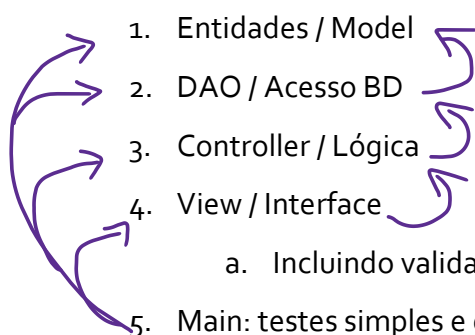


POO & Jornada - Atividade do 2º Bimestre

Complemento - Aula 11

O repositório de código orientado à objetos (preferencialmente Java) deverá estar estruturado por pacotes, que representarão as camadas do sistema, sendo os seguintes:

- 
1. Entidades / Model
 2. DAO / Acesso BD
 3. Controller / Lógica
 4. View / Interface
 - a. Incluindo validação do input
 5. Main: testes simples e diretos (a qualquer camada anterior)

A ordem de implementação e a divisão entre a equipe é de critério de cada grupo, embora minha recomendação seja que façam juntos todas as partes, para que todos entendam todo o processo. A seguir vamos detalhar um pouco cada etapa que deverá ser desenvolvida.

Entidades / Model

É o código que se preocupa apenas na modelagem e representação das entidades do seu sistema, onde as classes representarão tabelas do banco de dados, sendo entidades originais ou derivadas de relacionamentos.

Nesse ponto é necessário a definição das classes e a relação entre elas. As classes podem se relacionar através de agregação, composição, uso ou herança. Além disso, se for pertinente, é possível também a criação de classes ou métodos abstratos e interfaces.

Tudo feito nessa camada deve ser realizado em correspondência direta a estrutura do seu banco de dados.

Cada entidade deve implementar o toString(), para fácil visualização durante os testes manuais realizados.

DAO / Acesso ao BD

O código aqui é responsável direto pela conexão com o banco de dados, que será feito através da biblioteca java.sql.* (Statement, PreparedStatement, executeQuery(), executeUpdate())

Além disso, deverá ser criada uma classe abstrata DAO genérica, que deverá definir as assinaturas de métodos básicos de sistemas de informação:

1. Create: recebe uma entidade T e a insere no BD, não possui retorno
2. Read: recebe a chave de uma entidade T e retorna ela completa
3. ReadAll: sem parâmetros, retorna uma Lista de TODAS as entidades
4. Update: recebe uma entidade T e a insere no BD, não possui retorno
 - a. A diferença do Create é que a chave deve estar incluída!!
5. Delete: recebe a chave de uma entidade T e a exclui do BD

Todas as operações deverão declarar que lançam uma SQLException. Sendo assim, uma forma de ter um retorno sobre o sucesso ou não da operação é executá-la sempre em um try/catch, a fim de conseguir saber o status da operação e informar o usuário.

Controller / Lógica

Sobre essa camada:

- Ter uma classe abstrata, similar à do DAO
- Criação de conexão unificada com o banco de dados + repasse da conexão para o DAO específico (UserController <-> UserDAO)
- Chamar os métodos do DAO, envolvendo em try/catch e preparar os dados para a camada de interface
- Implementar qualquer lógica específica ao problema
 - Ex: alunosDaDisciplina(), manipula a lista de alunos retornados para repassar para visualização apenas os que fazem parte da disciplina "D"

View / Interface do Usuário

Também deverá possuir classe abstrata, podendo ser 1 ou mais. É possível pensar nas interações de “menus” e nas interações “exibe dados” ou “coleta dados”:

- Menu: apresenta opções e espera por uma escolha do usuário, também redireciona para métodos específicos que implementam as funcionalidades requisitadas
- Método – coleta de dados: informa o usuário o que deve ser inserido e em que ordem, utilizando Scanner para obter as informações
 - Métodos do tipo também devem ter processos de **validação** inclusos, ou dentro do método ou chamando um outro método para validar (ou seja, a podem decidir onde ela ficará, mas os campos devem ser validados)
 - A **validação** pode até utilizar do mecanismo de tratamento de erros! Usando throws, try e catch. A validação deve informar o usuário sobre o que deve ser corrigido para que ele possa realizar a correção adequada
- Método – saída de dados: adquire os dados do controller e os exibe de maneira formatada, considerando os atributos e se é objeto único ou Lista

Além disso, o menu deve ser multi-nível, possuindo pelo menos o “Menu Geral” e o menu de “Gerenciamento de Entidade”, sendo que a comunicação entre os menus deve ser controlada corretamente através dos laços de repetição e uso de break.

Main / Testes Diretos

Por simplicidade, pode ser uma classe única, com múltiplos métodos estáticos. Cada teste de uma funcionalidade específica (de qualquer camada) será um método estático separado e o método main() deve chamar todos os métodos de teste. Sendo assim, para facilitar a construção do sistema e ir testando um a um, após a funcionalidade funcionar, você pode comentar sua chamada. Quando quiser testar toda a arquitetura integrada, chame o método “menuGeral()” para iniciar testes.