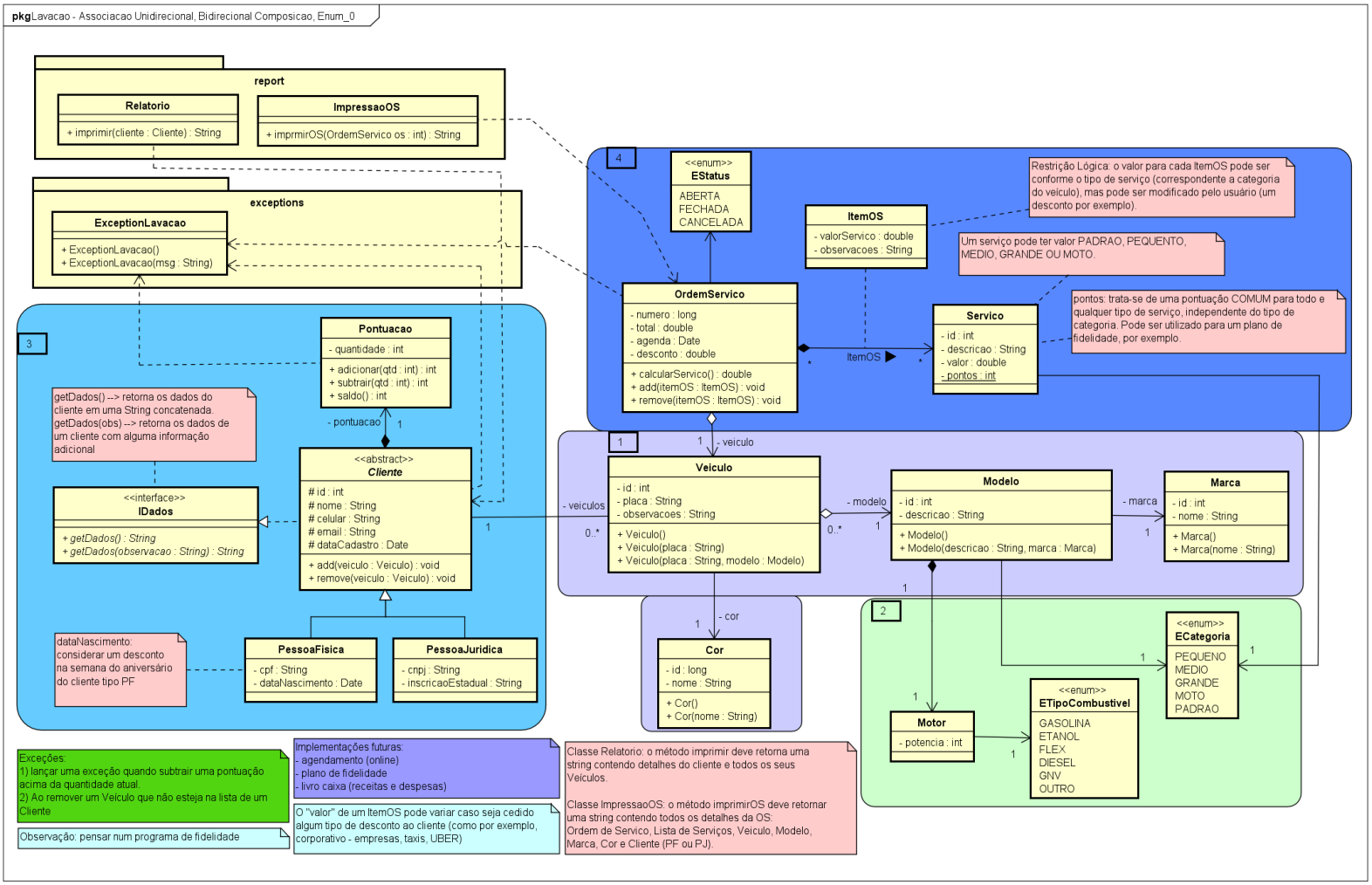
Atividade mão na massa 3

Observe o diagrama de classe apresentado na figura a seguir (a imagem do diagrama completo, considerando uma continuidade das atividades Mão na Massa 1 e 2, pode ser visualizada melhor na Figura 3, no final do documento).



Implemente os seguintes requisitos dos projetos:

1. Implemente a classe Servico (os atributos de instância e o membro de classe estático **pontos**). Observe que essa classe deve ter um atributo ECategoria também. O atributo ECategoria é útil para determinar o **valor** do serviço de acordo com o tipo de categoria do serviço, por exemplo:
    - Uma lavação completa de categoria PEQUENO terá valor de R$ 40,00
    - Uma lavação completa de categoria MEDIO terá valor de R$ 50,00
2. Implemente o enum EStatus
3. Implemente a classe OrdemServido
4. Implemente a classe ItemOS (o atributo valorServico desta classe poderá ser o mesmo do valor do serviço, mas pode ser diferente se o usuário desejar definir outro valor para a execução do serviço, por exemplo, um desconto no ato da execução do serviço).

5. Declarações apropriadas para a implementação de classe Associativa entre OrdemServico, ItemOS e Servico.
    - A classe ItemOS deverá ter uma relação com uma OrdemServico e um Servico
    - A classe OrdemServico deverá ter uma lista de ItemOS
6. Os métodos da classe OrdemServico:
    - add(itemOS: ItemOS) - método para adicionar um novo item de ordem de serviço na lista de ItensOS da OrdemServico
    - remove(itemOS: ItemOS) - método para remover um novo item de ordem de serviço na lista de ItensOS da OrdemServico
    - calcularServico() - método para calcular o valor total da OrdemServico. Esse método deverá percorrer a lista de ItemOS da OrdemServico e totalizar o valor da OrdemServico conforme cada um dos serviços adicionados na OS. Caso o atributo **desconto** (um índice percentual) da classe OrdemServico tenha um valor diferente de zero, esse método, ao final da execução, deverá calcular o desconto em relação o valor total da ordem.
7. Na classe MainApp implemente as rotinas para criar objetos que permitam a criação de uma OrdemServico, isto objeto:
    - Objeto Veiculo (com Modelo, Categoria, Marca e Cor e Cliente)
    - Objetos Servico
    - Objeto OrdemServico com ItemOS
8. Na classe MainApp, implemente um método print que receba como parâmetro um objeto do tipo OrdemServico e imprima um cupom da Ordem, isto é contendo detalhes do Cupom.

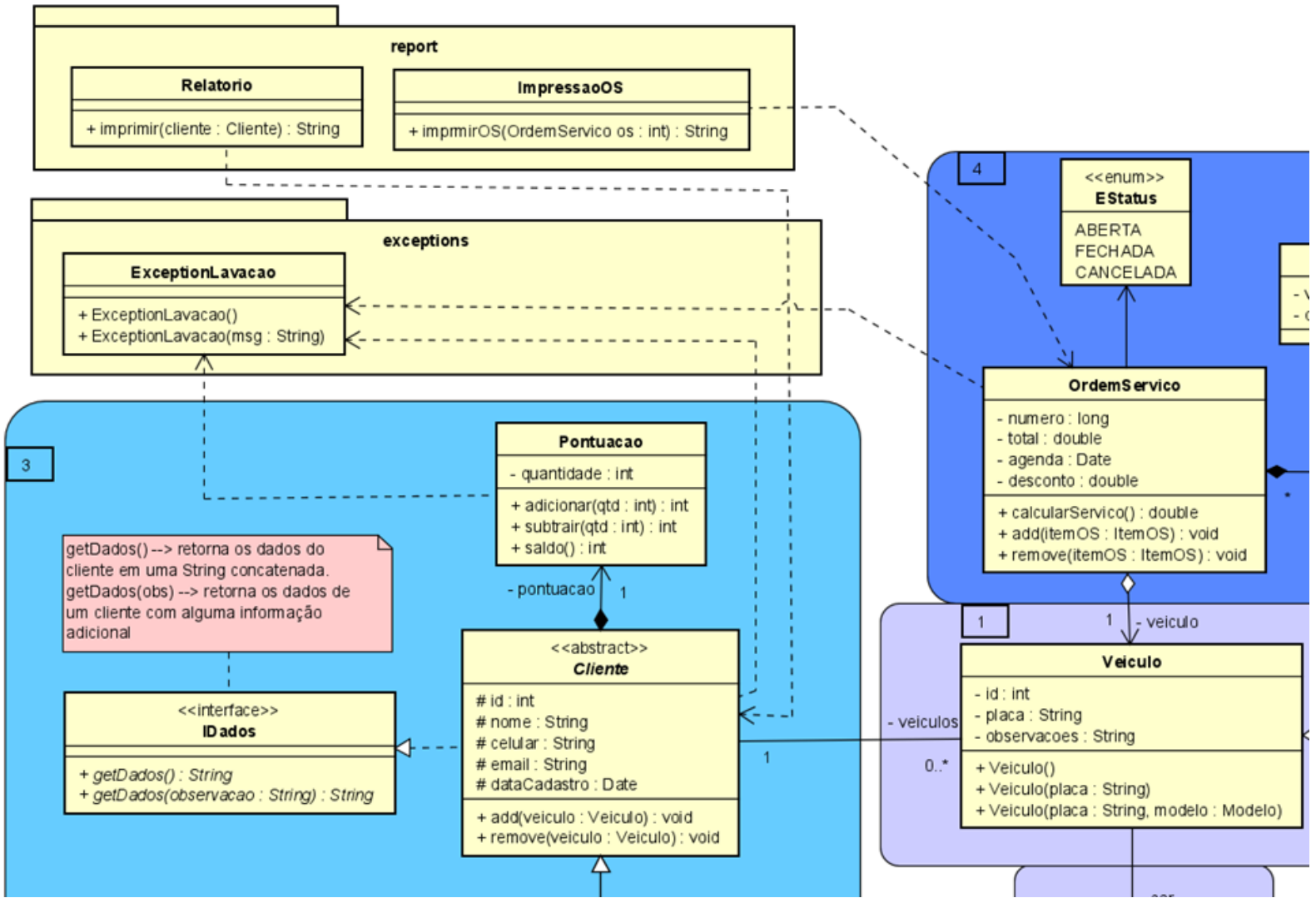
    Exemplo do protótipo do método:

```
public static void print(OrdemServico os) {
     // instruções print para mostrar os detalhes do veículo
}
```

Exemplo de cupom:

```
Número: 1    dia: 05/05/2024      status: FECHADA
Cliente: Fulano de Tal
Veiculo: ABC-1234       Modelo: S10/GM/GRANDE
================================================
ITEM DESCRICAO                                VALOR
================================================
1    LAVACAO COMPLETA                        100,00
2    CERA                                     50,00
================================================
SUBTOTAL                                     150,00
DESCONTO (10%)                               15,00
================================================
TOTAL                                        135,00
```

Dando continuidade à atividade, com base na outra parte do diagrama mostrado a seguir:



1. Implemente o conceito de Associação por Dependência por meio da classe **Relatório**.
    - O método ***imprimir*** terá o mesmo conteúdo do método ***print*** implementado na atividade Mão Na Massa 2, com a diferença agora de que essa função será delegada para a classe Relatorio, evitando um número demasiado de funções na classe MainApp.

2. Implemente o conceito de Associação por Dependência por meio da classe **ImpressaoOS**.
    - Da mesma forma, essa classe terá a implementação do método ***imprimirOS***, que nada mais é do que as instruções do método *print(ItemOS)* implementado anteriormente, tirando da classe MainApp os detalhes do print para exibir o cupom da OS.
3. Sobre exceções:
    - Implemente a classe ExceptionLavacao (declaração, lançamento e tratamento)
    - No método ***calcularServico*** declare ExcpetionLavacao e lance essa mesma exceção com a mensagem **"não há serviços na lista para serem calculados"** quando a lista de serviços estiver vazia.
    - Na chamada do método ***calcularServico***, trate a exceção ExceptionLavacao.