**PROJETO (ESTRUTURA) DO PROGRAMA**

Para construir o programa em Java capaz de sanar as necessidades propostas, com boas práticas de programação e utilizando alguns conceitos importantes apresentados para garantir que as funcionalidades sejam implementadas e demonstrar os conhecimentos vistos em sala de aula, será necessário criar 4 classes e uma interface. Essa interface terá apenas um método que mostrará os resultados para o usuário no terminal, que será implementado na classe “UsaClasses”, responsável por executar todos os métodos de todas as classes, assim como tratar os dados do usuário e manipular atributos de forma devida.

As outras três classes, serão a base de toda a proposta do trabalho. Nelas terão os atributos, os fatores de multiplicação para conversão da queima de combustível em CO2, os métodos que serão invocados na “UsaClasses”, e a grande maioria dos conceitos de LPOO descritos no tópico “Elementos de programação orientada a objetos” deste trabalho.

**Figura x –** Diagrama de classes do sistema a ser criadoDiagrama

Descrição gerada automaticamente

**Fonte:** Autoral

Este diagrama acima, demonstra as classes, relação entre elas, seus métodos, atributos e modificadores de acesso e comportamento. Inicialmente é possível verificar que todas as classes têm seus atributos como private (-), sendo necessário gerar os métodos get e set de todos os atributos do programa para utilização deles através da UsaClasses.

Logo em seguida, evidenciasse o uso de herança nesse programa, dado que a classe veículo é uma generalização comum entre a classe Carro e a classe TransportePúblico, já que carro e ônibus são ambos veículos. Dessa forma, utilizando o método construtor com parâmetros para definir os atributos da classe veículos, é possível usar o método construtor das suas classes filhas para setar além do atributo delas próprias, os atributos da classe mãe, através do comando super, que indicia superclasse. Com isso, ao instanciar os objetos Carro e ônibus, todos os atributos serão definidos dado os parâmetros definidos pelo usuário como entrada de dados.

Quanto aos modificadores de comportamento dos elementos das classes, a classe veículo possui o modificador abstract, tornando-a abstrata. Esse recurso serve para deixar a codificação padronizada e organizada, pois, já que a classe Veículo é uma superclasse, então, seus elementos apenas podem ser acessados através de suas classes filhas.

Como a classe é abstrata, o intuitivo é que se utilize também métodos abstratos dentro dela, para que seja acessado através apenas de suas classes filhas. No entanto, como forma de aplicação dos conhecimentos adquiridos em sala de aula, nesse trabalho será criado métodos estáticos na classe veículos, como forma de aplicação dos importantes conceitos de LPOO e para tornar mais fácil a utilização desses métodos.

Para utilizar os modificadores de comportamento de forma assertiva, também foi declarado as classes filhas como finais, com o comando final. Pois elas não terão outras classes que herdarão seus elementos gerando uma herança encadeada, com isso, não serão instanciadas e conterão o comando final.

Por final, a interface MostrarDados, serve para organizar as saídas de dados para o usuário no terminal, após o processamento da entrada de dados dada pelo usuário. Dessa forma, a UsaClasses que a implementará, será obrigada a construir o método mostrarDados():void, que será invocado no final do programa.

A tela do programa está funcionando da seguinte forma, primeiramente para se o usuário escolhe utilizar gasolina como parâmetro:

**Figura x** – Tela de resultado Gasolina

![Texto

Descrição gerada automaticamente]()

**Fonte**: Autoral

Dessa vez, para se o usuário deseja utilizar etanol como parâmetro para cálculo:

**Figura x –** Tela resultado Etanol

![Texto

Descrição gerada automaticamente]()

**Fonte:** Autoral