Universidade Federal do Ceará
Centro de Ciências/Departamento de Computação
Código da Disciplina: CK0084
Professor: Ismayle de Sousa Santos

Sistemas de Informações e Banco de Dados

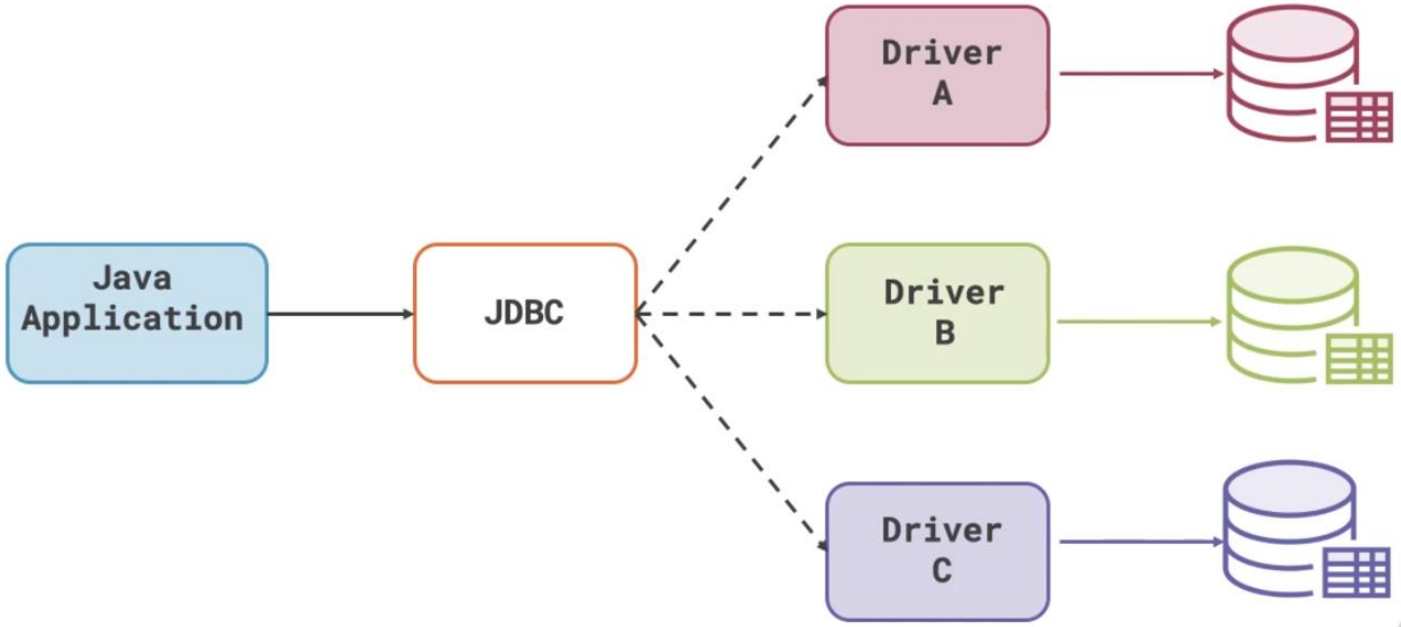
Banco de Dados - JDBC

 rfbrkh3  ismaylesantos@great.ufc.br  @IsmayleSantos

Agora vamos ver como combinar Java e SQLITE3

JDBC

- JDBC (Java Database Connectivity)
  - É uma API (application programming interface) para uma aplicação java se conectar a um banco de dados



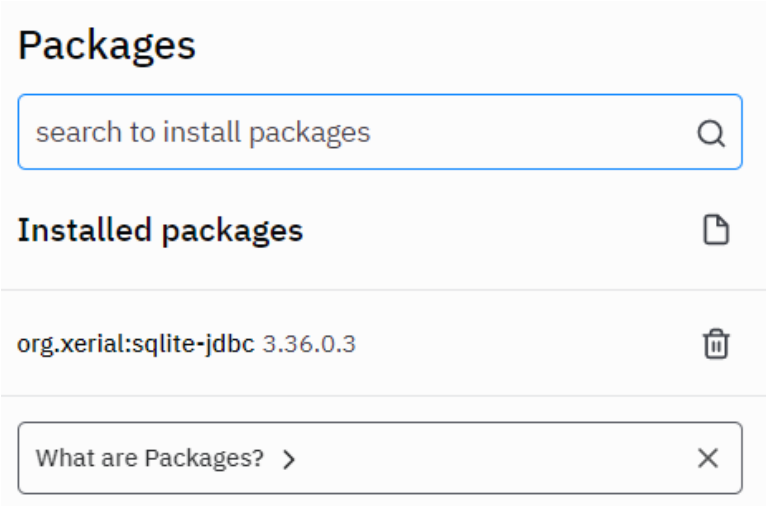
JDBC

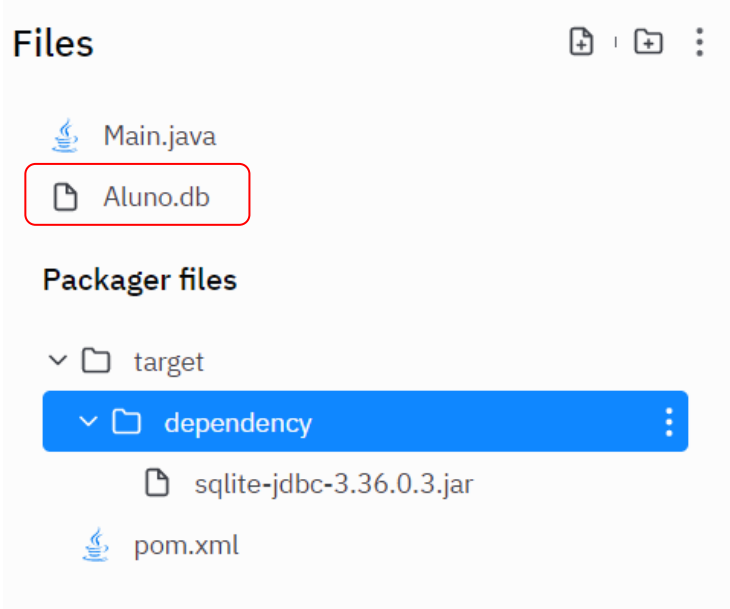
- Configurando JDBC no Eclipse
  - https://github.com/xerial/sqlite-jdbc/releases
  - Passo 1: Fazer o download do sqlite-jdbc-3.36.0.1.jar
  - Passo 2: Adicionar o arquivo .jar (Java ARchive) no Class Path
    - Eclipse: Botão direito em cima do projeto -> Build Path -> configure Build Path -> Aba Libraries -> Clicar em Adicionar External JAR

Define onde as Classes definidas pelo usuário estão localizados

JDBC

- Configurando JDBC no REPLIT
  - Passo 1: Clicar em Packages -> buscar por “sqlite-jdbc” -> instalar opção org.xerial:sqlite-jdsbc 3.36.0.3 (ou versão superior)
  - Passo 2: Fazer upload do arquivo do BD





Pacote java.sql

- Fornece a API para acesso e processamento de dados
- Principais classes e interfaces:
  - DriverManager, responsável por criar uma conexão com o banco de dados
  - Connection, classe responsável por manter uma conexão aberta com o banco
  - Statement, gerencia e executa instruções SQL

Pacote java.sql

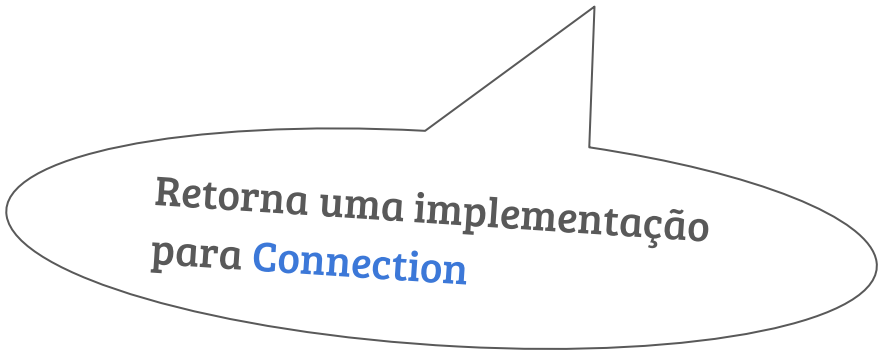
- PreparedStatement, gerencia e executa instruções SQL, permitindo a passagem de parâmetros em uma instrução
- ResultSet, responsável por receber os dados obtidos em uma pesquisa ao banco

DriverManager

- Responsável pelo gerenciamento de drivers JDBC
- Estabelece conexões a bancos de dados
  - Tentando estabelecer conexão com o Banco de Dados

```
String url = "jdbc:sqlite:C:/Aluno.db";

Connection conn = DriverManager.getConnection(url);
```



Connection

- Representa a conexão com o banco de dados
- Proporcionar informações sobre as tabelas do banco através de transações
- Métodos desta interface freqüentemente utilizados:
  - commit(), executa todas as alterações feitas com o banco de dados pela atual transação
  - rollback(), desfaz qualquer alteração feita com o banco de dados pela atual transação
  - close(), libera o recurso que estava sendo utilizado pelo objeto

Statement

- Implementação de uma Interface que fornece métodos para executar uma instrução SQL
- Não aceita a passagem de parâmetros
- Principais métodos da Interface Statement são:
  - executeUpdate(), executa instruções SQL do tipo: INSERT, UPDATE e DELETE
  - execute(), executa instruções SQL de busca de dados do tipo SELECT
  - close(), libera o recurso que estava sendo utilizado pelo objeto

Statement

- **Instanciando o objeto statement** (stmt)
  - Statement stmt = conn.createStatement();
- **Executando uma instrução SQL**
  - stmt.executeUpdate("INSERT INTO ALUNO VALUES (1, 'Pedro da Silva')");

Exemplo com Statement

```
public class Main {
    public static void main(String[] args) {
        Connection conn = null;
        try {
            String url = "jdbc:sqlite:C:/Aluno.db";
            conn = DriverManager.getConnection(url);
            Statement statement = conn.createStatement();
            statement.executeUpdate("insert into Aluno values(1,'leo')");
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException ex) {
                System.out.println(ex.getMessage());
            }
        }
    }
}
```

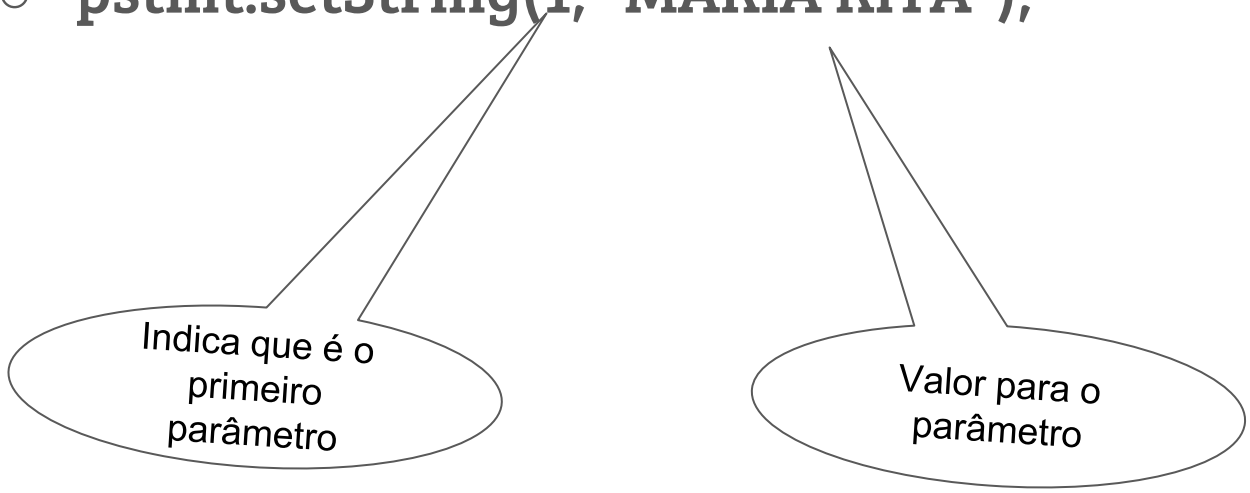
O bloco try-catch-finally é para tratar exceções

PreparedStatement

- A interface PreparedStatement possui todos os recursos da interface Statement
- Acrescentando a utilização de parâmetros em uma instrução SQL
- Métodos da interface PreparedStatement são:
  - execute(), consolida a instrução SQL informada
  - setDate(), método utilizado para atribuir um valor do tipo Data
  - setInt(), utilizado para atribuir valores do tipo inteiro
  - setString(), método utilizado para atribuir valores do tipo Alfa Numéricos

PreparedStatement

- **Instanciando o objeto preparedStatement** (pstmt)
  - PreparedStatement pstmt = conn.prepareStatement("UPDATE ALUNO SET NOME = ?");
- **Configurando o valor ao parâmetro**
  - pstmt.setString(1, “MARIA RITA”);



PreparedStatement

- Outro Exemplo

```
public class Aluno {

    public void adicionaAluno(Connection conn, int id, String nome)
            throws SQLException {
        PreparedStatement prepStmt = conn.prepareStatement("insert "
                + "into Aluno values(?,?)");
        prepStmt.setInt(1, id);
        prepStmt.setString(2, nome);
        prepStmt.executeUpdate();
    }
}
```

ResultSet

- Esta interface permite o recebimento e gerenciamento do conjunto de dados resultante de uma consulta SQL
- Métodos capazes de acessar os dados
- Métodos desta interface freqüentemente utilizados:
  - next(), move o cursor para a próxima linha de dados, já que o conjunto de dados retornados pela consulta SQL é armazenado como em uma lista
  - close(), libera o recurso que estava sendo utilizado pelo objeto
  - getString(String columnName), recupera o valor da coluna informada como parâmetro, da linha atual do conjunto de dados recebidos pelo objeto ResultSet

ResultSet

- Recebendo o conjunto de dados da consulta SQL
  - ResultSet rs = stmt.executeQuery("SELECT * FROM ALUNO");
- Se houver resultados, posiciona-se o cursor na próxima linha de dados
  - Métodos como o getInt(), getString() para recuperar os valores

```
ResultSet rs = statement.executeQuery("Select * from Aluno");
while(rs.next()){
  int id = rs.getInt("id");
  String nome = rs.getString("nome");
  System.out.println("Id: "+id+" nome: "+nome);
}
```

JDBC

Vamos para Prática!!

Obrigado!

Por hoje é só pessoal...

Dúvidas?


rfbrkh3


ismaylesantos@great.ufc.br


@IsmayleSantos