UNIVERSIDADE
FEDERAL DO CEARÁ

UNIVERSIDADE FEDERAL DO CEARÁ
CURSO DE TECNÓLOGO EM SEGURANÇA DA INFORMAÇÃO

JOSÉ BASÍLIO BRANDÃO COELHO
RYAN GUILHERME

CRIAÇÃO DE UM CARRO ECOLÓGICO EM PROGRAMAÇÃO ORIENTADA A OBJETO NA LINGUAGEM PYTHON

ITAPAJÉ
2023

Matéria: Desenvolvimento de Sistemas
Curso de Segurança da Informação
Semestre 2023.2

Relatório de Trabalho - Carro Ecológico

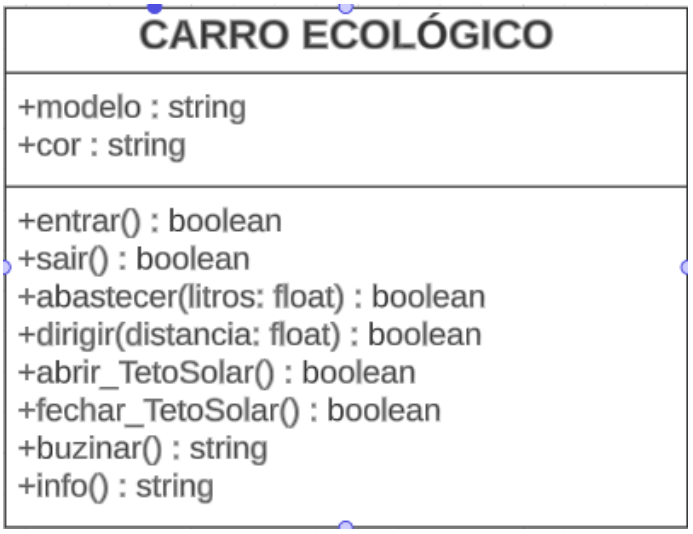
Autores:
Ryan Guilherme
Basílio Brandão

Resumo:
Este relatório descreve o desenvolvimento de uma classe em Python para um carro ecológico. O projeto envolve a implementação de métodos e atributos que permitem operações como entrar, sair, abastecer, dirigir, abrir e fechar o teto solar, buzinar e fornecer informações sobre o carro.

Introdução
Neste projeto, criamos uma classe denominada "Carro_Ecologico" que representa um veículo ecológico. A classe permite simular várias operações que um carro real pode realizar.

Desenvolvimento

Diagrama de Classe:



A classe Carro representa o veículo do sistema. Ela possui os atributos: modelo e cor. Os métodos são : entrar, sair, abastecer, dirigir, abrir teto solar, fechar teto solar, buzinar e informações. Esses métodos permitem interagir com o objeto carro, podendo pegar passageiros , abastecer , ver quilometragem percorrida, dirigir ,buzinar, abrir e fechar teto solar e ver suas informações.

Código

```
class Carro_Ecologico:
    def __init__(self,modelo,cor):
        self.modelo = modelo
        self.cor = cor
        self.tanque = 0
        self.quilometragem = 0
        self.pessoas = 0
        self.max_pessoas = 2
        self.teto_solar_aberto = False

    def entrar(self):

        if self.pessoas < self.max_pessoas:
            self.pessoas += 1
            return "Uma pessoa entrou no carro."

        else:
            return "O carro está lotado."

    def sair(self):

        if self.pessoas > 0:
            self.pessoas -= 1
            return  "Uma pessoa saiu no carro."

        else:
```

```
            return "Não há pessoas dentro do carro."

    def abastecer(self, litros):

        if self.tanque + litros <= 100:
            self.tanque += litros
            return f"O carro foi abastecido com {litros} litros de água."

        else:
                return "O tanque atingiu seu máximo de combustível. O excesso foi descartado."

    def dirigir(self,distancia):

        if self.pessoas > 0:
            kilometros_maximos = min(self.tanque, distancia)
            self.tanque -= kilometros_maximos
            self.quilometragem += kilometros_maximos
            if kilometros_maximos < distancia:
                    return f"A viagem não concluída totalmente. Só foi possível andar {kilometros_maximos} Km."

            else:
                return f"O carro andou {kilometros_maximos} Km."

        else:
             print("Não é possível andar com o carro. Certifique-se se existem pessoas e combustível suficiente no carro.")
            kilometros_maximos = 0
        return kilometros_maximos

    def abrir_TetoSolar(self):
        if not self.teto_solar_aberto:
            self.teto_solar_aberto = True
```

```
            return "Teto solar abriu."

        else:
            return "O teto solar já está aberto."

    def fechar_TetoSolar(self):

        if self.teto_solar_aberto:
            self.teto_solar_aberto = False
            return "Teto solar fechou."
        else:
            return "O teto solar já fechou."

    def buzinar(self):
        return "Bip Bip, sai da frente!!!"

    def info(self):
        teto_solar = "Aberto" if self.teto_solar_aberto else "Fechado."
            return f"Carro Ecológico: \n Tanque: {self.tanque} litros.\n Quilometragem: {self.quilometragem} km.\n Pessoas: {self.pessoas}.\n Teto Solar: {teto_solar}. \n Modelo: {self.modelo}. \n Cor: {self.cor}. "

carro = Carro_Ecologico('Fiat', 'Preto')
```

Classe Carro_Ecologico

A classe "Carro_Ecologico" foi implementada com os seguintes atributos:

- tanque: Representa o nível do tanque de combustível.
- quilometragem: Registra a quilometragem percorrida.
- pessoas: Contabiliza o número de pessoas a bordo.
- max_pessoas: Define o número máximo de pessoas permitidas no carro.
- teto_solar_aberto: Indica se o teto solar está aberto.
- modelo: Representa o modelo do carro.
- cor: Representa a cor do carro.

O método construtor da classe inicia os valores iniciais dos atributos, com o tanque em 0, a quilometragem em 0, o número de pessoas em 0 e o valor máximo de pessoas definido como 2.

Métodos da Classe

- Método "entrar": Verifica se a quantidade de pessoas para entrar é menor que a quantidade limite de pessoas permitidas, se for menor, ele incrementa esse valor das pessoas.

- Método "sair": Se a quantidade de pessoas no carro for maior que 0, o valor de pessoas vai receber -1, se não, não existe ninguém dentro do carro.

- Método "abastecer(litros)": Ele pega o valor do atributo tanque e soma com a quantidade de litros que a pessoa quer abastecer, tendo o limite de até cem litros, caso ultrapasse o limite definido, a quantidade que passou vai ser descartada.

- Método "dirigir(distancia)": Inicia verificando a quantidade de pessoas no carro, se a quantidade de pessoas no carro for maior do que 0, ele vai começar a andar, caso a quantidade de pessoas seja 0, ele não vai andar, a contagem dos quilômetros ocorre se houver pelo menos 1 pessoa no carro, o método min é usado para determinar o valor mínimo entre o nível de água atual no tanque (self.tanque) e a distância desejada (distancia).

Isso garante que o carro não tente percorrer mais distância do que a quantidade de água disponível no tanque, a atualização do estado do carro com base na distância máxima calculada, primeiro subtrai-se o valor de kilometros_maximos com o nível do tanque (self.tanque -= kilometros_maximos) e depois a mesma quantidade de quilômetros é adicionada a quilometragem total percorrida (self.quilometragem += kilometros_maximos).

Após atualizar o tanque e a quilometragem, o código verifica se a distância máxima percorrida (kilometros_maximos) é menor do que a distância total desejada (distancia), if kilometros_maximos < distancia. Essa verificação indica que o carro não conseguiu percorrer toda a distância desejada com a água disponível no

tanque. Nesse caso, o método retorna uma mensagem informando que a viagem não foi concluída totalmente e especifica quantos quilômetros foram possíveis percorrer. Se o carro conseguir percorrer a distância total desejada com a água disponível no tanque, o método retornará uma mensagem indicando quantos quilômetros foram percorridos com sucesso.

- Método "abrir_teto_solar": É usada a condição if junto com o not para dizer que se o teto solar está fechado, ele vai abrir atribuindo o True para teto_solar_aberto, e vai receber a mensagem “Teto solar abriu”, se não, a mensagem “Teto solar já está aberto”.

- Método "fechar_teto_solar": Verificamos com a condição if se o teto_solar_aberto é igual a False, se sim recebe a mensagem “Teto solar fechou”, se não, recebe a mensagem “Teto solar já está fechado”.

- Método "buzinar": Caso alguém utilize a buzina, recebe a mensagem “Bip bip, sai da frente!!”.

- Método "info": Fornecer informações sobre o carro, incluindo o status do teto solar, o nível do tanque, a quilometragem,o número de pessoas a bordo, modelo e cor.

Conclusão

O projeto de desenvolvimento de uma classe de carro ecológico foi concluído com êxito. Todos os métodos foram implementados e testados. O código demonstra o funcionamento adequado das operações relacionadas a um carro ecológico.

Referências

pythonando. O guia completo de POO com python. 2022.
YouTube. https://youtu.be/jeLeW6q9Mo4?si=2vVFFcSyWt35I2_m
Hashtag Programação. Como sair do zero em Classes no python - self e init explicados.2022.Youtube. https://youtu.be/gomDSZaay3E?si=G9tDdPVsW-XtxXYn
pythonando. O guia completo de POO com python - staticmethods vs class methods. 2022. YouTube.
https://youtu.be/gLaZQYZjcOE?si=8ZZwzscbY52yhYyp