sãojudas	FTCE Faculdade de Tecnologia e Ciências Exatas

USJT – 2019/1 - Programação Orientada a Objetos - Laboratório de Exercícios

Professores: Bonato, Hamilton e Machion

Aula: 08
Assunto: Banco de Dados

Exemplo Resolvido: Junto com seu professor, abra o código abaixo, fornecido junto com o exercício, no JGrasp; seu professor dar uma explicação geral do código, entenda-o e depois execute-o para ver os resultados.

Crie a classe Livro com três construtores (padrão, que recebe só idLivro, que recebe todos os parâmetros), métodos de acesso e modificadores e os atributos privados titulo, do tipo String, edicao, do tipo int e idLivro, do tipo int. Crie o método toString que retorna o valor dos atributos.

Crie os métodos de persistência da classe Livro:
- public void inserir(Connection conn);
- public void alterar(Connection conn);
- public void excluir(Connection conn);
- public void carregar(Connection conn);

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Livro {
    private int idLivro;
    private String titulo;
    private int edicao;

    public Livro(){

    }
    public Livro(int idLivro){
        this.idLivro = idLivro;
    }
    public Livro(int idLivro, String titulo, int edicao) {
        this.idLivro = idLivro;
        this.titulo = titulo;
        this.edicao = edicao;
    }

    public int getIdLivro() {
        return idLivro;
    }
```

```
public String getTitulo() {
    return titulo;
}

public int getEdicao() {
    return edicao;
}

public void setIdLivro(int idLivro) {
    this.idLivro = idLivro;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public void setEdicao(int edicao) {
    this.edicao = edicao;
}

public String toString() {
    return "Livro [idLivro=" + idLivro + ", titulo=" + titulo + ",
 edicao=" + edicao + "]";
}

public void incluir(Connection conn) {
    String sqlInsert =
 "INSERT INTO livro(idLivro, Titulo, Edicao) VALUES (?, ?, ?)";

    PreparedStatement stm = null;
    try {

        stm = conn.prepareStatement(sqlInsert);
        stm.setInt(1, getIdLivro());
        stm.setString(2, getTitulo());
        stm.setInt(3, getEdicao());
        stm.execute();
    } catch (Exception e) {
        e.printStackTrace();
        try {
            conn.rollback();
        } catch (SQLException e1) {
            System.out.print(e1.getStackTrace());
        }
    } finally {
        if (stm != null) {
            try {
                stm.close();
            } catch (SQLException e1) {
                System.out.print(e1.getStackTrace());
            }
        }
    }
}

public void excluir(Connection conn) {
    String sqlDelete = "DELETE FROM LIVRO WHERE idLivro = ?";
    PreparedStatement stm = null;
    try {
        stm = conn.prepareStatement(sqlDelete);
        stm.setInt(1, getIdLivro());
```

```
            stm.execute();
        } catch (Exception e) {
            e.printStackTrace();
            try {
                conn.rollback();
            } catch (SQLException e1) {
                System.out.print(e1.getStackTrace());
            }
        } finally {
            if (stm != null) {
                try {
                    stm.close();
                } catch (SQLException e1) {
                    System.out.print(e1.getStackTrace());
                }
            }
        }
    }

    public void atualizar(Connection conn) {
        String sqlUpdate =
      "UPDATE LIVRO SET Titulo = ?, Edicao = ? WHERE IdLivro = ?";
        PreparedStatement stm = null;
        try {
            stm = conn.prepareStatement(sqlUpdate);
            stm.setString(1, getTitulo());
            stm.setInt(2, getEdicao());
            stm.setInt(3, getIdLivro());

            stm.execute();
        } catch (Exception e) {
            e.printStackTrace();
            try {
                conn.rollback();
            } catch (SQLException e1) {
                System.out.print(e1.getStackTrace());
            }
        } finally {
            if (stm != null) {
                try {
                    stm.close();
                } catch (SQLException e1) {
                    System.out.print(e1.getStackTrace());
                }
            }
        }
    }

    public void carregar(Connection conn) {
        String sqlSelect =
       "SELECT Titulo, Edicao FROM LIVRO WHERE idLivro = ?";

        PreparedStatement stm = null;
        ResultSet rs = null;
        try {
            stm = conn.prepareStatement(sqlSelect);
            stm.setInt(1, getIdLivro());
            rs = stm.executeQuery();

            if (rs.next()) {
                this.setTitulo(rs.getString(1));
                this.setEdicao(rs.getInt(2));
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
            try {
                conn.rollback();
            } catch (SQLException e1) {
                System.out.print(e1.getStackTrace());
            }
        } finally {
            if (rs != null) {
                try {
                    rs.close();
                } catch (SQLException e1) {
                    System.out.print(e1.getStackTrace());
                }
            }
            if (stm != null) {
                try {
                    stm.close();
                } catch (SQLException e1) {
                    System.out.print(e1.getStackTrace());
                }
            }
        }
    }
}
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexaoBD {

  static {
    try {
      Class.forName("com.mysql.jdbc.Driver");
    }
    catch (ClassNotFoundException e) {
      throw new RuntimeException(e);
    }
  }

  public Connection conectar() throws SQLException {
    String servidor = "localhost";
    String porta = "3306";
    String database = "editora";
    String usuario = "alunos";
    String senha = "alunos";
    return DriverManager
          .getConnection("jdbc:mysql://"+servidor+":"+porta+
        "/"+database+"?user="+usuario+"&password="+senha);
  }
}
```

```
import java.sql.SQLException;
import java.sql.Connection;

public class Teste {
```

```
    /**
     * Programa principal
     */
  public static void main(String[] args) {
    Connection conn = null;
    Livro livro;

    try {
      ConexaoBD bd = new ConexaoBD();
      conn = bd.conectar();

    // *** Inclusao do Primeiro Livro ***
      livro = new Livro(1, "O Senhor dos Aneis", 32);
      livro.incluir(conn);
      System.out.println(livro);

    // *** Inclusao do Segundo Livro ***
      livro = new Livro();
      livro.setIdLivro(2);
      livro.setTitulo("1984");
      livro.setEdicao(22);
      livro.incluir(conn);
      System.out.println(livro);

    // *** Carregar o segundo livro a partir do bd ***
      livro = new Livro(2);
      System.out.println(livro);
      livro.carregar(conn);
      System.out.println(livro);
    // *** Alterar o livro 2 (carregado em livro) do bd
      livro.setTitulo("Admiravel Mundo Novo");
      livro.setEdicao(3);
      livro.atualizar(conn);
      livro = new Livro(2);
      livro.carregar(conn);
      System.out.println(livro);

    }
    catch (Exception e) {

      e.printStackTrace();
      if (conn != null) {
        try {
          conn.rollback();
        }
        catch (SQLException e1) {
          System.out.print(e1.getStackTrace());
        }
      }
    }
    finally {
      if (conn != null) {
        try {
          conn.close();
        }
        catch (SQLException e1) {
          System.out.print(e1.getStackTrace());
        }
      }
    }
  }
}
```

Problemas Propostos:

Exercícios iniciais: valor 0,5 ponto
Resolva os exercícios desta seção para conquistar 0,5 ponto

1) Crie a classe Professor com três construtores (um padrão, um que receba só matrícula, e outro que receba todos os parâmetros), métodos de acesso e modificadores e os atributos privados nome, do tipo String, idade, do tipo int e matricula, do tipo int. Crie o método toString que retorna o valor dos atributos.

Crie os métodos de persistência da classe Professor:
- `public void inserir(Connection conn);`
- `public void alterar(Connection conn);`
- `public void excluir(Connection conn);`
- `public void carregar(Connection conn);`

Exercícios intermediários: valor 0,5 ponto
Resolva os exercícios desta seção para conquistar mais 0,5 ponto

2) Crie a classe Disciplina com três construtores (um padrão, um que receba só codigo, e outro que receba todos os parâmetros), métodos de acesso e modificadores e os atributos privados nome, do tipo String, professores, do tipo ArrayList<Professor>, codigo, do tipo String. Crie o método toString que retorna o valor dos atributos.

Crie os métodos de persistência da classe Disciplina:
- `public void inserir(Connection conn);`
- `public void alterar(Connection conn);`
- `public void excluir(Connection conn);`
- `public void carregar(Connection conn);`

Exercícios complementares (para praticar)
Resolva os exercícios desta seção para aprimorar seus conhecimentos

3) Crie as classes Cliente, ContaCorrente e Agencia conforme abaixo (ou use as que fez na aula 06):

a) A classe Cliente possui os atributos nome e cpf, ambos do tipo String, e um atributo conta do tipo ContaCorrente. Crie um construtor que recebe os atributos como parâmetros e os métodos de acesso e os modificadores.

b) A classe ContaCorrente tem os atributos numero e digito, ambos inteiros, o atributo agencia do tipo Agencia e o atributo saldo do tipo double. Crie um construtor que recebe os atributos como parâmetros e os métodos de acesso e os modificadores. Crie também um método depositar que receba um parâmetro double com o valor do depósito e aumente o saldo da conta. Crie também um método sacar que receba um parâmetro double com o valor do saque e diminua o saldo da conta. A conta não pode ficar negativa. Neste caso, deve ser dada uma mensagem que o

saque não foi efetuado e o retorno deve ser zero. Caso contrário o retorno deve ser o valor sacado. Crie também um método consultarSaldo que não recebe parâmetros e retorne o saldo. Crie, finalmente, um método imprimirSaldo que imprima o numero da conta corrente com dígito, o número da agência com dígito e o saldo da conta corrente.

c) Ainda na classe ContaCorrente, o numero da conta deve ter no máximo 4 dígitos e ser positivo. O digito da conta deve ser validado a partir do seguinte algoritmo de módulo 11: multiplique o primeiro dígito da conta por 4, o segundo por 6, o terceiro por 8 e o quarto por 2; some tudo e calcule o resto da divisão (módulo) da soma por 11. Este é o valor do dígito. Obs: se o resultado for 10 o dígito é 0.

d) A classe Agencia tem os atributos nome do tipo String, numero e digito do tipo int. Crie um construtor que recebe os atributos como par,metros e os métodos de acesso e os modificadores. O numero e o digito da Agencia devem seguir os mesmos padrões do número e do dígito da conta corrente.

e) Em cada uma das três classes crie os métodos CRUD de persistência. O método atualizar sempre deve atualizar todos os campos da tabela exceto a chave primária (PK). Não persista todos os campos. Vamos fazer isso na próxima aula. Faça conforme abaixo:

Cliente: cpf (pk), nome
Conta Corrente: numero(pk), digito
Agencia: numero(pk), digito

Bibliografia

LOPES, ANITA. GARCIA, GUTO. Introdução à Programação: 500 algoritmos resolvidos. Rio de Janeiro: Elsevier, 2002.

DEITEL, P. DEITEL, H. Java: como programar. 8 Ed. São Paulo: Prentice – Hall (Pearson), 2010.