Programação Orientada a Objetos

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

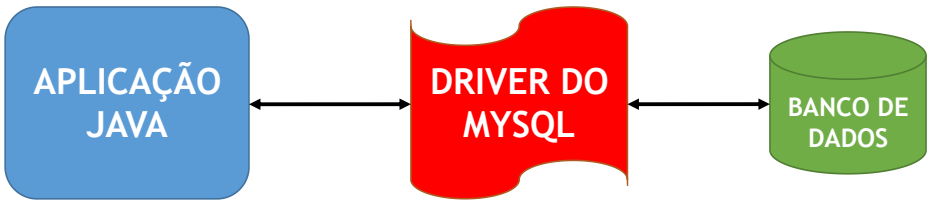
Sumário

Esquema de funcionamento

Objetivo: criar aplicações Java envolvendo banco de dados. O sistema deverá ser capaz de manipular dados armazenados no banco de dados.

Componentes do projeto: projeto Java com API SQL, *driver* do SGBD (MySQL), linguagem SQL para "conversar" com o banco de dados.

Esquema:



Esquema de funcionamento

- Iremos utilizar banco de dados relacionais;
- Existem vários Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBD-R): Oracle, MS SQL Server, Sybase, IBM DB2, PostgreSQL e MySQL.

	colunas				
linhas	**cod**	**titulo**	**tipo**	**ano_lancamento**	**fabricante**
	1	Dota 2	MOBA	2013	Valve
	2	League of Legends	MOBA	2009	Riot
	3	Valorant	FPS	2020	Riot

Comandos SQL (banco de dados)

SQL - *Structured Query Language*

a) **DDL** (Data **Definition** Language – Linguagem de **Definição** de Dados)

b) **DML** (Data **Manipulation** Language – Linguagem de **Manipulação** de Dados)

c) **DCL** (Data **Control** Language – Linguagem de **Controle** de Dados)

Comandos SQL (banco de dados)

Comandos DDL

SENTENÇA SQL	SINTAXE
CREATE DATABASE	CREATE DATABASE nome_bd
CREATE TABLE	CREATE TABLE nome_tabela (nome_coluna1 tipo_dado, nome_coluna2 tipo_dado, nome_coluna3 tipo_dado, ...)

Comandos SQL (banco de dados)

Comandos DDL

SENTENÇA SQL	SINTAXE
DROP DATABASE	DROP DATABASE nome_bd
DROP TABLE	DROP TABLE nome_tabela

Comandos SQL (banco de dados)

Comandos DDL

SENTENÇA SQL	SINTAXE
ALTER TABLE	ALTER TABLE nome_tabela ADD nome_coluna ALTER TABLE nome_tabela DROP COLUMN nome_coluna

Comandos SQL (banco de dados)

Comandos DML

SENTENÇA SQL	SINTAXE
SELECT	SELECT nome_coluna(s) FROM nome_tabela WHERE nome_coluna = / > / < / IN / LIKE ...
INSERT INTO	INSERT INTO nome_tabela VALUES (valor1, valor2, valor3, ...) ou INSERT INTO nome_tabela (coluna1, coluna2, coluna3, ...) VALUES (valor1, valor2, valor3, ...)

Comandos SQL (banco de dados)

Comandos DML

SENTENÇA SQL	SINTAXE
UPDATE	UPDATE nome_tabela SET coluna1 = valor, coluna2 = valor, ... WHERE alguma_coluna = algum_valor
DELETE	DELETE FROM nome_tabela WHERE alguma_coluna = algum_valor ou DELETE FROM nome_tabela (Obs: Deleta todos os registros da tabela)

Comandos SQL (banco de dados)

Comandos DCL

SENTENÇA SQL	SINTAXE
GRANT	**Autorizar o usuário a executar ou setar operações** GRANT nome_privilégio ON nome_objeto TO {nome_usuário |PUBLIC |nome_papel} [WITH GRANT OPTION];
REVOKE	**Restringir ou remover acesso de executar operações** REVOKE nome_privilégio ON nome_objeto FROM {nome_usuário |PUBLIC |nome_papel}

Comandos SQL (banco de dados)

Funções de Agregação

FUNÇÃO	DESCRIÇÃO
MIN( )	Pega o valor **mínimo** de um campo/coluna
MAX( )	Pega o valor **máximo** de um campo/coluna
COUNT( )	Pega o **número de registros** de uma consulta (select)
SUM( )	Pega a **soma** (*sum*) de todos os valores de um campo/coluna
AVG( )	Pega a **média** (*average*) de todos os valores de um campo/coluna

Comandos SQL (banco de dados)

Banco de Dados - Mais detalhes...

http://www.w3schools.com/sql/sql_quickref.asp
http://www.w3schools.com/sql/sql_join.asp

Criação do banco de dados

IMPORTANTE: Antes de tudo, certifique-se que você possui o MySQL 8 e o MySQL Workbench instalado no seu computador.
Se quiser saber como instalar, acesse:
https://www.youtube.com/watch?v=fmerTu7dWk8

Dentro do MySQL Workbench, executar os comandos:
create database bd_jogos;
use bd_jogos;

Criação da tabela

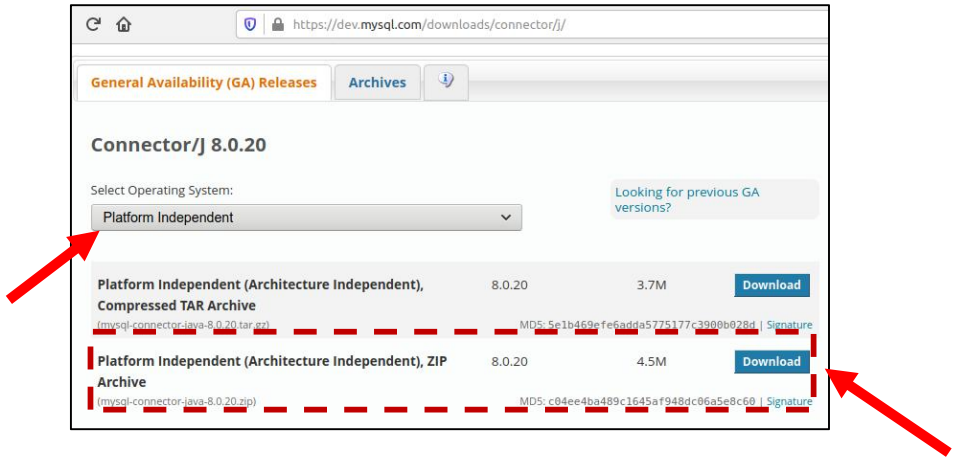
	colunas				
linhas	**cod**	**titulo**	**tipo**	**ano_lancamento**	**fabricante**
	1	Dota 2	MOBA	2013	Valve
	2	League of Legends	MOBA	2009	Riot
	3	Valorant	FPS	2020	Riot

```
create table tab_jogo (
   id int not null primary key auto_increment,
   titulo varchar(100) not null,
   tipo varchar(30) not null,
   ano_lancamento int not null,
   fabricante varchar(100) not null );
```

Instalação do driver do SGBD

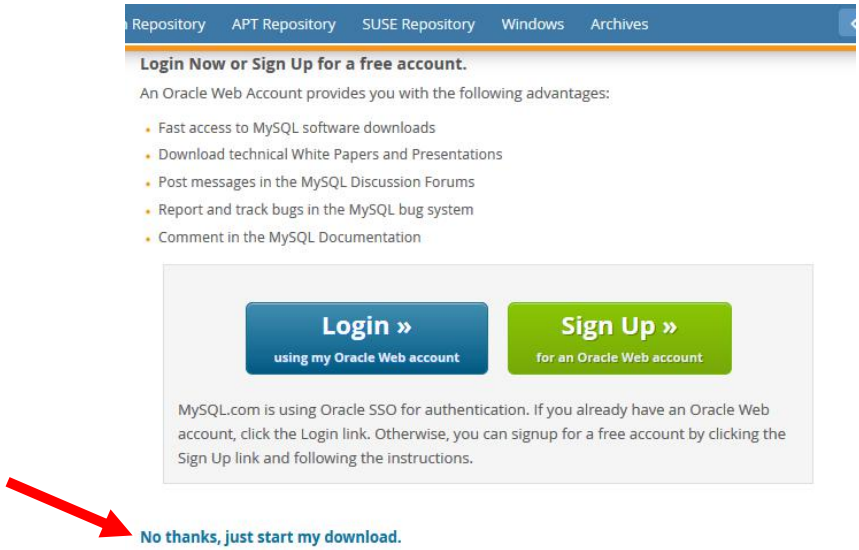
Baixar *driver* do MySQL (1/3):

http://dev.mysql.com/downloads/connector/j/



Instalação do driver do SGBD

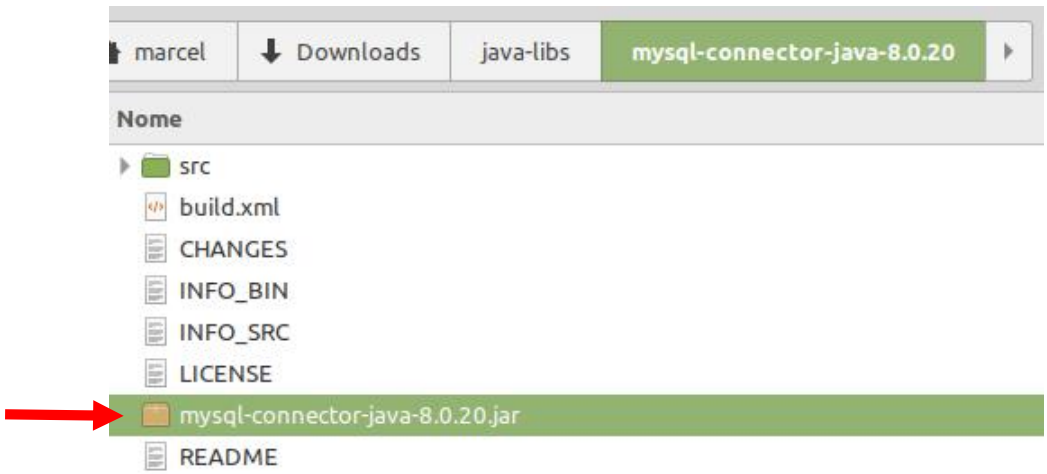
Baixar *driver* do MySQL (2/3):



Instalação do driver do SGBD

Deszipar o arquivo zip e pegar o arquivo (3/3)

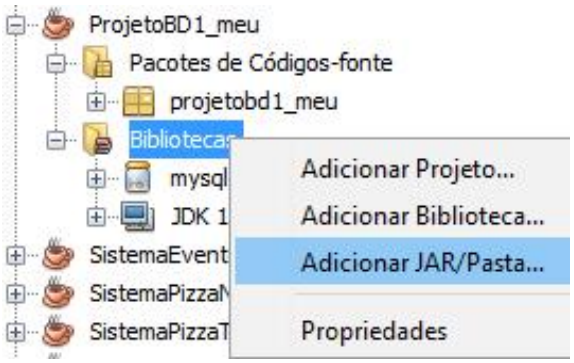
mysql-connector-java-8.0.20.jar



Instalação do driver do SGBD

Criando o projeto no Netbeans:
Nome: ProjetoJogosBD

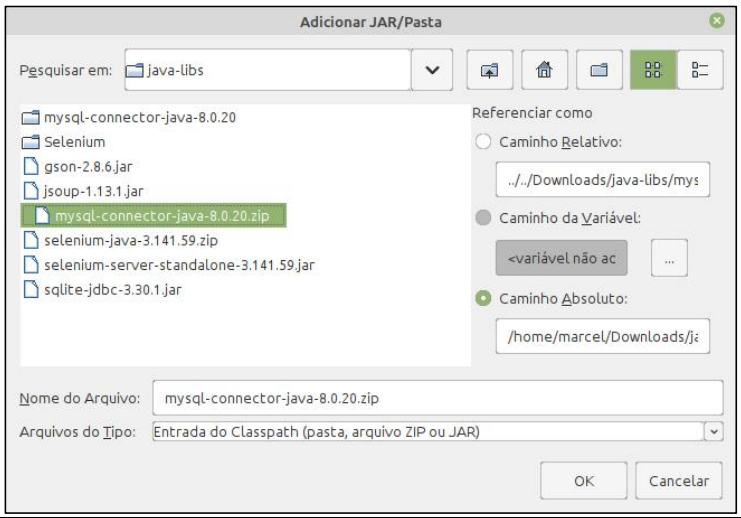
Colocar o driver no projeto:
Clique com o botão direito do mouse na pasta "Bibliotecas" do projeto e escolha a opção "Adicionar JAR/Pasta..."



Instalação do driver do SGBD

Apontar para o arquivo (baixado e deszipado do site):

mysql-connector-java-8.0.20.jar



Criação da classe de BD

```
importações...
public class BancoDados {
  Connection conexao = null;
  public Connection conectar() {
    try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      String url = "jdbc:mysql://localhost/bd_jogos", usuario = "root", senha = "";
      conexao = DriverManager.getConnection(url, usuario, senha);
    } catch ( ClassNotFoundException e ) {
      System.out.println("Erro! Classe não encontrada. Detalhes: " + e.getMessage());
    } catch ( SQLException e ) {
      System.out.println("Erro! Problema de SQL. Detalhes: " + e.getMessage());
    }
    return conexao;
  }
}
```

Criação da classe de conexão com o BD

```
importações...
public class BancoDados {
  public boolean isConectado() {
      try {
          if (conexao != null && ! conexao.isClosed())
              return true;
      } catch (SQLException e) {
          System.out.println("Erro ao verificar conexão. Detalhes: " +
                e.getMessage());
      }
      return false;
   }
}
```

Criação da classe de conexão com o BD

```
importações...
public class BancoDados {
  public void desconectar() {
      try {
          conexao.close();
      } catch (SQLException e) {
          System.out.println("Erro ao fechar conexão. Detalhes: " +
              e.getMessage());
      }
  }
}
```

Criação da classe referente à tabela

```
public class Jogo {
    private int id;
    private String titulo;
    private String tipo;
    private int anoLancamento;
    private String fabricante;

    // construtores...
    // sets e gets...

}
```

Criação da classe referente à tabela

```
public class Jogo {
   public void inserir() {
      BancoDados bd = new BancoDados();
      Connection con = bd.conectar();
      PreparedStatement inserir = null;
      try {
         inserir = con.prepareStatement("INSERT INTO tab_jogo (titulo,
tipo, ano_lancamento, fabricante) VALUES (?, ?, ?, ?)");
         inserir.setString(1, this.titulo);
         inserir.setString(2, this.tipo);
         inserir.setInt(3, this.anoLancamento);
         inserir.setString(4, this.fabricante);
         inserir.executeUpdate();
      }  // continua...
```

Criação da classe referente à tabela

```
public class Jogo {
   public void inserir() {
      // código anterior...
      } catch(SQLException e) {
         System.out.println("Erro ao inserir jogo. Detalhes: " + e.getMessage());
      } finally {
         try {
            if(inserir != null) inserir.close();
         } catch (SQLException e) {
            System.out.println("Erro ao fechar insert. Detalhes: " + e.getMessage());
         }
         bd.desconectar();
      }
   } // fim inserir
}
```

Criação da classe referente à tabela

```
public class Jogo {
    public static void listar() {
        BancoDados bd = new BancoDados();
        Connection con = bd.conectar();
        Statement consulta = null;
        ResultSet rs = null;
        try {
            consulta = con.createStatement();
            rs = consulta.executeQuery("SELECT * FROM tab_jogo");
            int num = 0;
            System.out.println("Jogos cadastrados:");
            // continua...
```

Criação da classe referente à tabela

```
public class Jogo {
    public static void listar() {
            // continua...
            while(rs.next()) {
                System.out.printf("%d %s", rs.getInt("id"), "|");
                System.out.printf("%-20s %s", rs.getString("titulo"), "|");
                System.out.printf("%-10s %s", rs.getString("tipo"), "|");
                System.out.printf("%-3s %s", rs.getInt("ano_lancamento"), "|");
                System.out.printf("%-10s %s", rs.getString("fabricante"), "|\n");
                num++;
            }
            System.out.println("Quantidade de jogos: " + num);
        } // fim do try
         // continua...
```

Criação da classe referente à tabela

```
public class Jogo {
   public static void listar() {
      // código anterior...
      catch(SQLException e) {
         System.out.println("Erro ao listar jogos. Detalhes: " + e.getMessage());
      } finally {
         try {
            if(consulta != null) consulta.close();
            if(rs != null) rs.close();
         } catch (SQLException e) {
            System.out.println("Erro ao fechar select. Detalhes: " + e.getMessage());
         }
         bd.desconectar();
      }
   } // fim listar
```

Criação da classe referente à tabela

```
public class Jogo {
   public static void alterar(Jogo jogo) {
      BancoDados bd = new BancoDados();
      Connection con = bd.conectar();
      PreparedStatement alterar = null;
      try {
         alterar = con.prepareStatement("UPDATE tab_jogo SET titulo = ?,
tipo = ?, ano_lancamento = ?, fabricante = ?  WHERE id = ?");
         alterar.setString(1, jogo.getTitulo());
         alterar.setString(2, jogo.getTipo());
         alterar.setInt(3, jogo.getAnoLancamento());
         alterar.setString(4, jogo.getFabricante());
         alterar.setInt(5, jogo.getId());
         alterar.executeUpdate();
      } // fim do try, continua...
```

Criação da classe referente à tabela

```
public class Jogo {
  public static void alterar(Jogo jogo) {
    // código anterior...
    catch(SQLException e) {
         System.out.println("Erro ao alterar jogo. Detalhes: " + e.getMessage());
    } finally {
       try {
          if(alterar != null) alterar.close();
        } catch (SQLException e) {
          System.out.println("Erro ao fechar update. Detalhes: " + e.getMessage());
        }
          bd.desconectar();
      }
  } // fim do alterar
```

Criação da classe referente à tabela

```
public class Jogo {
    public static void remover(int id) {
        BancoDados bd = new BancoDados();
        Connection con = bd.conectar();
        PreparedStatement remover = null;
        try {
            remover = con.prepareStatement("DELETE FROM tab_jogo
WHERE id = ?");
            remover.setInt(1, id);
            remover.executeUpdate();
        } // fim do try
```

Criação da classe referente à tabela

```
public class Jogo {
   public static void remover(int id) {
      // código anterior
     catch(SQLException e) {
         System.out.println("Erro ao remover jogo. Detalhes: " + e.getMessage());
      } finally {
        try {
           if(remover != null) remover.close();
        } catch (SQLException e) {
          System.out.println("Erro ao fechar delete. Detalhes: " + e.getMessage());
        }
        bd.desconectar();
      }
   } // fim do remover
```

Testes no método “main”

```
public static void main(String[] args) {
      Jogo j1 = new Jogo(1, "Dota 2", "MOBA", 2013, "Valve");
      Jogo j2 = new Jogo(2, "League of Lengends", "MOBA", 2009, "Riot");
      Jogo j3 = new Jogo(3, "Valorant", "FPS", 2013, "Valve");
      j1.inserir();
      j2.inserir();
      j3.inserir();
     Jogo.listar();
     // continua...
}
```

Testes no método “main”

```
public static void main(String[] args) {
        j3.setFabricante(“Riot”);
        Jogo.atualizar(j3);
        Jogo.listar();
       // continua...
}
```

Testes no método “main”

```
public static void main(String[] args) {
        Jogo.remover(2);
        Jogo.listar();
}
```

Refatorar/melhorar o projeto

- Analisar cada classe do projeto e verificar o que pode ser melhorado;
- A cada melhoria, realizar novos testes para saber se a aplicação continua funcionando;
- Siga o professor.

Tarefas

1) Realizar um laço de repetição e solicitar ao usuário que informe, através do teclado, os dados dos jogos que deseja cadastrar. Crie uma classe Util com métodos estáticos de “entrada” e “saida” que fazem uso da classe JOptionPane.

2) Crie uma forma do usuário alterar, remover e selecionar os dados através dos métodos estáticos “entrada” e “saida”.

3) Modificar todas as impressões de dados contidos em todas as classes para o método “saida” da classe Util criada no exercício 1.

4) Implementar regras de validação nos métodos de configuração “set”.