```
class Rental(object):
#Classe pai que contém as listas que armazenam os valores dos obje

#Método que inicia primeiro na locadora, e que defineas listas
    def __init__(self):
        self.ClientList = []
        self.CarList = []
        self.RentList = []
        self.menu()

#Método que realiza a locação dos veículos apartir das informações
#Trás uma função que virifica se o cpf digitado já existe e permit
#Trás uma função que verifica se existe carro cadastrado, e ultili
#Trás uma função que retorna uma lista com os carros cadastrados
    def rent(self):

        print("="*50)
        self.cpf = str(input("Digite o seu CPF: "))
        print("="*50)
        print("\n\n")
        while True:
            if not Client.check_client(self):
                print("="*50)
                print("Novo Cadastro!")
                print("="*50)
                Client.new_client(self)
            else:
                break
        Car.car_list(self)
        Car.find_car_rent(self)
        Rental.new_rent_list(self)
        Rental.rent_list(self)


#Função que insere os carros alugados em uma lista
    def new_rent_list(self):
        alugado = {
            'nome': self.nome,
            'cpf': self.cpf,
```

```
            'rg': self.rg,
            'placa': self.placa,
            'ano': self.ano,
            'categoria': self.categoria,
            'transmissao': self.transmissao,
            'combustivel': self.combustivel,
            'marca': self.marca,
            'modelo': self.modelo,
            'aluguel': self.aluguel
        }
        self.RentList.append(alugado)
        print("="*50)
        print("Carro alugado com sucesso!")
        print("="*50)


    #Função que trás uma listagem de carros alugados
    def rent_list(self):
        if len(self.RentList) > 0:
            for i, loc in enumerate(self.RentList):
                print(f"Carro {i+1}:")
                print(f"Nome: {loc['nome']}")
                print(f"CPF: {loc['cpf']}")
                print(f"RG: {loc['rg']}\n")
                print(f"Placa: {loc['placa']}")
                print(f"Ano: {loc['ano']}")
                print(f"Categoria: {loc['categoria']}")
                print(f"Transmissão: {loc['transmissao']}")
                print(f"Combustível: {loc['combustivel']}")
                print(f"Marca: {loc['marca']}")
                print(f"Modelo: {loc['modelo']}")
                print(f"Aluguel:{loc['aluguel']}")
            print(f"\nTotal de veículos é: {len(self.RentList)}\n"
        else:
            print("\nNão há veículo alugado para listar!\n")

    #Função que chama o menu do sistema, e pela função digitada faz a
    def menu(self):

        while True:
```

```
print("================================================
print("####         LOCADORA THALLES & INGRID
print("================================================
print("####   1 - Cadastrar novo veículo
print("####   2 - Cadastrar novo cliente
print("####   3 - Locação de veículo
print("####   4 - Relatório de locação
print("####   5 - Busca de veículos cadastrados
print("####   6 - Busca de clientes cadastrados
print("####   7 - Relatório de veículos cadastrados
print("####   8 - Relatório de clientes cadastrados
print("####   9 - Alterar dados veículos cadastrados
print("####   10 - Alterar dados clientes cadastrados
print("####   11 - Finalizar o programa !!!
print("================================================

while True:
    try:
        opc = int(input("\nDigite a opção desejada: ")
        break
    except ValueError:
        print("\nOpss, não aceita letras!\n")

if opc == 1:
    Car.new_car(self)

elif opc == 2:
    Client.new_client(self)

elif opc == 3:
    Rental.rent(self)

elif opc == 4:
    Rental.rent_list(self)

elif opc == 5:
    Car.find_car(self)

elif opc == 6:
    Client.find_client(self)
```

```
            elif opc == 7:
                Car.car_list(self)

            elif opc == 8:
                Client.client_list(self)

            elif opc == 9:
                Car.change_car(self)

            elif opc == 10:
                Client.change_client(self)

            elif opc == 11:
                print("\nSistema Finalizado!!!\nObrigado pelo uso"
                break

            else:
                print("\nOpção inválido!\n")


class Car(Rental):
#Classe filho Car da classe pai Locadora

#Método que inicia a classe, trazendo junto os dados de cada veícu
    def __init__(self, placa, ano, categoria, transmissao, combust
        self.placa = placa
        self.ano = ano
        self.categoria = categoria
        self.transmissao = transmissao
        self.combustivel = combustivel
        self.marca = marca
        self.modelo = modelo
        self.aluguel = aluguel

#Função que altera os dados de um veículo já cadastrado
    def change_car(self):
        if len(self.CarList) > 0:
            self.placa = str(input("Digite a placa : ")).upper()
            if Car.check_car(self):
```

```
                for car in self.CarList:
                    if car['placa'] == self.placa:
                        print(f"\nPlaca: {car['placa']}")
                        print(f"Ano: {car['ano']}")
                        print(f"Categoria: {car['categoria']}")
                        print(f"Transmissão: {car['transmissao']}"
                        print(f"Combustível: {car['combustivel']}"
                        print(f"Marca: {car['marca']}")
                        print(f"Modelo: {car['modelo']}")
                        print(f"Aluguel: {car['aluguel']}\n")

                        car['placa'] = str(input("Placa:")).upper(
                        car['ano'] = str(input("Ano:"))
                        Car.category(self)
                        car['categoria'] = self.categoria
                        Car.transmission(self)
                        car['transmissao'] = self.transmissao
                        Car.fuel(self)
                        car['combustivel'] = self.combustivel
                        car['marca'] = str(input("Marca:")).upper(
                        car['modelo'] = str(input("modelo:")).uppe
                        car['aluguel'] = print(
                            f"\nAluguel: {car['aluguel']}\n")

                        print(f"\nOs dados da {self.placa} foram a
                        break

            else:
                print(
                    f"\nNão existe veiculo cadastrado com a placa
        else:
            print("\nNão existe veículo a ser alterado!\n")

    #Função que faz um relatório dos carros já cadastrados
        def car_list(self):
            if len(self.CarList) > 0:
                for i, car in enumerate(self.CarList):
                    print("\n")
                    print("="*50)
                    print(f"\nCarro {i+1}:")
```

```
                print(f"Placa: {car['placa']}")
                print(f"Ano: {car['ano']}")
                print(f"Categoria: {car['categoria']}")
                print(f"Transmissão: {car['transmissao']}")
                print(f"Combustível: {car['combustivel']}")
                print(f"Marca: {car['marca']}")
                print(f"Modelo: {car['modelo']}")
                print(f"Aluguel:{car['aluguel']}")
                print("="*50)
                print("\n")

            print(
                f"\n\nTotal de veículos cadastrados é: {len(self.C
        else:
            print("\nNenhum veículo cadastrado para listar!\n")

    #Função que busca um carro
    def find_car(self):
        if len(self.CarList) > 0:
            self.placa = str(input("Digite a placa: ")).upper()
            for car in self.CarList:
                if car['placa'] == self.placa:
                    print(f"Placa: {car['placa']}")
                    print(f"Ano: {car['ano']}")
                    print(f"Categoria: {car['categoria']}")
                    print(f"Transmissão: {car['transmissao']}")
                    print(f"Combustível: {car['combustivel']}")
                    print(f"Marca: {car['marca']}")
                    print(f"Modelo: {car['modelo']}")
                    print(f"Aluguel:{car['aluguel']}")
                    break
                else:
                    print("\nPlaca não cadastrada\n")
        else:
            print("\nNenhum veiculo cadastrado\n")



    #Dunção que busca o carro que se pretende alugar
    def find_car_rent(self):
```

```
        if len(self.CarList) > 0:
            self.placa = str(input("Digite a placa: ")).upper()
            for car in self.CarList:
                if car['placa'] == self.placa:
                    print("O carro selecionado é:")
                    print(f"Placa: {car['placa']}")
                    print(f"Ano: {car['ano']}")
                    print(f"Categoria: {car['categoria']}")
                    print(f"Transmissão: {car['transmissao']}")
                    print(f"Combustível: {car['combustivel']}")
                    print(f"Marca: {car['marca']}")
                    print(f"Modelo: {car['modelo']}")
                    print(f"Aluguel:{car['aluguel']}")
                    self.data_ininio = str(input("Qual a data de i
                    self.data_fim = str(input("Qual a data final d
                    if car['aluguel'] == "ALUGADO":
                        print("\nO veículo já está locado!")
                    else:
                        car['aluguel'] = "ALUGADO"
        else:
            print("\nNenhum veiculo disponível para alugar!!!")

    #Função que verifica se a placa cadastrada já existe no sistema
    def check_car(self):
        if len(self.CarList) > 0:
            for car in self.CarList:
                if car['placa'] == self.placa:
                    return True
        return False

    #Função para cadastrar os valores possíveis de categorias pré cada
    def category(self):
        while True:
            print("Escolha uma opção\n1 - Sedan\n2 - Picape\n3 - S
            while True:
                try:
                    ctg = int(input("\nDigite uma opção: "))
                    break
                except ValueError:
                    print("\nEsse sistema não aceita letras!\n")
```

```
            if ctg == 1:
                self.categoria = "SEDAN"
                break
            elif ctg == 2:
                self.categoria = "PICAPE"
                break
            elif ctg == 3:
                self.categoria = "SUV"
                break
            else:
                print("\nOpção inválida!")

    #Função para cadastrar os valores possíveis de transmissão pré cad
        def transmission(self):
            while True:
                print("Escolha uma opção\n1 - Automático\n2 - Manual")
                while True:
                    try:
                        trns = int(input("\nDigite uma opção: "))
                        break
                    except ValueError:
                        print("\nNão aceita letras!\n")

                if trns == 1:
                    self.transmissao = "AUTOMATICO"
                    break
                elif trns == 2:
                    self.transmissao = "MANUAL"
                    break
                else:
                    print("\nOpção inválida!")

    #Função para cadastrar os valores possíveis de combustível pré cad
        def fuel(self):
            while True:
                print(
                    "Escolha uma opção\n1 - Gasolina\n2 - Álcool\n3 -
                while True:
                    try:
```

```
                    cmbst = int(input("\nDigite uma opção: "))
                    break
                except ValueError:
                    print("Esse sistema não aceita letras!\n")

            if cmbst == 1:
                self.combustivel = "GASOLINA"
                break
            elif cmbst == 2:
                self.combustivel = "ALCOOL"
                break
            elif cmbst == 3:
                self.combustivel = "FLEX"
                break
            elif cmbst == 4:
                self.combustivel = "DIESEL"
                break
            elif cmbst == 5:
                self.combustivel = "GNV"
                break
            else:
                print("\nOpção inválida!")

    #Função que recebe os atributos dos veiculos e insere dentro de um
        def new_car(self):

            while True:
                self.placa = str(input("\n\nDigite a placa: ")).upper(
                if not Car.check_car(self):
                    break
                else:
                    print("\nCarro já cadastrado no sistema")
            self.ano = int(input("Digite o ano: "))

            Car.category(self)

            Car.transmission(self)

            Car.fuel(self)
```

```
            self.marca = str(input("Digite a marca: ")).upper()
            self.modelo = str(input("Digite o modelo: ")).upper()
            self.aluguel = "DISPONIVEL"
            dict_car = {
                'placa': self.placa,
                'ano': self.ano,
                'categoria': self.categoria,
                'transmissao': self.transmissao,
                'combustivel': self.combustivel,
                'marca': self.marca,
                'modelo': self.modelo,
                'aluguel': self.aluguel
            }
            self.CarList.append(dict_car)

    class Client(Rental):
    #Classe filha Client da classe pai Locadora, com os métodos dos cl

    #Método com os atributos dos clientes
        def __init__(self, nome, cpf, rg):
            self.nome = nome
            self.cpf = cpf
            self.rg = rg

    #Função que altera os dados dos clientes pré-cadastrados no sistem
        def change_client(self):
            if len(self.ClientList) > 0:
                self.cpf = str(input("Digite o CPF do cliente: ")).upp
                if Client.check_client(self):
                    for clt in self.ClientList:
                        if clt['cpf'] == self.cpf:
                            print(f"\nNome: {clt['nome']}")
                            print(f"CPF: {clt['cpf']}")
                            print(f"RG: {clt['rg']}")

                            clt['nome'] = str(input("Nome: ")).upper()
                            clt['cpf'] = str(input("CPF: "))
                            clt['rg'] = str(input("RG: "))

                            print(
                                f"Os dados de {self.nome} foi alterado
```

```
                            print(f"Os dados de {self.nome} foi alterado
                        break

            else:
                print("\nEsse CPF não está cadastrado no sistema\n
        else:
            print("\nNão existem clientes a serem alterados no sis

    #Função que trás um relatório de cleintes já cadastrados
    def client_list(self):
        if len(self.ClientList) > 0:
            for i, clt in enumerate(self.ClientList):
                print(f"\nCliente {i+1}:")
                print(f"Nome: {clt['nome']}")
                print(f"CPF: {clt['cpf']}")
                print(f"RG: {clt['rg']}\n")
            print(
                f"\nTotal de clientes cadastrados é: {len(self.Cli
        else:
            print("\nNenhum cliente para listar!\n")

    #Função que busca clientes cadastrados
    def find_client(self):
        if len(self.ClientList) > 0:
            self.cpf = str(input("Digite o cpf: ")).upper()
            for car in self.ClientList:
                if car['cpf'] == self.cpf:
                    print(f"\nNome: {car['nome']}")
                    print(f"CPF: {car['cpf']}")
                    print(f"RG: {car['rg']}\n")
                    break
        else:
            print("\nNenhum cliente cadastrado no sistema!\n")

    #Função que verifica se o cpf já foi ultiliado anteriormente
    def check_client(self):
        if len(self.ClientList) > 0:
            for clt in self.ClientList:
                if clt['cpf'] == self.cpf:
                    return True
        return False
```

```
    #Função que faz o cadastro de um novo cliente, e insere os dados e
        def new_client(self):
            while True:
                self.cpf = str(input("Digite o CPF: "))
                if not Client.check_client(self):
                    break
                else:
                    print("Cliente já cadastrado no sistema")
            self.nome = str(input("Nome do cliente: ")).upper()
            self.rg = str(input("Difite o RG: "))
            user = {
                'nome': self.nome,
                'cpf': self.cpf,
                'rg': self.rg
            }

            self.ClientList.append(user)

    Rental()
```

```
Ano: 2022
Categoria: SUV
Transmissão: AUTOMATICO
Combustível: ALCOOL
Marca: LAMBORGINI
Modelo: URUS
Aluguel:DISPONIVEL
Carro 3:

Nome: KAUAN
CPF: 123456789
RG: 123456789

Placa: JIU2366
Ano: 2022
Categoria: SUV
Transmissão: AUTOMATICO
Combustível: ALCOOL
Marca: LAMBORGINI
Modelo: URUS
Aluguel:DISPONIVEL
Carro 4:
Nome: KAUAN
```

```
CPF: 123456789
RG: 123456789

Placa: URU1234
Ano: 2022
Categoria: SUV
Transmissão: AUTOMATICO
Combustível: ALCOOL
Marca: LAMBORGINI
Modelo: URUS
Aluguel:DISPONIVEL

Total de veículos é: 4

====================================================
####         LOCADORA THALLES & INGRID           ####
====================================================
####   1 - Cadastrar novo veículo                ####
####   2 - Cadastrar novo cliente                ####
####   3 - Locação de veículo                    ####
####   4 - Relatório de locação                  ####
####   5 - Busca de veículos cadastrados         ####
####   6 - Busca de clientes cadastrados         ####
####   7 - Relatório de veículos cadastrados     ####
####   8 - Relatório de clientes cadastrados     ####
####   9 - Alterar dados veículos cadastrados    ####
####   10 - Alterar dados clientes cadastrados   ####
####   11 - Finalizar o programa !!!             ####
====================================================


Digite a opção desejada: 11

Sistema Finalizado!!!
Obrigado pelo uso
<__main__.Rental at 0x7f67133e3410>
```

Tive muita dificuldade em entender o paradigma orientado a objeto, devido a sua grande diferença com o ultimo paradigma apresentado a nós, que foram os procedurais e também o funcional, tive que fazer grande uso do "Stack Over Flow" para tentar sanar minhas milhares

de dúvidas, e buscar em vídeos na internet para encontrar algumas soluções!

✓ 6m 34s completed at 3:29 AM