CRUD e SGBD

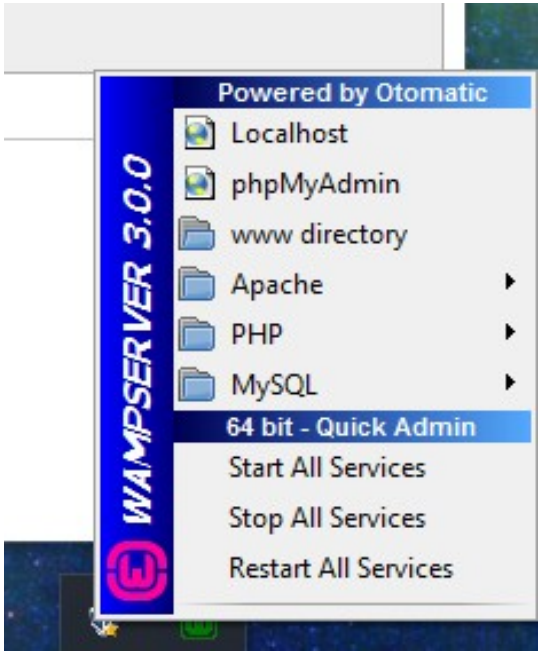
Prof. Dr. João Paulo Lemos Escola

Tópicos da aula

- Nesta aula vamos aprender a acessar um servidor de banco de dados a partir de uma classe Java;
- Vamos utilizar o SGBD MySQL instalado na máquina atual, utilizando o WampServer;
- As classes de banco de dados serão criadas no pacote DAO;
- Antes de codificar as DAOs, vamos criar um banco de dados e algumas tabelas para testes.

PhpMyAdmin

- Cliente MySQL que permite manipular bases de dados;
- Este cliente é instalado automaticamente junto com o WampServer;



Criando a base de dados

- Vamos criar uma base de dados e uma tabela para iniciar nossos testes;
- A base de dados “Loja” vai armazenar todos os dados da aplicação;
- A tabela Pessoas será a primeira de muitas que utilizaremos para armazenar dados na aplicação.

```
1 # cria a base de dados
2 create database Loja;
3
4 # ativa a base de dados
5 use Loja;
6
7 # cria a tabela
8 create table Pessoas(
9         codigo int primary key auto_increment,
10        nome varchar(200),
11        cpf char(11)
12 );
13
14 # insere um registro
15 insert into Pessoas values(0,'José', '11122233344');
```

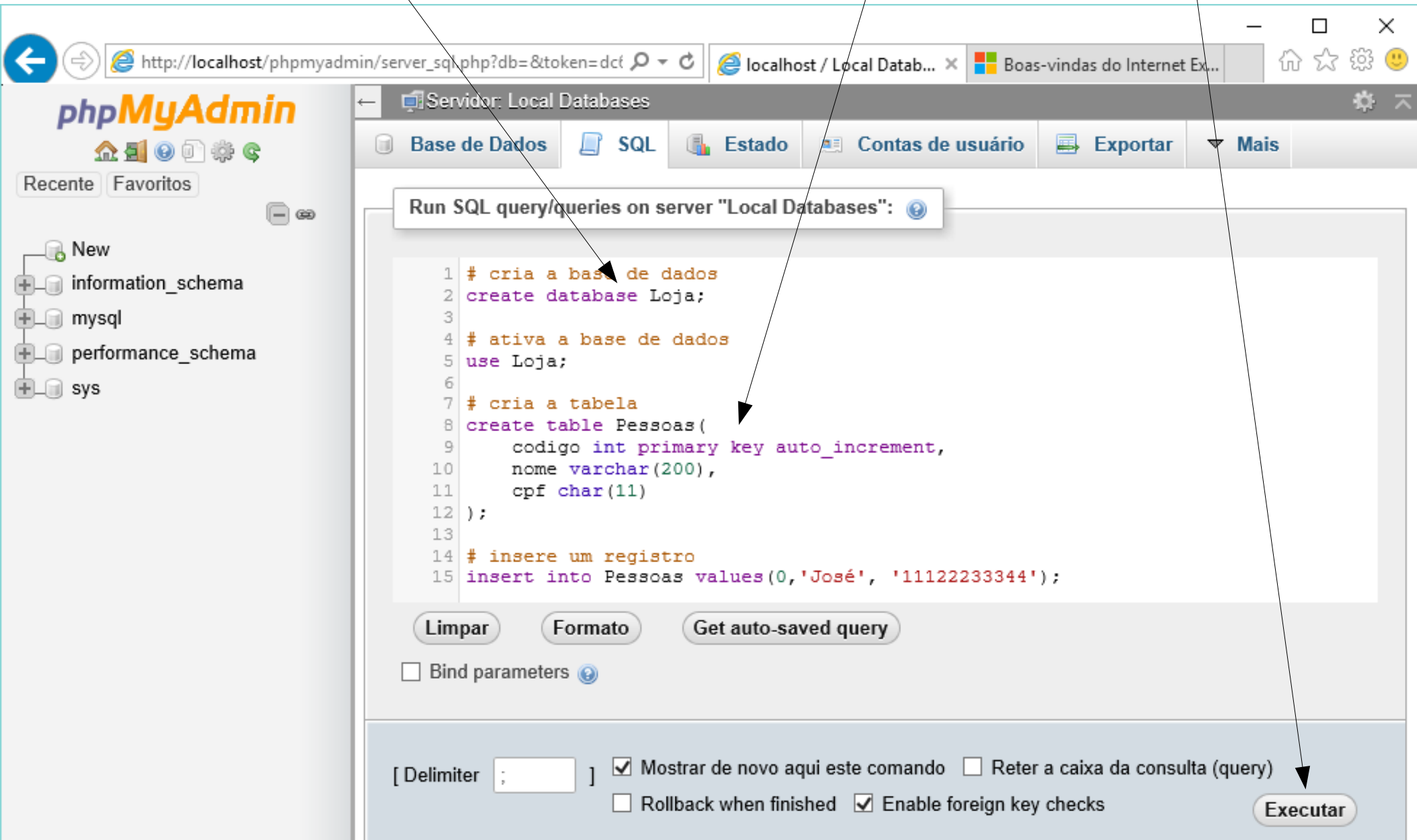
phpMyAdmin

- Logue com o usuário padrão: root sem senha:

Clique na aba SQL

Insira o código SQL

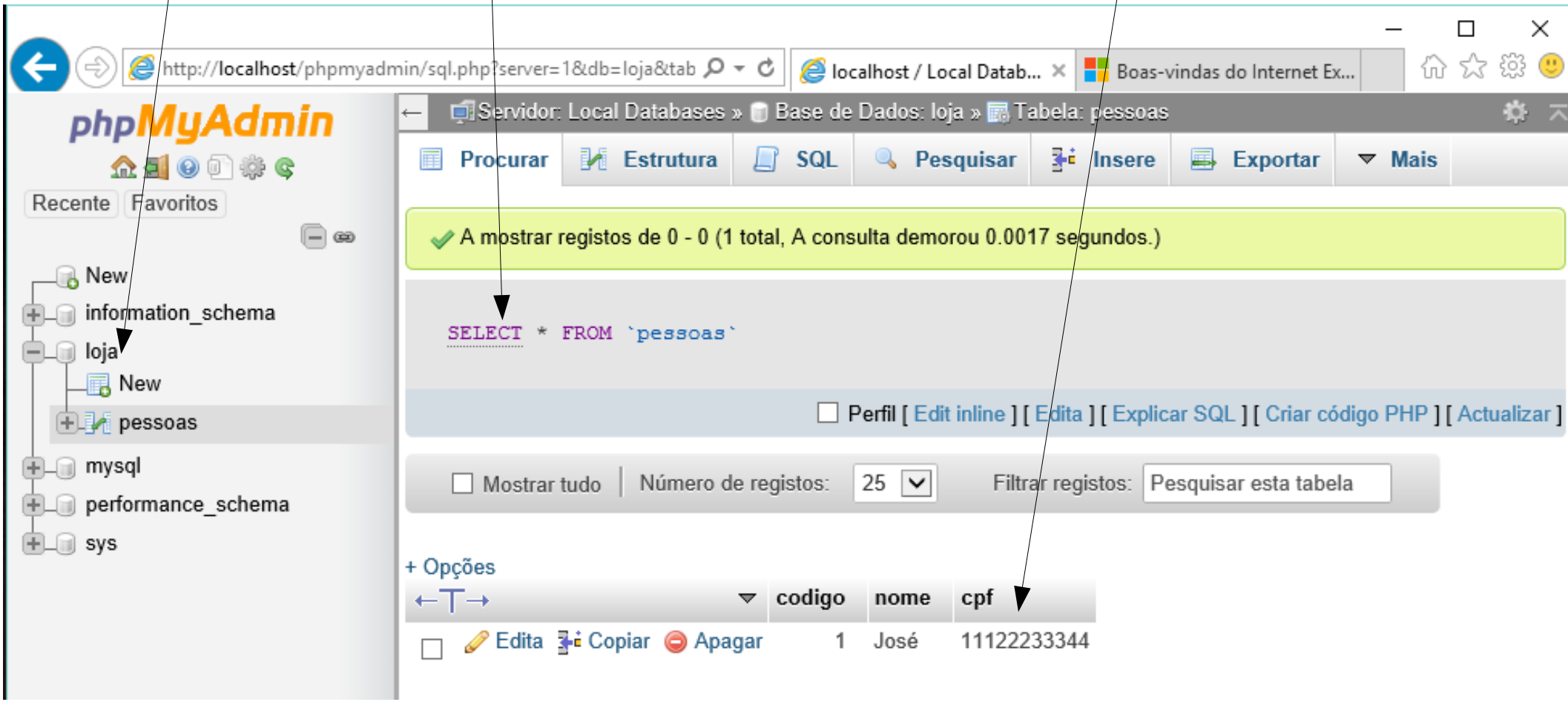
Execute



Base de dados (loja)

Comando para listar o conteúdo da tabela

Listagem

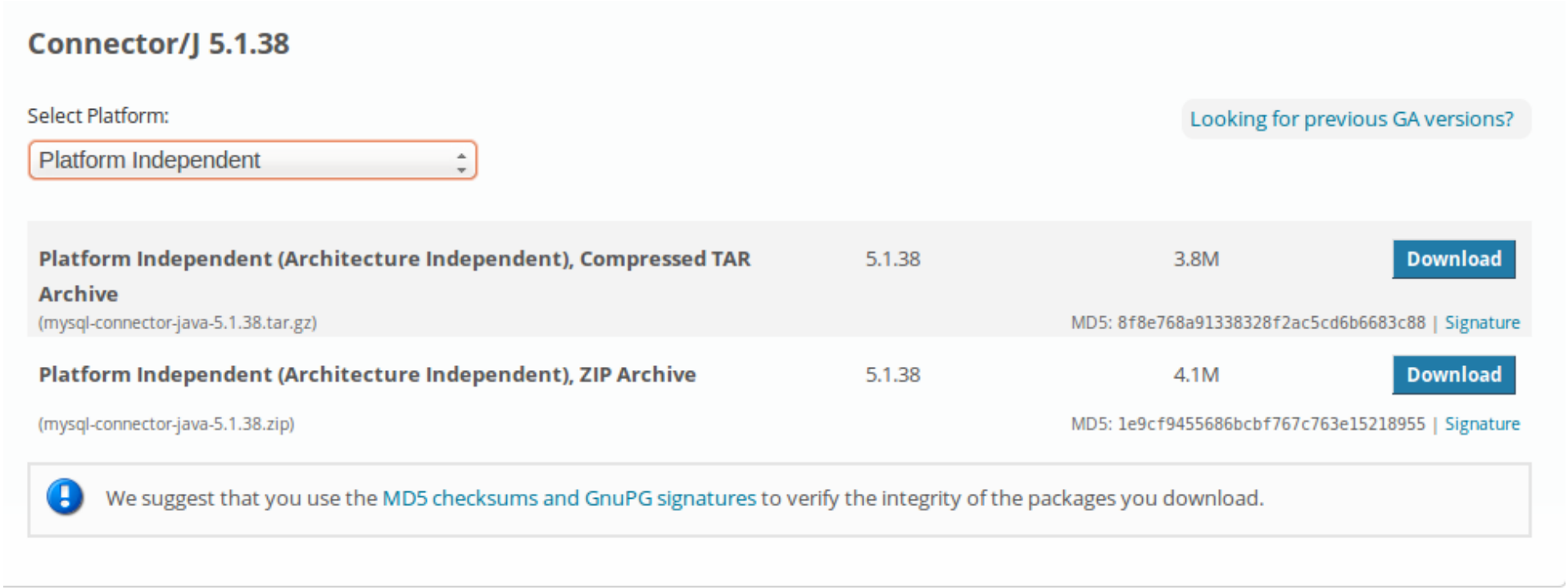


MySQL Connector

- Permite a conexão entre a classe Java e o SGBD;
- Pode ser baixado no site gratuitamente;
- Necessário incluir nas bibliotecas do projeto.

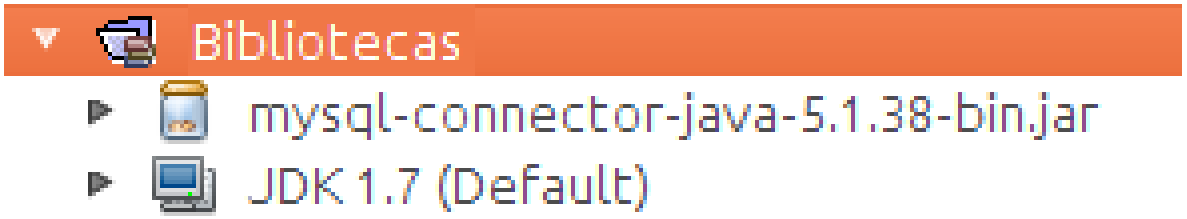
MySQL Connector (cont.)

- Baixe a versão Java:



MySQL Connector (cont.)

- Inclua no projeto: botão direito do mouse sobre a pasta “Bibliotecas” > Adicionar JAR



Classe Conexão

- Vamos criar uma classe que vai ser herdada pelas DAOs:

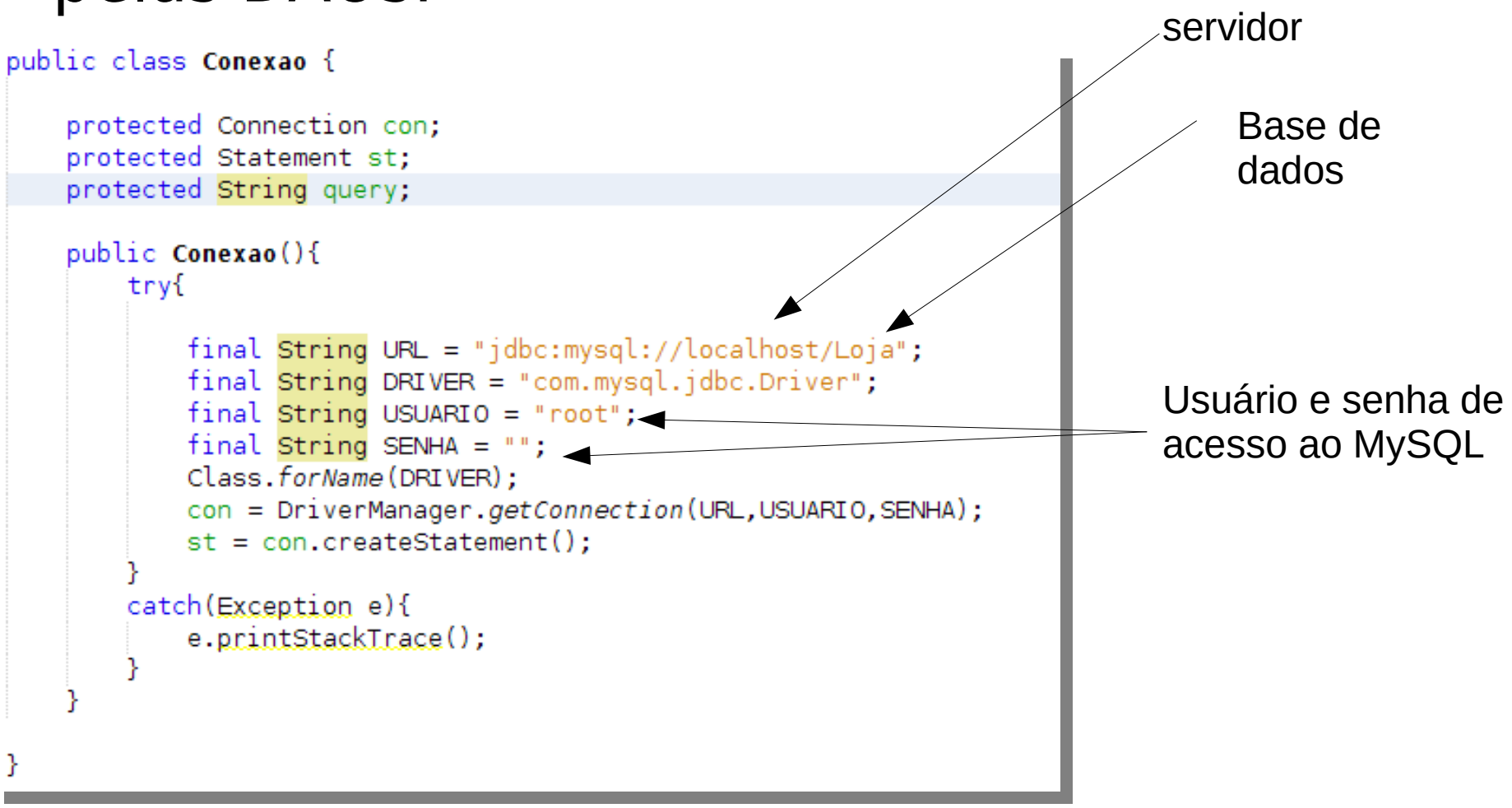
```
public class Conexao {

    protected Connection con;
    protected Statement st;
    protected String query;

    public Conexao(){
        try{

            final String URL = "jdbc:mysql://localhost/Loja";
            final String DRIVER = "com.mysql.jdbc.Driver";
            final String USUARIO = "root";
            final String SENHA = "";
            Class.forName(DRIVER);
            con = DriverManager.getConnection(URL,USUARIO,SENHA);
            st = con.createStatement();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

}
```



PessoaDao

- Vamos ajustar a classe PessoaDao para herdar de Conexão;
- Além disso, vamos implementar o código necessário para fazer o CRUD (Create, Read, Update, Delete) na tabela Pessoas.

Método inserir

```
public class PessoaDao extends Conexao {

    private PreparedStatement ps=null;
    private ResultSet rs=null;

    public boolean inserir(Pessoa p){
        try{
            query = "insert into Pessoas values(0,?,?)";
            ps = con.prepareStatement(query);
            ps.setString(1, p.getNome());
            ps.setString(2, p.getCpf());

            if (ps.executeUpdate()>0)
                return true;
        }
        catch (Exception e){
            e.printStackTrace();
        }
        return false;
    }
```

Método excluir

```
public boolean excluir(Pessoa p){
    try{
        query = "delete from Pessoas where codigo=?";
        ps = con.prepareStatement(query);
        ps.setInt(1, p.getCodigo());

        if (ps.executeUpdate()>0)
            return true;
    }
    catch (Exception e){
        e.printStackTrace();
    }
    return false;
}
```

Método alterar

```
public boolean alterar(Pessoa p){
    try{
        query = "update Pessoas set nome=?, cpf=? where codigo=?";
        ps = con.prepareStatement(query);
        ps.setString(1, p.getNome());
        ps.setString(2, p.getCpf());
        ps.setInt(2, p.getCodigo());

        if (ps.executeUpdate()>0)
            return true;
    }
    catch (Exception e){
        e.printStackTrace();
    }
    return false;
}
```

Método buscar

```
public List<Pessoa> buscar(String nome){
    List<Pessoa> lista = new ArrayList<Pessoa>();
    try{
        query = "select * from Pessoas where nome LIKE CONCAT('%', ? , '%') order by nome";
        ps = con.prepareStatement(query);
        ps.setString(1, nome);
        rs = ps.executeQuery();

        while (rs.next()){
            lista.add(
                new Pessoa(
                    rs.getInt("codigo"),
                    rs.getString("nome"),
                    rs.getString("cpf")
            ));
        }

    }
    catch (Exception e){
        e.printStackTrace();
        lista = null;
    }
    return lista;
}
```

A11ex01.jar

- Crie um projeto que utilize as classes da aula atual, fazendo CRUD na tabela Pessoas;
- Utilize como base o projeto da aula anterior (MVC).

A11ex02.jar

- Copie o projeto anterior e ajuste a classe PessoaDao para listar pessoas utilizando o CPF e código como parâmetro.

A11ex03.jar

- Copie o projeto anterior e ajuste-o para permitir CRUD de Fabricante, com os campos código e nome.

A11ex04.jar

- Copie o projeto anterior e ajuste-o para permitir CRUD de Vendedor, herdando de Pessoa, com os campos login, senha, foto e ultimoLogin.

A11ex05.jar

- Copie o projeto anterior e ajuste-o para permitir CRUD de Produto, com os campos codigo, nome, modelo, cor, peso, valor e Fabricante.

A11ex06.jar

- Copie o projeto anterior e ajuste-o para permitir CRUD de Venda, com os campos codigo, Produtos (List<Produto>), data, Vendedor, Cliente;
- Dica: crie a tabela ItensVenda com os campos venda, produto, valor (valor de venda) e quantidade.

O que aprendemos?

- Nesta aula aprendemos a acessar um servidor de banco de dados a partir de uma classe Java;
- Utilizamos o SGBD MySQL instalado na máquina atual, utilizando o WampServer;
- Implementamos o pacote DAO.

Na próxima aula...

- Framework Hibernate.