

PROGRAMAÇÃO PARA WEB I

AULA 7

Profa. Silvia Bertagnolli

USANDO DAO

PADRÃO DAO

O padrão de projeto DAO tem como objetivo evitar a exposição da camada de persistência para outras camadas

O DAO isola as operações JDBC, como criação, recuperação, atualização e exclusão, para um objeto de negócio

DEFININDO UMA INTERFACE GENÉRICA

A interface GenericDAO é usado para definir um comportamento padrão para todos os DAOs criados

```
public interface GenericDAO <T> {  
    public boolean insert(T obj);  
    public int update(T obj);  
    public int delete(T obj);  
    public java.util.List<T> listAll();  
    public T findById(int id);  
}
```

CRIAR CLASSE DAO

Classe que implementa a interface GenericDAO e é responsável pelo código que realiza as operações com o banco de dados

```
public class PessoaDAO implements GenericDAO<Pessoa>{  
  
    //...  
  
}
```

Analisar o código da classe PessoaDAO

PESSOADA0

```
public class PessoaDA0 implements GenericDA0<Pessoa>{  
    @Override  
    public int insert(Pessoa p) { ... }  
  
    @Override  
    public List<Pessoa> listAll() { ... }  
}
```

PESSOA

```
public class Pessoa {  
    // ...  
  
    public int insert() {  
        return new PessoaDAO().insert(this);  
    }  
  
    public static List<Pessoa> listAll(){  
        return new PessoaDAO().listAll();  
    }  
}
```

EXERCÍCIOS

EXERCÍCIOS

Faça os exercícios 17 a 19 da lista de exercícios

ENUM

ENUM

Definida na versão 1.5

Palavra reservada que significa enumeração

Usada para definir um conjunto de constantes de enumeração

As constantes enum são implicitamente static e final e não é possível alterar o seu valor

VANTAGENS DE USAR ENUM

É “type safe” – não é possível atribuir valor que não estejam definidos na enumeração

Tem seu próprio name-space

Dá mais legibilidade ao código

Adição de novas constantes é fácil

CARACTERÍSTICAS DE USAR ENUM

As instâncias dos tipos enum são criadas e nomeadas como se fosse uma declaração da classe, sendo fixas e imutáveis (o valor é fixo)

Não é possível criar instâncias com a palavra chave new

O construtor é private, embora não precise de modificador private explícito

CARACTERÍSTICAS DE USAR ENUM

São considerados objetos constantes e imutáveis (static final), lgo os nomes declarados recebem todas as letras em MAIÚSCULAS

As instâncias dos tipos enum devem obrigatoriamente ter apenas um nome

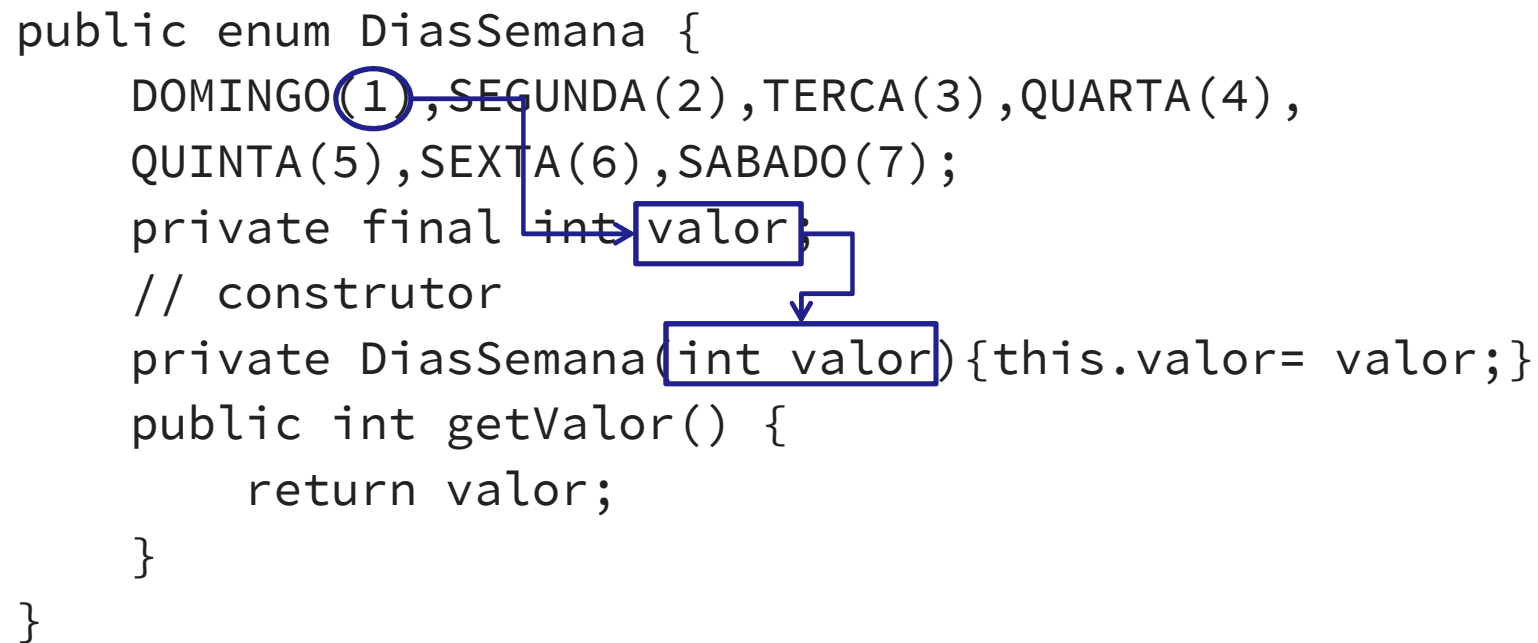
Podem incluir variáveis de instância, construtor, métodos de instância, de classe, etc.

DECLARANDO UMA ENUM

```
public enum DiasSemana {  
    DOMINGO(1),SEGUNDA(2),TERCA(3),QUARTA(4),  
    QUINTA(5),SEXTA(6),SABADO(7);  
    private final int valor;  
    // construtor  
    private DiasSemana(int valor){this.valor= valor;}  
    public int getValor() {  
        return valor;  
    }  
}
```

DECLARANDO UMA ENUM

```
public enum DiasSemana {  
    DOMINGO(1), SEGUNDA(2), TERCA(3), QUARTA(4),  
    QUINTA(5), SEXTA(6), SABADO(7);  
    private final int valor;  
    // construtor  
    private DiasSemana(int valor) { this.valor = valor; }  
    public int getValor() {  
        return valor;  
    }  
}
```



IMPRIMINDO

```
DiasSemana dia = DiasSemana.QUARTA;  
System.out.println("Dia:"+dia);
```

PERCORRENDO

```
for(DiasSemana dias : DiasSemana.values())  
    System.out.println("dias:"+dias.getValor());
```

COMPARANDO

```
public static void comparaEnum(DiasSemana opcao){  
    switch(opcao){  
        case DOMINGO:  
            System.out.println("Valor é domingo!");  
            break;
```

EXERCÍCIOS

EXERCÍCIOS

Faça os exercícios 20 a 23 da lista de exercícios

SQL COM ENUM

ENUMS PARA SQLS

```
public enum PessoaSQLs {  
    INSERT("insert into pessoa(nome, endereco) values (?, ?)"),  
    LISTALL("select * from pessoa");  
  
    private final String sql;  
    PessoaSQLs(String sql){  
        this.sql = sql;  
    }  
    public String getSql() {  
        return sql;  
    }  
}
```

ENUMS PARA SQLS

```
public class PessoaDAO implements GenericDAO<Pessoa>{

@Override
public List<Pessoa> listAll() {
    List<Pessoa> lista = new LinkedList<>();
    try(Connection connection = new
            ConnectionFactory().getConnection());
        PreparedStatement stmt =
connection.prepareStatement(PessoaSQLs.LISTALL.getSql())){
```


EXERCÍCIOS

EXERCÍCIOS

Faça os exercícios 24 e 25 da lista de exercícios

RELACIONAMENTOS

COMO IMPLEMENTAR ESSE RELACIONAMENTO?



COMO IMPLEMENTAR RELACIONAMENTOS 1..1?



Traduzindo para ER

Cpf(idcpf, numero, digito)

Empregado(idempregado, nome, endereco, telefone, dataNascimento, idcpf)

CLASSE CPF

```
public class Cpf {  
    private int idCpf;  
    private long numero;  
    private int digito;  
    public Cpf(){}  
    public Cpf(long numero, int digito) {  
        this(-1, numero, digito);  
    }  
    public Cpf(int idCpf, long numero, int digito) {  
        this.idCpf = idCpf;  
        this.numero = numero;  
        this.digito = digito;  
    }  
    //outras definições
```

CLASSE EMPREGADO

```
public class Empregado {  
    private int idEmpregado;  
    private String nome;  
    private String endereco;  
    private String telefone;  
    private Cpf cpf;
```

CLASSE EMPREGADO

```
public class Empregado {  
    //...  
    public int insert() {  
        return new EmpregadoDAO().insert(this);  
    }  
}
```


EXERCÍCIOS

EXERCÍCIOS

Faça os exercícios 26 e 30 da lista de exercícios