**ESCOLA SENAI “PROFESSOR VICENTE AMATO”**

Curso Técnico de Informática

JEAN CIGOLI DE ALMEIDA

**PROJETO ESTACIONAMENTO**

JANDIRA

2019

**PROJETO ESTACIONAMENTO**

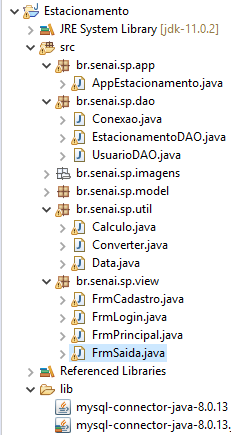
Para fazer login no aplicativo, terá que usar esse usuário:

Usuário: JeanCigoli

Senha: 12345

# 1 Estrutura de Código

Minha estrutura de pasta (onde ficaram todas as classes) que comporta o MVC, ficou assim:

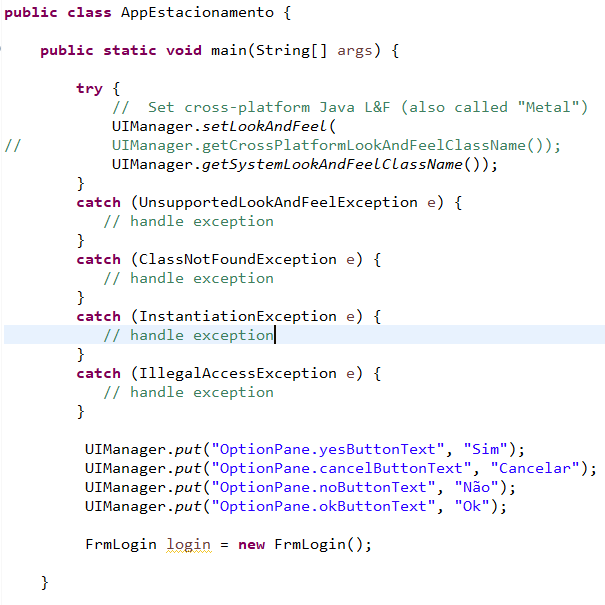


Agora será explica todas as classes com seus métodos e atributos;

## 1.1 Diretório App

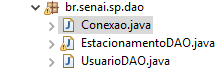
Esse diretório só é composto pela classe AppEstacionamento que nela contém o método *main*.



Dentro dessa classe só utilizei o *Look and Feel* para deixar a interface padrão do sistema operacional, e a criação do objeto FrmLogin.

## 1.2 Diretório dao

Dentro desse diretório contém três classes:



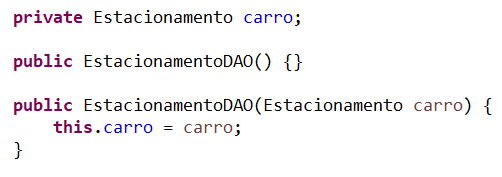
### 1.2.1 Classe Conexao

Esta foi utilizada para abertura do túnel entre o Banco de Dados e a aplicação, e foi composta por um atributo que seria a *Connection:*

E possui dois métodos um para abertura da conexão e uma para fechar:

### 1.2.2 Classe EstacionamentoDAO

Nesta classe foi utilizado para registrar e buscar os veículos no banco, ela possui um atributo e dois métodos construtores além dos que pertence ao CRUD (Create, *Read*, *Update* e *Delete*).



No *CREATE* existe ométodo gravar, onde pega os dados do carro e manda para o Banco:



No *READ* existe três métodos, um pegar os veículos estacionados, outro para pegar os que já saíram e o último para pegar a informação de um veículo.



Este que busca os veículos estacionados com acaba retornando mais de um, é necessário criar um *ArrayList* para poder trabalhar com os dados depois.

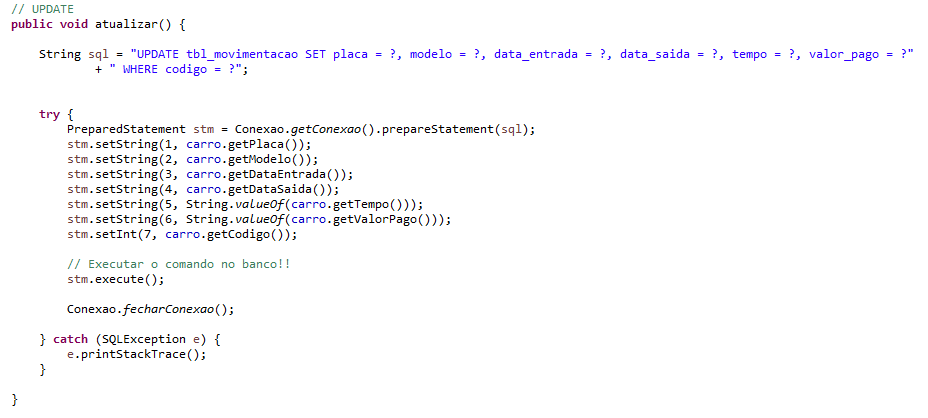
O próximo método que busca os que já saíram também será utilizado uma estrutura deste tipo.



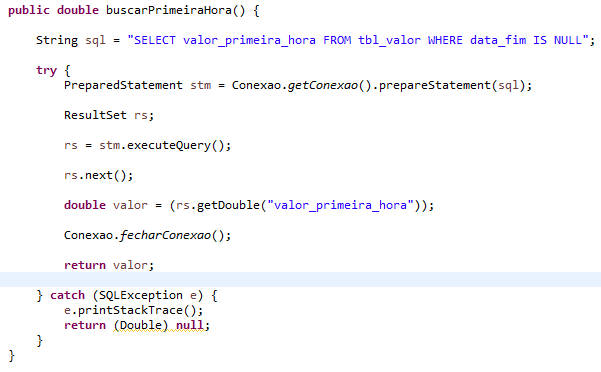
Já o ultimo é buscando só um e para isso acontecer ele precisa ter o número do código do veículo:



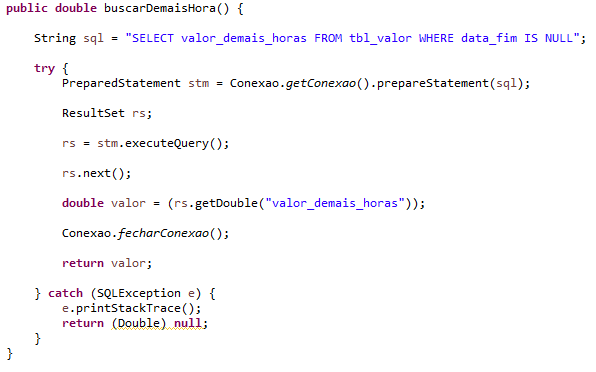
No *UPDATE* só possui um método, mas com duas funções. Como no nosso projeto não vai ter a exclusão de nenhum veículo porque para ele sair do estacionamento a única coisa que acontece é que ganha uma data de saída, nunca vai ser excluído seu registro do banco.



Como os valores da primeira hora e demais horas ficam em uma tabela no Banco, precisei crias dois métodos para pegar esses dois dados.

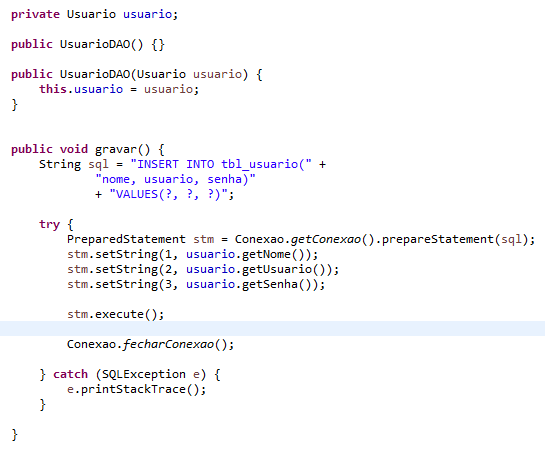
Um é o buscarPrimeiraHora:

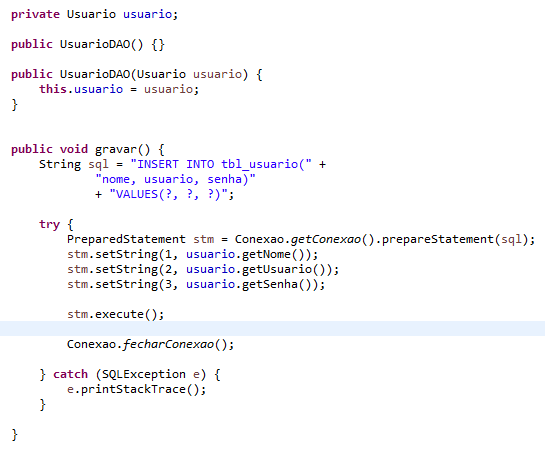
O outro é o buscarDemaisHoras:



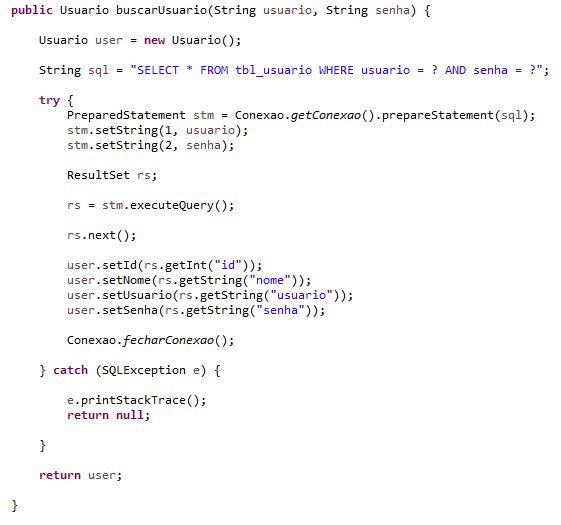
### 1.2.3 Classe UsuarioDAO

Esta classe foi utilizada para a verificação do usuário quando se logo no sistema do estacionamento, ela é composta por um atributo, dois métodos construtores e outros dois métodos: um para criar um usuário e outro para verificar se ele existe para poder logar.



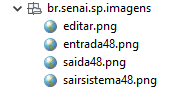


Neste próximo método onde verifica se o usuário existe, tem que passar como parâmetro duas Strings que é o usuário e a senha, assim o comando consegue retornar o usuário, ou se não existir retorna nulo.



## 1.3 Diretório imagens

Neste diretório é armazenado todos os ícones e imagens que serão utilizados no projeto, no projeto só utilizei quatro imagens:



Sendo essas imagens:



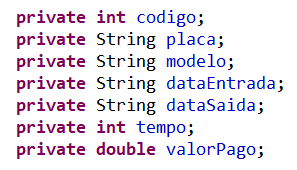
## 1.4 Diretório model

Neste diretório fica todos os modelos, todas as receitas dos objetos principais, que no caso desse projeto é o Estacionamento e o Usuário.



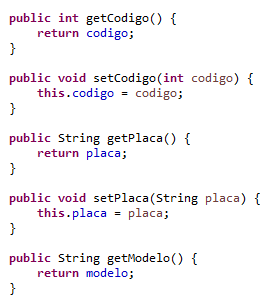
### 1.4.1 Classe Estacionamento

No estacionamento é composto pelos atributos:

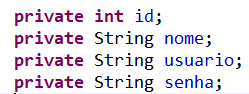


Que acaba sendo a descrição de tudo o que um estacionamento precisa para registrar um veículo.

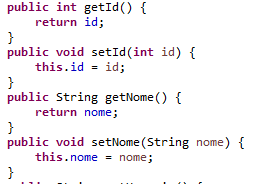
Os métodos desta classe acaba sendo os *Getters* e *Setters* de todos os atributos.



### 1.4.2 Classe Usuario

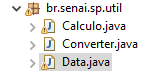
No usuário é composto pelos atributos:

Os métodos desta classe acaba sendo os *Getters* e *Setters* de todos os atributos.



## 1.5 Diretório utils

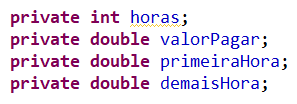
Neste diretório fica como diz o próprio nome as utilidades que nos ajudaram no projeto, ela é composta por três classes: Cálculo, Converter e Data.



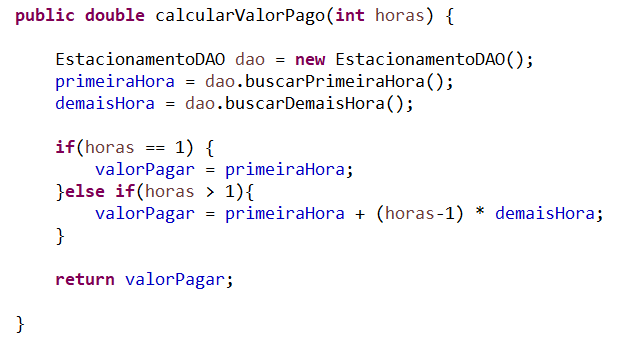
### 1.5.1 Classe Cálculo

Esta classe terá como função calcular o valor a pagar de cada veículo, pegando como referência as horas que cada veículo ficou, ela que utilizará os valores das horas direto do banco de dados.

Ela possui esses atributos:



E só um método que gera o valor para pagar:

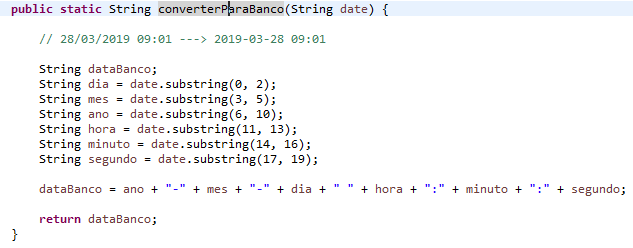


### 1.5.2 Classe Converter

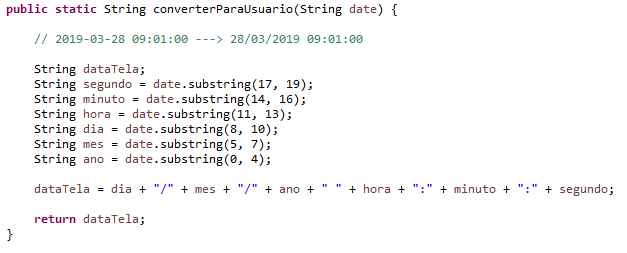
Essa classe é responsável para mudar os dados que vemos para o formato padrão que o banco de dados consegue armazenar.

Ela não possui nenhum atributo, só métodos que são todos estáticos, que são eles:

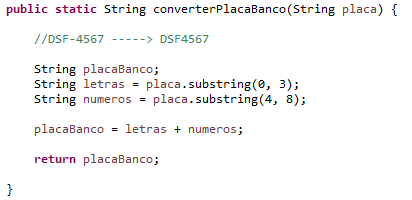
O converter data para banco:



O converter data para usuário:



E o último método, converter placa para banco:

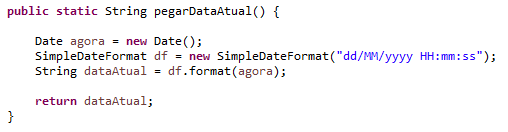


### 1.5.3 Classe Data

Esta classe é responsável para pegar a data atual, servindo para preencher tanto a hora que chegou e a hora que vai sair, e responsável em calcular o tempo que o veículo ficou estacionado pensando nos minutos de tolerância a cada uma hora.

Não possui nenhum atributo só dois métodos, sendo eles:

Pegar a data atual:

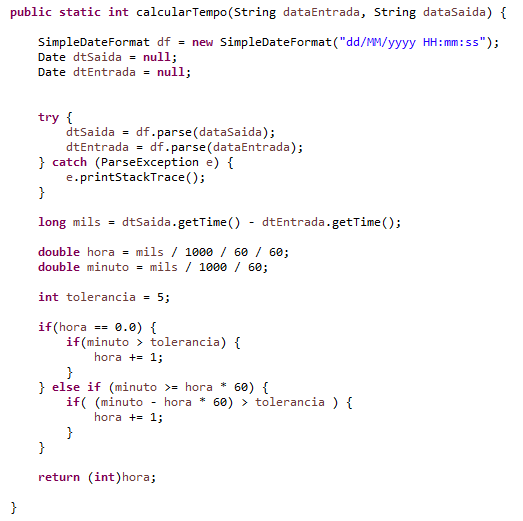


E o cálculo de tempo estacionado:

Neste método tem que mandar as datas de entrada e saída, assim ele transforma em milissegundos a subtração delas; depois transformando em horas e minutos, para facilitar a checagem da tolerância de 5 minutos.

Para verificar as horas utilizei: se a hora for igual a 0, ou seja, quando o veículo estiver a 39 minutos estacionados por exemplo, ele verifica se os minutos é maior que a tolerância, assim ele adiciona uma hora.

Quando estiver passado, se minutos for maior que 1 hora aí ele faz a subtração de minutos por essa hora vezes 60, e se esse resultado for maior que a tolerância então ele adiciona mais uma hora.



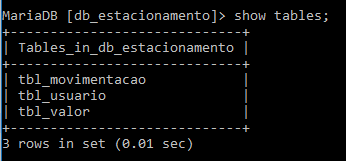
\*Obs.: Essa hora \* 60 irá trazer sempre redondo como 60 quando for igual a 1 hora, 120 = 2 horas, 180 = 3 horas, já os minutos serão quebrados.

## 1.6 Diretório view

# 2 Estrutura do banco de dados

Foi criado o banco db\_estacionamento.

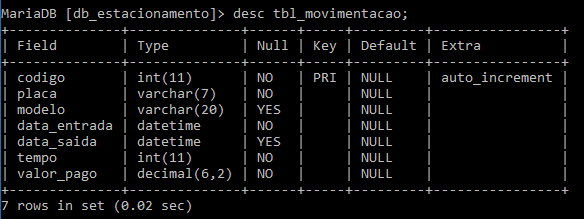
A nossa estrutura de banco de dados foi composta por três tabelas sendo elas: tbl\_movimentacao, tbl\_valor e a tbl\_usuario.



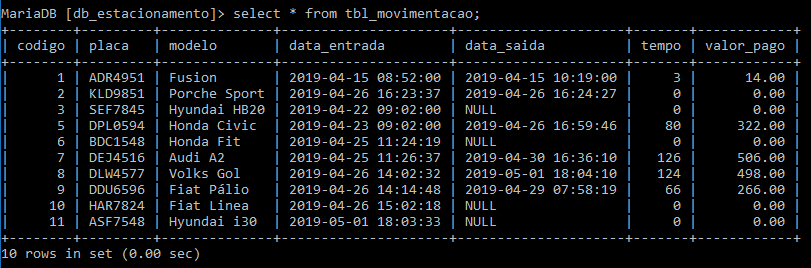
## 2.1 Tabela movimentação

Esta tabela ficou responsável por cuidar de todos os veículos que entraram no estacionamento.

Sua estrutura é:



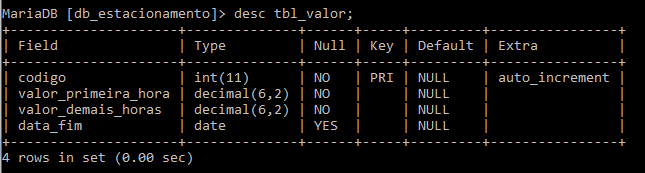
Já preenchida ficará assim:



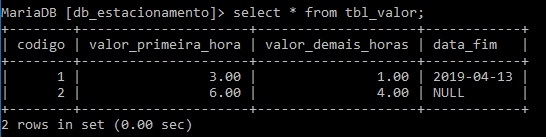
## 2.2 Tabela valor

Esta tabela é responsável por armazenar os valores que serão cobrados como o valor primeira hora e valor demais horas.

Sua estrutura é:



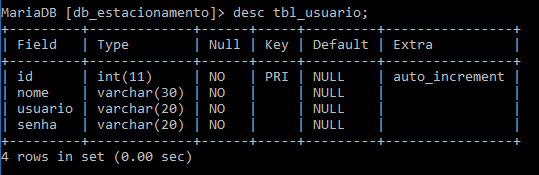
Preenchida fica:



## 2.3 Tabela usuário

Esta ficou responsável por armazenar os usuários que poderão usar o sistema, será utilizada na hora de logar.

Sua estrutura é:



Preenchida ficará:

