Universidade Presbiteriana Mackenzie
Faculdade de Computação e Informática

Lista de Exercícios 2

NOTAS DE AULA - Teoria 12

Linguagem de Programação II
2º semestre de 2015
Versão 1.0

Enunciado

Na aula de laboratório desenvolvemos a classe **ContaDaoRelacional**:

```
public class ContaDaoRelacional implements ContaDaoInterface {
    private ConexaoInterface conexao;
    public ContaDaoRelacional(ConexaoInterface conexao) {
        this.conexao = conexao;
    }
    @Override
    public List<Conta> listarTudo() {
        List<Conta> contas = new ArrayList<>();
        try {
            Statement st = conexao.getConnection().createStatement();
            String sql = "SELECT nro_conta, saldo FROM contas";
            ResultSet resultados = st.executeQuery(sql);
            while (resultados.next()) {
                long n = resultados.getLong("nro_conta");
                BigDecimal b = resultados.getBigDecimal("saldo");
                Conta c = new Conta(n, b);
                contas.add(c);
            }
        } catch (SQLException ex) { ex.printStackTrace(); }
        return contas;
    }
}
```

continuação do Enunciado

A classe **ContaDaoRelacional** é uma implementação da interface **ContaDaoInterface**:

```
public interface ContaDaoInterface {
    List<Conta> listarTudo();
}
```

continuação do Enunciado

A classe **AppSelectContasComDao** mostra um exemplo de uso da classe ContaDaoRelacional para imprimir na tela as informações de todas as contas.

```
public class AppSelectContasComDao {
    public static void main(String[] args) {
        ConexaoInterface conexao;
        conexao = new ConexaoJavaDb("app", "app", "127.0.0.1", 1527,
                                    "sistema_bancario");
        ContaDaoInterface dao;
        dao = new ContaDaoRelacional(conexao);
        List<Conta> todasContas;
        todasContas = dao.listarTudo();
        for (Conta c: todasContas) {
            System.out.print("Nro: " + c.getNumero());
            System.out.print(" - ");
            System.out.println("Saldo: R$ " + c.getSaldo());
        }
        conexao.close();
    }
}
```

Note que uma das vantagens de encapsular o acesso à base de dados na classe **ContaDaoRelacional** é a a nossa aplicação (neste caso, a classe **AppSelectContasComDao**) não precisa conhecer e utilizar a API JDBC.

continuação do Enunciado

Note que a exceção **SQLException** está sendo tratada, mas só imprime mensagens na tela. Desta forma, a aplicação que chama o método **listarTudo** não é informada explicitamente que ocorreu uma exceção.

```
public class ContaDaoRelacional implements ContaDaoInterface {
    // código omitido
    @Override
    public List<Conta> listarTudo() {
        List<Conta> contas = new ArrayList<>();
        try {
            // código omitido
        } catch (SQLException ex) { ex.printStackTrace(); }
        return contas;
    }
}
```

Exercício 1

Para que a nossa aplicação possa ser notificada, vamos declarar uma exceção customizada do tipo *checked* que chamaremos de **ContaDaoException**.

Escreva o código de **ContaDaoException**.

Exercício 2

Vamos alterar a interface **ContaDaoInterface** declarando que a execução do método **listarTudo** pode lançar a exceção **ContaDaoException**.

Escreva o código de **ContaDaoInterface** com esta alteração.

Exercício 3

Vamos agora alterar o método **listarTudo** da classe **ContaDaoRelacional** de forma a capturar a exceção **SQLException** e lançar a exceção **ContaDaoException**.

Escreva o código de **ContaDaoRelacional** com esta alteração.

Exercício 4

Vamos agora alterar a nossa aplicação para que ela apresente a mensagem "Falha ao listar os dados das contas" caso a exceção **ContaDaoException** seja disparada.

Escreva o código de **AppSelectContasComDao** com esta alteração.

Bom estudo!