Aplicações de Linguagem de Programação Orientada a Objetos

Modulo 8 - Padrão DAO (Dta Access Object)

Prof. Sheila Cáceres

O padrão de projeto *DAO* surgiu para facilitar a comunicação entre as camadas de negócio e de persistência.

Seguindo este padrão, é possível substituir uma sequência complexa de troca de comandos entre a aplicação e o banco de dados por uma única chamada do tipo inserir(Objeto) ou apagar(Objeto).

// Pessoa.java

```
public class Pessoa {
	private int id;
	private String nome;
	private String sobrenome;
	private int idade;

	// ************ Constructores ******************
	public Pessoa(int id, String nome) {
		this.id = id;
		this.nome = nome;
	}

	public Pessoa(int id, String nome, String sobrenome, int idade) {
		this.id = id;
		this.nome = nome;
		this.sobrenome = sobrenome;
		this.idade = idade;
	}
	// ************* Gets e Sets ********************
	public String getNome() {	return nome;	}
	public void setNome(String name) {	this.nome = name; }
	public int getId() {	return id;  }
	public void setId(int id) {	this.id = id;	}
	public String getSobrenome() {	return sobrenome; }
	public void setSobrenome(String sobrenome) {	this.sobrenome =
sobrenome;  }
	public int getIdade() {	return idade;	}
	public void setIdade(int idade) {	this.idade = idade;	}
}
```

// IPessoaDao.java

```
import java.util.List;

public interface IPessoaDao {
   public List<Pessoa> getAllPessoas();
   public Pessoa findPessoa(int id);
   public void insertPessoa (Pessoa pessoa);
   public void updatePessoa(Pessoa pessoa);
   public void deletePessoa(Pessoa pessoa);
}
```

// PessoaDaoList.java

```
import java.util.ArrayList;
import java.util.List;

public class PessoaDaoList implements IPessoaDao {

   // *** Esta lista está substituindo um banco de dados ***
   List<Pessoa> pessoas;

   public PessoaDaoList(){
      pessoas = new ArrayList<Pessoa>();
   }
   @Override
   public void insertPessoa(Pessoa pessoa) {
         pessoas.add(pessoa);
   }

   @Override
   public void deletePessoa(Pessoa pessoa) {
      pessoas.remove(pessoa);
      System.out.println("Pessoa: Id " + pessoa.getId() +" apagada da lista
(bd)");
   }

   // Recupera a lista de estudantes
   @Override
   public List<Pessoa> getAllPessoas() {
      return pessoas;
   }

   @Override
   public Pessoa findPessoa(int id) {
        // TODO
   }

   @Override
   public void updatePessoa(Pessoa newPessoa) {
         // A pessoa já foi atualizada ao usar as funções set.
      System.out.println("A Pessoa com Id: " + newPessoa.getId() +", foi
atualizada na lista (bd)");
   }
}
```

// PessoaDaoJDBC.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class PessoaDaoJDBC implements IPessoaDao{
      private String driver = "org.postgresql.Driver";
      private String sUsuario = "postgres";
      private String sSenha = "root";
      private String sFonte = "jdbc:postgresql://localhost:5432/postgres";
      private Connection con;
      private Statement st;
```

```
	public PessoaDaoJDBC(){
		this.criarTabela();
	}

	public void openConnection(){
		try {
			Class.forName(driver);
			con = DriverManager.getConnection(sFonte, sUsuario, sSenha);
		} catch (Exception e) { // demais exceções
			e.printStackTrace();
			System.out.println("Falha com o driver ou com a conexão com o
banco!\n" +e.getMessage());
			System.exit(0);
		}
	}

	/* Cria a Tabela pessoa no banco de dados postgres
	 */
	public void criarTabela(){
		System.out.print("Criando a tabela pessoa: ");
		String sentencaSQL = "CREATE TABLE pessoa (id integer PRIMARY KEY,
nome VARCHAR(50), sobrenome VARCHAR(50), idade integer );";
		this.atualizar(sentencaSQL);
	}

	/* Realiza updates no banco de dados.
	*/
	public void atualizar(String sentencaSQL){
		try{
			this.openConnection();
			st = con.createStatement();// cria statement para consultar
banco de dados
			st.executeUpdate(sentencaSQL);
			System.out.println("Update com sucesso!");
			st.close();
			con.close();
		} catch (SQLException eSQL) {
			eSQL.printStackTrace();
			System.out.println("Não foi possível realizar o update!\n" +
eSQL.getMessage());
			System.exit(2);
		}
	}

	/* Realiza uma consulta sobre a tabela pessoa e retorna uma lista
contendo objetos Pessoa
	 */
	public List<Pessoa> consultarPessoas(String setencaSQL){
		List<Pessoa> pessoas = new ArrayList<Pessoa>();
		try {
			this.openConnection();
			st = con.createStatement();
			ResultSet rs = st.executeQuery(setencaSQL);
			while (rs.next()) {
				Pessoa p=new Pessoa(rs.getInt("id"),
rs.getString("nome"), rs.getString("sobrenome"), rs.getInt("idade"));
				pessoas.add(p);
			}
			rs.close();
			st.close();
			con.close();
```

```
            } catch (SQLException eSQL) {
                  System.out.print("Erro na consulta SQL" +
eSQL.getMessage() );
                  System.exit(1);
            }
            return pessoas;
      }

      @Override
      public void insertPessoa (Pessoa pessoa){
            System.out.print("Inserindo uma pessoa: ");
            String sentencaSQL = "INSERT INTO pessoa values("+pessoa.getId()
+","
                        + "'"+pessoa.getNome() +"','"+pessoa.getSobrenome()
+"',"+pessoa.getIdade()+");";
            this.atualizar(sentencaSQL);
      }

      @Override
      public void updatePessoa(Pessoa pessoa) {
            // TODO
      }

      @Override
      public List<Pessoa> getAllPessoas() {
            return this.consultarPessoas("SELECT * FROM pessoa;");
      }

      @Override
      public Pessoa findPessoa(int id) {
            // TODO
            return null;
      }

      @Override
      public void deletePessoa(Pessoa pessoa) {
            // TODO
      }
}
```

// DaoPatternDemo.java

```
public class DaoPatternDemo {

      public static void main(String[] args) {
            Pessoa pessoa0 = new Pessoa(0, "Joao", "Pereira", 25);
            Pessoa pessoa1 = new Pessoa(1, "Roberto", "Carvalho",30);
            Pessoa pessoa2 = new Pessoa(2, "Maria", "Ferraz",50);

            // Podemos usar qualquer implementacao de IPessoaDao
            IPessoaDao pessoaDao =new PessoaDaoList();// Ex: PessoaDaoList,
PessoaDaoJDBC

            // Inserindo pessoas
            pessoaDao.insertPessoa(pessoa0);
            pessoaDao.insertPessoa(pessoa1);
            pessoaDao.insertPessoa(pessoa2);

            for (Pessoa pessoa : pessoaDao.getAllPessoas()) {
```

```
                System.out.println("Pessoa => [Id: " + pessoa.getId()+
                            ", Nome : " + pessoa.getNome() +
                            ", Sobrenome : " + pessoa.getSobrenome() +
                            ", Idade : " + pessoa.getIdade() +

                            " ]");
        }

        // update pessoa
        Pessoa pessoa = pessoaDao.getAllPessoas().get(0);
        pessoa.setNome("Michael");
        pessoa.setSobrenome("da Silva");
        pessoaDao.updatePessoa(pessoa);

        System.out.println("Pessoa Atualizada => [Id: " + pessoa.getId()+
                    ", Nome : " + pessoa.getNome() +
                    ", Sobrenome : " + pessoa.getSobrenome() +
                    ", Idade : " + pessoa.getIdade() + " ]");

        pessoaDao.deletePessoa(pessoa0); //Apagar uma pessoa

        Pessoa p= pessoaDao.findPessoa(1); //Achar uma pessoa
        System.out.println("O nome da pessoa achada é: "+p.getNome());
    }
}
```

EXERCICIOS PARA LABORATÓRIO

1) Implemente as funções faltantes nas classes ***PessoaDaoList*** e ***PessoaDaoJDBC***

2) Adicione outra classe que implemente a interface IpessoaDao e teste desde ***DaoPatternDemo.java***

3) Adicione interfaces gráficas (pode se basear no JTable visto anteriormente)