Fuctura

# Revisão – Aula 26/11


Prof. Antonio Ricardo

# Conteúdo programático


- Projeto:
  - Apresentação;
  - Entidades;
  - DAO;
  - Controllers.
- Conexão com o banco:
  - JDBC;
  - JPA.

Fuctura

# Projeto - Apresentação


Criar um sistema de CRUD para gerenciar uma empresa de cursos.

Criar as classes de entidade para alunos, cursos e professores, criar os seus respectivos DAOs (interface e implementação)e fazer um menu com as opções para cadastrar, pesquisar, alterar e remover alunos, cursos e professores.

A entidade de aluno deve ter como atributos: nome, telefone, matricula, curso, turma e módulos concluídos. A entidade de professor deve ter como atributos: nome, telefone, curso e turmas. A entidade de curso deve ter como atributos: nomeDoCurso, professores, alunos, turmas e módulos.

# Projeto - Estrutura


src

- entidade
- dao
- util

# Projeto - Entidades


- Aluno:

  Deve ter como atributos: nome, telefone, matricula, curso, turma e módulos concluídos.
- Professor:

  Deve ter como atributos: nome, telefone, curso e turmas.
- Curso:

  Deve ter como atributos: nomeDoCurso, professores, alunos, turmas e módulos.

# Padrão – DAO (Data Access Object)


- Padrão de projeto que surgiu com a necessidade de separar a lógica de negócios da lógica de persistência de dados.
- Permite a mudança da forma de persistência sem que isso influencie na lógica de negócio, além de tornar as classes mais legíveis.
- As classes DAO são as responsáveis pela troca de informações com o SGBD (Sistema Gerenciador de Banco de Dados) e pelo fornecimento das operações de CRUD e de pesquisas.
- Toda interação com a base de dados é feita através destas classes, nunca através de classes de negócio, muito menos de formulários.

# Padrão – DAO (Data Access Object)


- Se aplicado corretamente, ele vai abstrair completamente o modo de busca e gravação de dados, fazendo com que essas operações sejam transparentes para a aplicação, tornando assim o trabalho de manutenção ou migração de banco de dados muito mais fácil.
- Com a centralização das operações da troca de dados com o SGBD, obtém-se um ponto único de acesso a dados fazendo com que a aplicação tenha um ótimo design orientado a objeto.

# Projeto - DAO


Criar as classes de entidade para alunos, cursos e professores, criar os seus respectivos DAOs (interface e implementação) e fazer um menu com as opções para cadastrar, pesquisar, alterar e remover alunos, cursos e professores.

# Projeto - DAO


Deve conter, pelo menos, os métodos para:

- Cadastrar;
- Pesquisar;
    - Listar todos;
    - Pesquisar por matrícula (id/código).
- Atualizar (alterar);
- Remover.

# Projeto - DAO


- Aluno:
  - cadastrarAluno;
  - listarAlunos;
  - pesquisarAluno;
  - atualizarAluno;
  - removerAluno.

# Projeto - DAO


- Professor:
  - cadastrarProfessor;
  - listarProfessores;
  - pesquisarProfessor;
  - atualizarProfessor;
  - removerProfessor.

# Projeto - DAO


- Curso:
  - cadastrarCurso;
  - listarCursos;
  - pesquisarCurso;
  - atualizarCurso;
  - removerCurso.

# Classe utilitária - List


- Interface que nos permite representar um grupo de objetos sequenciados
- Permite o armazenamento de elementos duplicados
- Fornece métodos para inserir, atualizar, pesquisar e deletar objetos (elementos) armazenados pela sua posição (índice) na lista
- Uma instância pode ser criada utilizando uma das classes que a implementam, como por exemplo ArrayList

# Classe utilitária - List


Métodos mais comuns:

- add(int index, Object obj)
- addAll(int index, Collection c)
- get(int index)
- indexOf(Object obj)
- iterator()
- remove(int index)
- set(int index, Object obj)

# Classe utilitária - ArrayList


- Classe que usa um array dinâmico para armazenar elementos.
- Se comporta como um array, porém pode ter o seu tamanho alterado, ou seja, podemos adicionar ou remover elementos quando quisermos.
- É mais flexivel que um array tradicional, pois implementa os métodos da interface List.

# Classe utilitária – List – Instanciando


Declaração:

```
List< tipoDoElemento > minhaLista = new ArrayList< tipoDoElemento >();
```

Exemplo:

```
List< String > minhaLista = new ArrayList< String >();
```

# Classe utilitária – List – Adicionando elementos


```
minhaLista.add(“elemento1");
minhaLista. add(“elemento2");
minhaLista. add(“elemento3");
```

# Classe utilitária – List – Iterando


```
Iterator<String> it = minhaLista.iterator();

while(it.hasNext()){
    System.out.println(it.next());
}
```

# Classe utilitária – List – Removendo elemento


```
it = minhaLista.iterator();

while(it.hasNext()){
    String elemento = it.next();
    if(elemento == "elemento2"){
        it.remove();
    }
}
```

# Projeto - Menu


Criar as classes de entidade para alunos, cursos e professores, criar os seus respectivos DAOs (interface e implementação) e fazer um menu com as opções para cadastrar, pesquisar, alterar e remover alunos, cursos e professores.

# Projeto - Menu


Podemos dividir o menu em, pelo menos, três áreas:

- Menu de alunos;
- Menu de professores;
- Menu de cursos.

# Projeto - Menu


Cada menu deve ter, pelo menos, as opções para:

- Cadastrar;
- Pesquisar;
- Atualizar;
- Remover.

# Projeto - Menu


Menus:

- Menu inicial;
- Menu de cursos;
- Menu de professores;
- Menu de alunos.