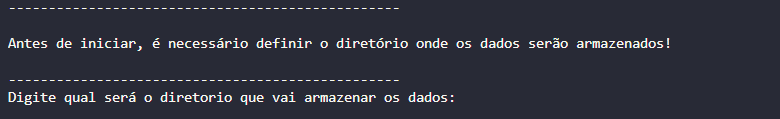
Documentação Trabalho de LPOO

**Objetivo:**

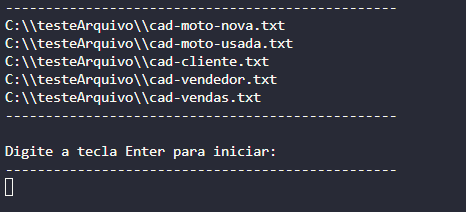
No Sistema de Concessionaria será possível fazer cadastro de motos tanto novas quanto de usadas, cadastro de cliente, de vendedor e também será possível realizar uma venda, relacionando o vendedor, o cliente e o produto que seria a moto escolhida.

Sistema simples para gestão de uma concessionaria, a proposta é facilitar o controle de vendas realizadas, e flexibilizar o dia a dia na empresa, nesse sistema também conseguirá puxar um relatório de todas a motos, clientes, vendedores e todas as vendas realizadas pela empresa.

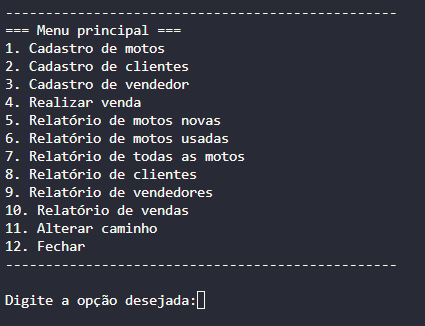
Ao rodar esse sistema no console, temos que primeiro definir qual será o diretório que armazenaram todos os arquivos. Cada um dos arquivos tem um nome próprio que serão concatenados a esse caminho e serão criados ao realizar o primeiro cadastro sozinhos.



Ao indicar qual será o caminho e clicar no (Enter), ele vai mostrar qual foi o caminho escolhido e o nome de cada um dos arquivos criados.



Digitando qualquer tecla ele vai para nova tela, onde teremos todas as opções possíveis.



Nessa tela conseguirá realizar todas estas opções indicadas, como o cadastro de motos ao escolher a opção 1, irá chamar um método dentro de um modulo chamado Cadastros, esse módulo possui toda a execução do programa, desde o 1 até o 11, temos também um outro modulo para as Consultas, esse módulo vai ler o arquivo e retornar todos os cadastros de acordo com o método escolhido, porém direcionam para o módulo Cadastros no lugar de chamar no programa principal.

**Parte do programa:**

Na parte do programa, foram necessários criar as seguintes classes:

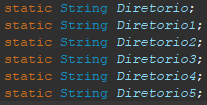
Super classes: Pessoa, Moto;

Sub classes: Vendedor, Cliente, MotoNova, MotoUsada, Diretorios, Vendas;

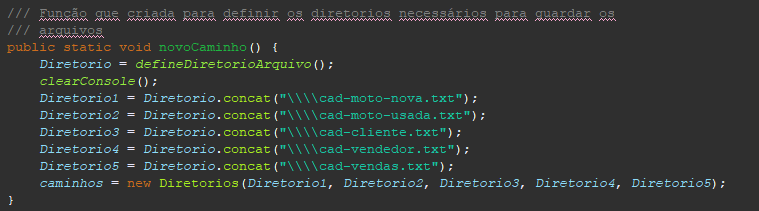
/// Diagramas e informações sobre as classes

No modulo Cadastro, teremos todas as execuções, como os cadastros, os relatórios e a definição dos arquivos.

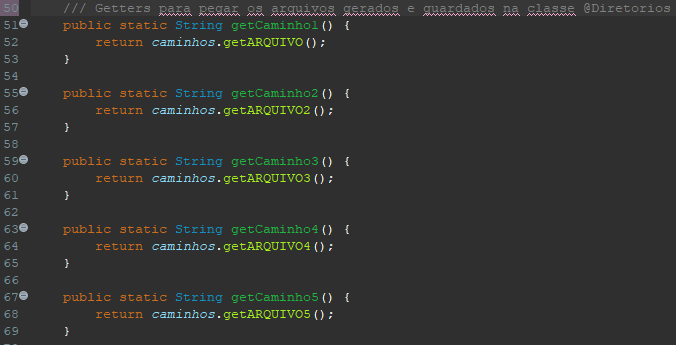
Para os arquivos, foram criadas as seguintes variáveis estáticas:



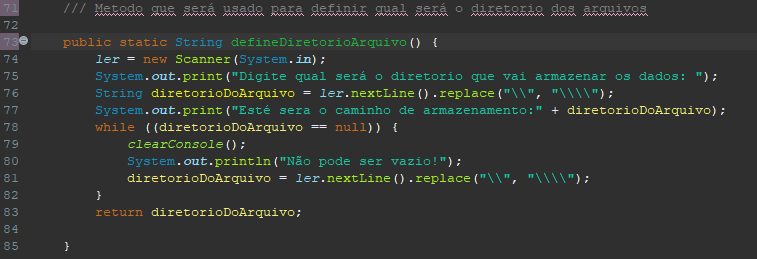
Essas variáveis vão receber cada um dos diretórios concatenando com o nome do arquivo. Após isso, será criado o nosso objeto caminhos do tipo Diretorios que receberá cada um desses caminhos e armazenará em constantes, dessa forma, não será apagada durante a execução do programa;



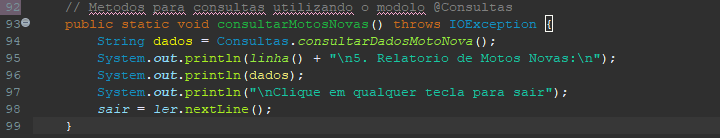
Também terão todos os Getters e Setters para esses valores serem usados futuramente nos cadastros e nos relatórios;



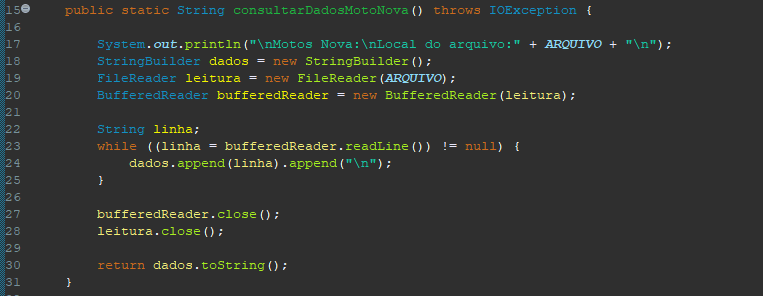
Método que define qual será o diretório:



Para as consultas foram criados métodos dentro do modulo Cadastros, a partir deles serão impressos os cadastros do método indicado, como o exemplo de consultarMotosNovas();

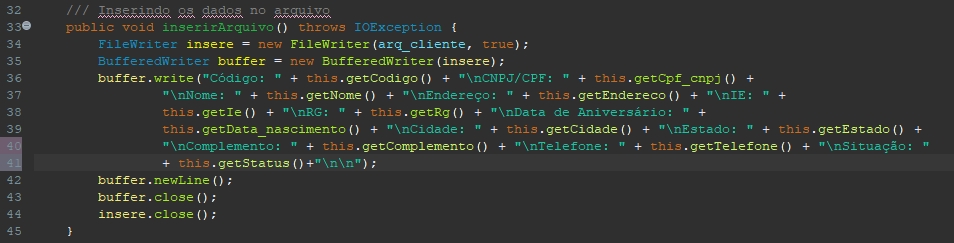


Nele também é chamado um outro método dentro do Modulo Consultas, que fará a leitura do arquivo para a exibição;



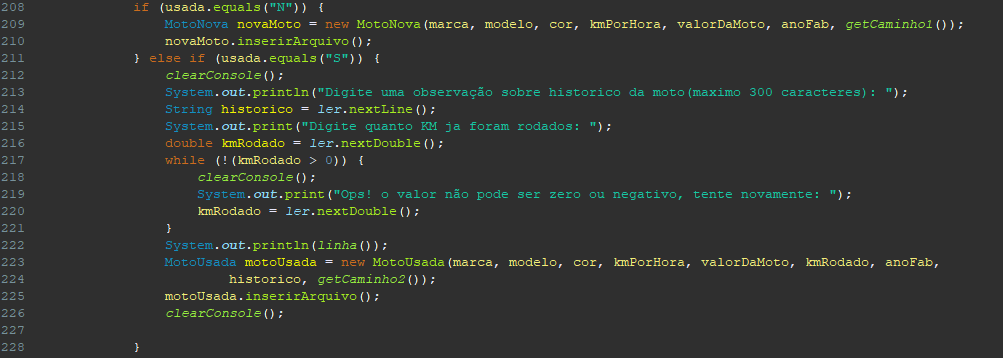
Todas as consultas tem a mesma lógica, o que irá mudar será o nome do arquivo, para que puxe cada um dos cadastros corretamente.

Para a inserção dos valores nos arquivo utilizei a seguinte logica:



Esse método está localizado na sua própria classe, dessa forma ao criar o objeto e inserir todos os dados necessários, que serão os nosso atributos e o nosso caminho com o nome do arquivo e que será armazenada em uma outra constante dentro da classe, utilizo o nomeObjeto.inserirArquivo(), ao fazer isso já teremos um novo registro dentro no nosso arquivo.

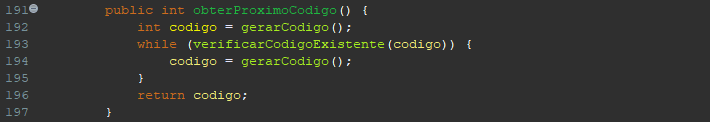
Temos por exemplo o Cadastro de motos, onde após inserir as informações necessárias vou incluir dentro do objeto.



Nessa parte do código ele verifica se é uma moto nova ou usada, caso for usada ele precisa inserir algumas informações a mais sobre o estado dessa moto, feito isso ele insere no objeto motoUsada e em seguida ele limpa o console.

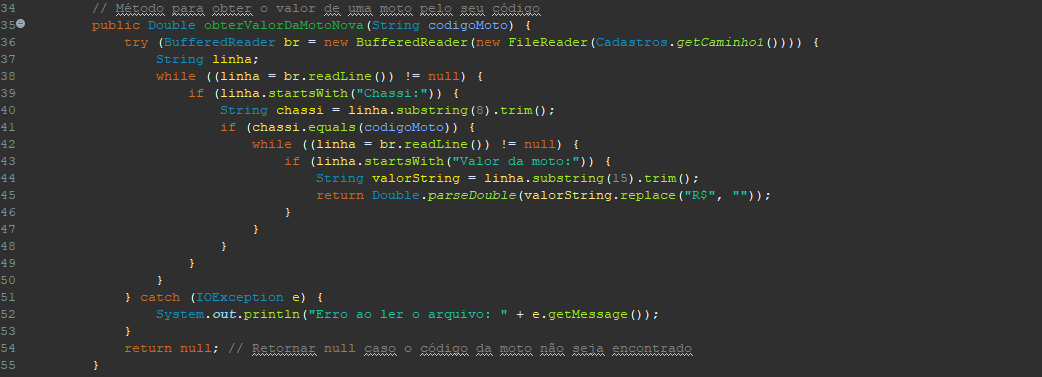
A classe Vendas representa a venda de motos em uma concessionária. Ela possui várias propriedades relacionadas à venda, como o código do vendedor, o nome do vendedor, o código do cliente, o nome do cliente, o CPF/CNPJ do cliente, o código da moto, o valor unitário da moto e o valor total da venda.

A classe possui dois construtores. O primeiro recebe o caminho do arquivo onde os dados das vendas serão armazenados. O segundo construtor também recebe os valores das propriedades da venda e o caminho do arquivo. Nesse segundo construtor, ele chama o método obterProximoCodigo() para gerar um novo código de venda e evitar duplicidade.

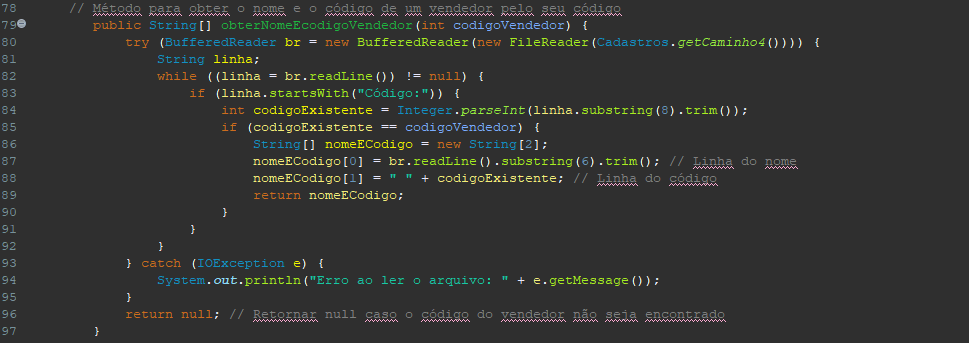


A classe possui vários métodos auxiliares. Alguns exemplos são:

obterValorDaMotoNova e obterValorDaMotoUsada: Esses métodos recebem o código da moto e retornam o valor correspondente da moto, lendo o arquivo de motos novas ou usadas, respectivamente.

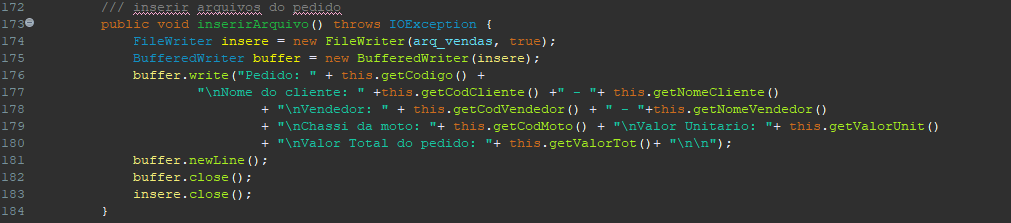


obterNomeEcodigoVendedor e obterCodigoENomeCliente: Esses métodos recebem o código do vendedor ou cliente e retornam o nome e código correspondentes, lendo o arquivo de vendedores ou clientes, respectivamente.



A classe também possui os métodos getter e setter para todas as propriedades da venda.

Além disso, a classe possui métodos para inserir a venda em um arquivo, gerar um código único para cada venda, verificar a existência de um código e obter o próximo código disponível.



No geral, a classe Vendas encapsula as informações e funcionalidades relacionadas às vendas de motos, permitindo o registro e manipulação desses dados.

No Menu ao seleciona a opção que realiza vendas, ele executara uma função RealizarNovaVenda() da classe Cadastros contém um loop while (true) que executa continuamente até que seja explicitamente interrompido pelo usuário. Isso permite que o usuário faça várias vendas consecutivas, se desejar. Dentro do loop, são exibidas várias mensagens de texto e solicitadas várias entradas ao usuário para coletar as informações necessárias para efetuar a venda.

Nessa Função são inseridos na nossa classe Vendas todas as informações fornecidas pelo cliente, ao realizar a venda ele insere todos os dados no nosso arquivo.

