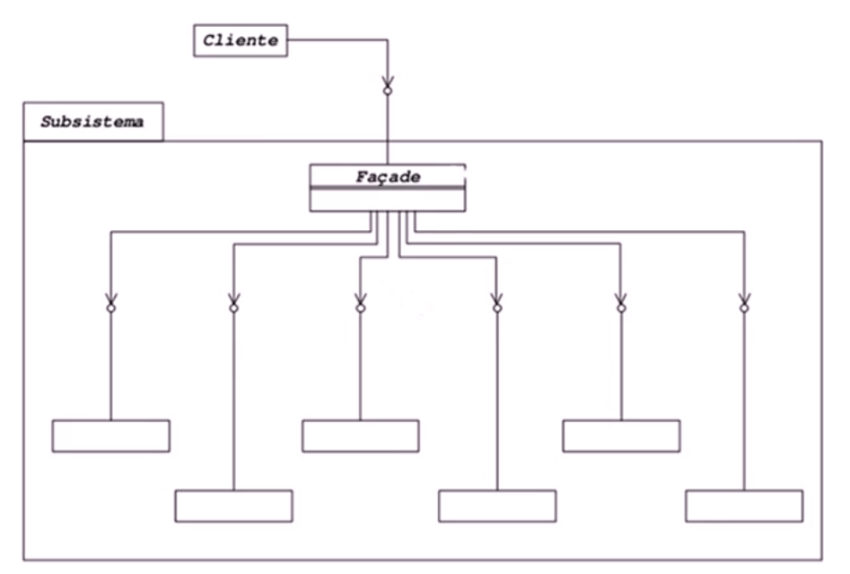
Neste roteiro trataremos do padrão Facade.

**Padrão Facade** – Tem como objetivo fornecer uma interface unificada para um conjunto de interfaces em um subsistema. O padrão facade define uma interface de nível mais alto que torna o subsistema mais fácil de ser utilizado.



Observe na figura 1 que o Cliente precisa se comunicar com diversos objetos presentes em um subsistema. Neste caso, existe um forte acoplamento entre o cliente e os diversos objetos do subsistema.

O objetivo do Facade é facilitar o uso de toda esta API (Interface de Programação de Aplicativos). Ou seja, todas as classes presentes neste subsistema.



A proposta é criar uma classe intermediaria entre a classe cliente e as classes do subsistema. Desta forma, a classe cliente precisa conhecer e se conectar apenas com uma classe (Facade) que por sua vez fará a interface com as outras classes facilitando o uso do subsistema. Reduzimos assim significativamente o acoplamento existente entre o cliente e as classes do subsistema.

Cenário:

Iremos simular um sistema de compra de produtos, onde o cliente pode selecionar uma relação de produtos do sistema e o sistema irá processar o pagamento.

**Pacote : roteiro6.parte1**

1 – Dentro do projeto criar um pacote chamado **roteiro6.parte1**

2 – Inicialmente crie a classe **Produto** dentro do pacote **roteiro6.parte1.** Observe abaixo que temos 3 atributos, o método construtor e a sobrecarga do método toString(). Adicione os gets e sets dos atributos.

package roteiro6.parte1;

public class Produto {

private int id;

private String nome;

private double preco;

public Produto(int id, String nome, double preco) {

this.id = id;

this.nome = nome;

this.preco = preco;

}

@Override

public String toString() {

return "Id : "+ this.id + "\n" +

"Nome : "+ this.nome + "\n"+

"Preço : "+ this.preco +"\n";

}

}

3 – Crie agora a classe **CarrinhoCompra**. Esta classe será responsável por acumular a lista de produtos e calcular o total da compra.

package roteiro6.parte1;

import java.util.ArrayList;

import java.util.List;

public class CarrinhoCompra {

private List<Produto> produtos;

public CarrinhoCompra() {

this.produtos = new ArrayList<Produto>();

}

public void addProduto(Produto p){

this.produtos.add(p);

}

public double getTotalCompra(){

double soma = 0;

for (Produto p: this.produtos){

soma = soma + p.getPreco();

}

return soma;

}

@Override

public String toString() {

String result = "";

for (Produto p: this.produtos) {

result = result + p.toString();

}

return result;

}

}

4 – A próxima classe a ser criada é **Cliente**. Observe abaixo que temos 3 atributos, o método construtor e a sobrecarga do método toString(). Adicione os gets e sets dos atributos.

package roteiro6.parte1;

public class Cliente {

private int id;

private String nome;

private CarrinhoCompra carrinho;

public Cliente(int id, String nome) {

this.id = id;

this.nome = nome;

}

@Override

public String toString() {

return "Cliente : " + this.id + "\n"+

"Nome : "+ this.nome + "\n";

}

}

5 – Criaremos agora uma classe chamada **Database**, em que irá simular a nossa base de dados de produtos e clientes. Observe que no construtor da classe já deixamos disponíveis 2 produtos (Prod A, Prod B), como se fossem os produtos a venda cadastrados previamente no nosso banco de dados.

public void addCliente(Cliente c){

this.clientes.add(c);

}

public void addProduto(Produto p){

this.produtos.add(p);

}

public void processarPagamento(Cliente c, double total){

System.out.println("Processando o pagamento : \n " + c);

System.out.println("Pagamento realizado com sucesso ! Total = " + total);

}

}

package roteiro6.parte1;

import java.util.ArrayList;

import java.util.List;

public class Database {

private List<Produto> produtos;

private List<Cliente> clientes;

public Database() {

this.produtos = new ArrayList<Produto>();

this.clientes = new ArrayList<Cliente>();

this.produtos.add(new Produto(1, "Prod A", 2.9));

this.produtos.add(new Produto(2, "Prod B", 3.1));

}

public Cliente selectCliente(int id){

for (Cliente c: this.clientes){

if (c.getId() == id)

return c;

}

return null;

}

public Produto selectProduto(int id){

for (Produto p: this.produtos){

if (p.getId() == id)

return p;

}

return null;

}

6 – Podemos agora criar uma classe de teste chamada **TesteSemFacade**, onde iremos simular todo o processo de compra das classes envolvidas.

package roteiro6.parte1;

public class TesteSemFacade {

public static void main(String[] args) {

// Criando uma referencia para o banco de dados

Database db = new Database();

// Criando o cliente e adicionando ao banco de dados

Cliente cli01 = new Cliente(1, "Jose");

db.addCliente(cli01);

// Criando o carrinho de compras e adicionando ao cliente criado

CarrinhoCompra carrinho = new CarrinhoCompra();

cli01.setCarrinho(carrinho);

// Cliente 01 comprando os produtos 1 e 2 que estão armazenados no DB

Produto p1 = db.selectProduto(1);

Produto p2 = db.selectProduto(2);

cli01.getCarrinho().addProduto(p1);

cli01.getCarrinho().addProduto(p2);

// Finalizando a compra

double total = cli01.getCarrinho().getTotalCompra();

db.processarPagamento(cli01, total);

}

}

7 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams

Obs: Outra possibilidade para geração dos diagramas é o StarUML : <https://staruml.io/> . Para fazer a engenharia reversa do código siga as orientações conforme o vídeo a seguir : <https://www.youtube.com/watch?v=z6DDuZQArro>

Seguindo para a parte 2 !

**Pacote : roteiro6.parte2**

Nesta etapa iremos aplicar o padrão Facade.

1 – No mesmo projeto crie o pacote roteiro6.parte2

2 – Copie todas as classes criadas na parte1 para o novo pacote.

3 – Crie agora uma classe chamada **Facade**, conforme segue abaixo

package roteiro6.parte2;

public class Facade {

private Database db;

private Cliente cliente;

private Produto produto;

private CarrinhoCompra carrinho;

public Facade() {

this.db = new Database();

}

public void registrarCliente(String nome, int id){

this.cliente = new Cliente(id, nome);

this.carrinho = new CarrinhoCompra();

this.cliente.setCarrinho(this.carrinho);

this.db.addCliente(this.cliente);

}

public void comprar(int productID, int clientID){

this.cliente = db.selectCliente(clientID);

this.produto = db.selectProduto(productID);

this.cliente.getCarrinho().addProduto(this.produto);

}

public void finalizarCompra(int clientID){

this.cliente = db.selectCliente(clientID);

double total = this.cliente.getCarrinho().getTotalCompra();

db.processarPagamento(cliente, total);

}

}

Observem que esta classe irá resumir todas as regras de negócio em 3 métodos :

- registrarCliente

- comprar

- finalizarCompra

4 – Vamos agora criar uma outra classe de teste chamada **TesteComFacade**, conforme segue abaixo

package roteiro6.parte2;

public class TesteComFacade {

public static void main(String[] args) {

Facade facade = new Facade();

facade.registrarCliente("Jose", 222);

facade.comprar(1, 222);

facade.comprar(2, 222);

facade.finalizarCompra(222);

}

}

5 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams

6 – Faça uma análise comparativa entre as classes **TesteSemFacade** e **TesteComFacade.**Avalie vantagens e desvantagens. Avalie possíveis impactos no caso de uma mudança na regra de negócio do sistema.

**Pacote : roteiro6.parte3**

Nesta etapa iremos fazer uma pequena mudança

1 – Dentro do projeto criar um pacote chamado **roteiro6.parte3**

2 – Na classe **Facade** vamos retirar a referência da classe CarrinhoCompra. Com essa mudança o cálculo da compra deve ser obtido pela classe Cliente. Abaixo temos em vermelho os trechos de código a serem retirados e em verde os trechos de código a serem adicionados.

package roteiro6.parte3;

public class Facade {

private Database db;

private Cliente cliente;

private Produto produto;

// private CarrinhoCompra carrinho;

public Facade() {

this.db = new Database();

}

public void registrarCliente(String nome, int id){

this.cliente = new Cliente(id, nome);

// this.carrinho = new CarrinhoCompra();

// this.cliente.setCarrinho(this.carrinho);

this.db.addCliente(this.cliente);

}

public void comprar(int productID, int clientID){

this.cliente = db.selectCliente(clientID);

this.produto = db.selectProduto(productID);

this.cliente.getCarrinho().addProduto(this.produto);

}

public void finalizarCompra(int clientID){

this.cliente = db.selectCliente(clientID);

//double total = this.cliente.getCarrinho().getTotalCompra();

double total = this.cliente.calcularTotal();

db.processarPagamento(cliente, total);

}

}

package roteiro6.parte3;

public class Cliente {

private int id;

private String nome;

private CarrinhoCompra carrinho;

public Cliente(int id, String nome) {

this.id = id;

this.nome = nome;

this.carrinho = new CarrinhoCompra();

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public CarrinhoCompra getCarrinho() {

return carrinho;

}

public void setCarrinho(CarrinhoCompra carrinho) {

this.carrinho = carrinho;

}

@Override

public String toString() {

return "Cliente : " + this.id + "\n"+

"Nome : "+ this.nome + "\n";

}

public double calcularTotal(){

return this.carrinho.getTotalCompra();

}

}

3 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams

4 – Faça uma pequena pesquisa e registre aqui o seu entendimento sobre Coesão x Acoplamento

5 – Faça uma pequena pesquisa sobre Lei de Demetrio (Lei de Demeter) ou Princípio do Conhecimento mínimo e registre aqui o seu entendimento.

6 – Qual associação você faz sobre a evolução do nosso projeto e as suas pesquisas feitas nas questões 4 e 5 ? Estes princípios e práticas de engenharia de software foram aplicados ? Como ?

**Pacote : roteiro6.parte4**

**Unindo os padrões Singleton e Facade**

1 – Dentro do projeto criar um pacote chamado **roteiro6.parte4**

2 – Copie todas as classes criadas da parte3 para o novo pacote.

3 – Se a ideia é unir os conceitos do Singleton e do Facade, aplique os conceitos do Singleton na classe já existente chamada **Facade**. Assim, esta classe terá uma função dupla.

4 – Que análise consegue fazer com essa refatoração ?.

5 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams