UNIVERSIDADE DE ÉVORA

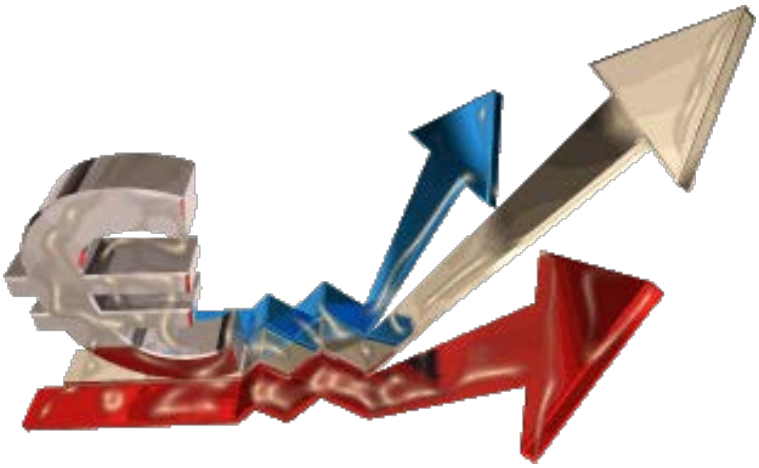
Departamento de Informática

licenciatura em Engenharia Informática

- ÁGUAS DE P2 -

(Cadeira de Programação II)

1ºAno



Eduardo Simão Ramos Almeida Costa Martins, 29035

Daniel Gonçalo Jesus Ramos, 29423

Évora, 2012

Introdução

Num programa de facturação deste género, para uma empresa de distribuição de água e saneamento, existe um certo número de variáveis relativas ao cliente a ter em conta, não contando com os escalões do tipo de serviço prestado (água, saneamento e remoção de resíduos sólidos), dado obrigatório em facturações deste género, a factura tem de ter presente o nome, morada e contacto do cliente, o tipo de consumidor, bem como o número de contribuinte do cliente.

De um ponto de vista mais técnico e referente à factura em si, temos um número desta, associado também a um período de facturação, um certo valor consumido em m^3 e o preço total a pagar pelo cliente.

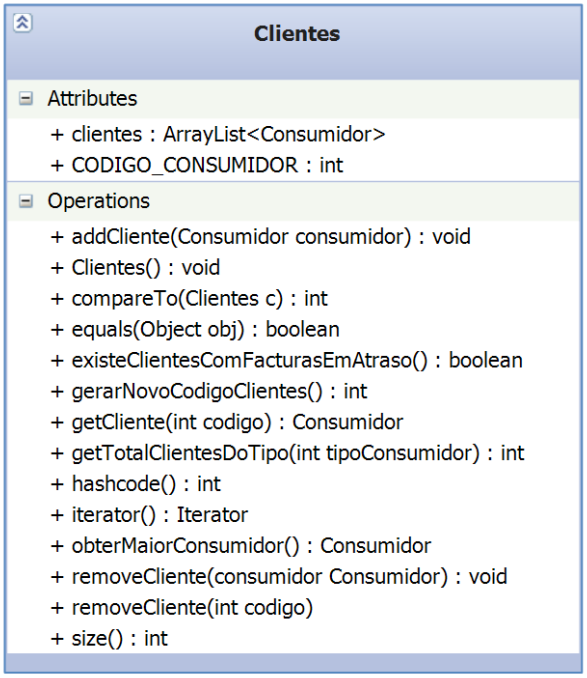
O programa deverá ser capaz de armazenar estes dados consoante o número do cliente e o consumo do mesmo, liquidar essa factura, permitir descobrir se esse cliente tem alguma factura por liquidar. E recolher dados como a média de consumo de anos anteriores e comparar com o registo de um determinado mês; saber quantos clientes de certo tipo de consumidor existem e identificar o maior consumidor.

Estrutura do programa

Classes implementadas

Classe Clientes

Esta classe faz a gestão dos clientes, permitindo adicionar e remover clientes entre outras operações requeridas para o bom funcionamento do programa.



Métodos a destacar:

-**addCliente:** permite adicionar consumidores, caso o consumidor já exista na base de dados é lançada a excepção ConsumidorJaExisteNaDBException;

-**removeCliente:** permite remover clientes dado o consumidor ou o código do consumidor. Caso o consumidor não exista na base de dados é lançada a excepção ConsumidorNaoExisteException;

-**existeClientesComFacturasEmAtraso:** verifica se existe clientes com facturas em atraso varrendo a lista dos mesmos, caso exista algum retorna true, caso não exista retorna false;

-**getTotalClientesDoTipo:** obtém o total de clientes de um determinado tipo retornando um integer com o total dos clientes desse mesmo tipo;

-**obterMaiorConsumidor:** obtem o maior consumidor, retorna o primeiro consumidor encontrado sempre que haja mais de um consumidor com o mesmo consumo. Varre a lista de consumidores e se o consumo for menor do que o do consumidor actual a ser verificado, guarda o consumo do mesmo, avançando para o seguinte consumidor;

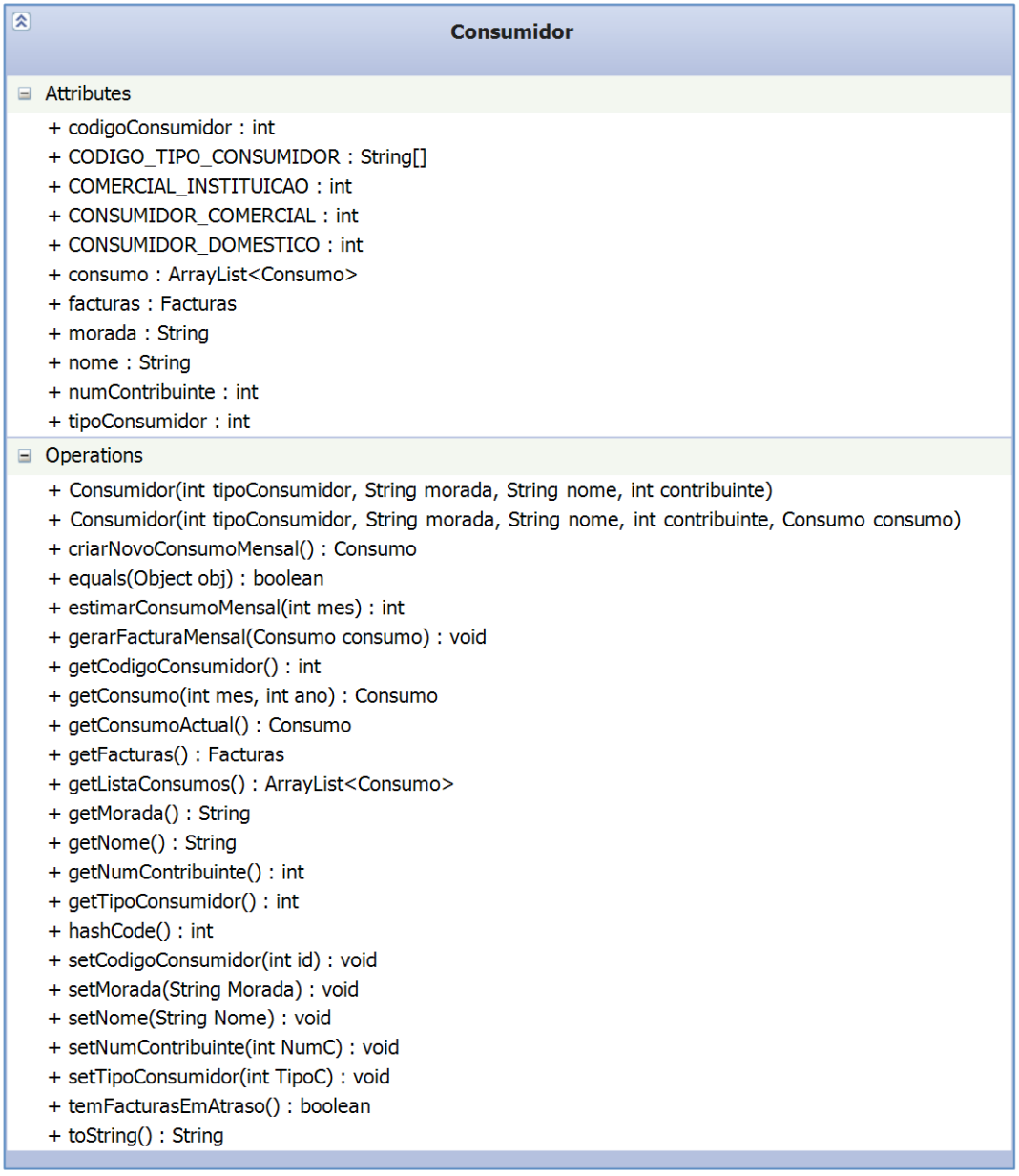
-**gerarNovoCodigoCliente:** gera um novo código para um novo consumidor, incrementando o valor base em 1.

-**getCliente** permite obter o consumidor baseado no seu código. Caso o consumidor não exista é lançada a excepção ConsumidorNaoExisteException.

-**size:** retorna a quantidade de clientes que existem na database.

Classe Consumidor

A classe contém todos os dados referentes ao consumidor, entre outras operações como por exemplo gerar facturas.



Métodos a destacar:

-**getConsumo:** obtem o consumo baseado num mês e ano especifico. Caso o consumo não exista, é lançada a excepção ConsumoNaoEncontradoException

-**getConsumoActual:** obtém o consumo do mês em que nos encontramos, para tal faz um varrimento de todos os consumos retornando aquele que que for igual a este mesmo mês e ano, caso o consumo não exista é criado um novo consumo para o mês em questão;

-**criarNovoConsumoMensal**: a partir da data actual cria um novo consumo mensal;

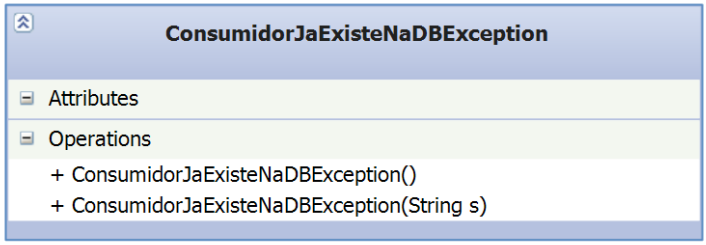
-**gerarFacturaMensal**: dado um consumo do respectivo consumidor gera uma factura detalhada dos serviços com os respectivos preços, para tal é invocado o método gerarFacturaMensal apartir da classe facturas que está instanciada dentro da classe Consumidor;

-**temFacturasEmAtraso:** verifica se o consumidor em questão apresenta alguma factura em atraso, varrendo a lista de facturas de cada consumidor e verificando se cada factura se encontra liquidada. Retorna verdadeiro se houver facturas em atraso ou falso se não houver;

-**estimarConsumoMensal:** obtém uma média do consumo de um determinado mês para o mesmo consumidor desde a sua criação.

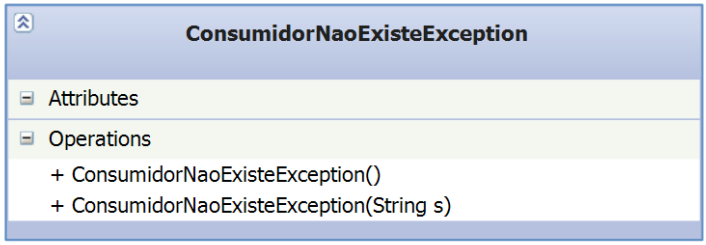
Classe ConsumidorJaExisteNaDBException

Implementa a excepção ConsumidorJaExisteNaDBException que é usado na classe Consumidor.



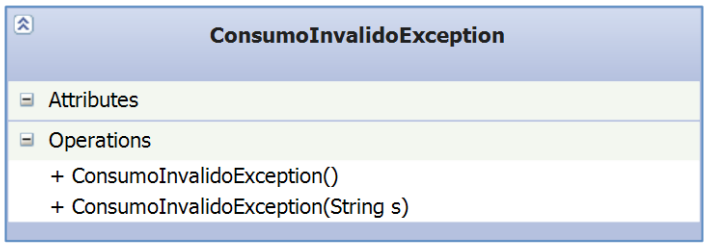
Classe ConsumidorNaoExisteException

Implementa a excepção ConsumidorNaoExisteException que é usado na classe Consumidor.



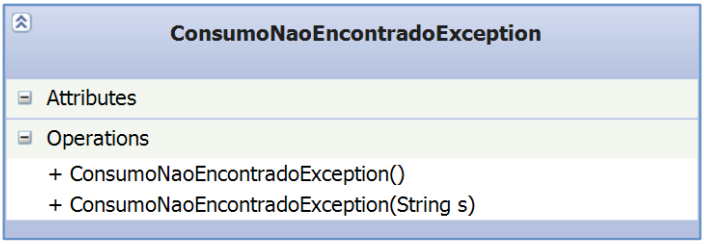
Classe ConsumoInvalidoException

Implementa a excepção ConsumoInvalidoException que é usado na classe Consumo.



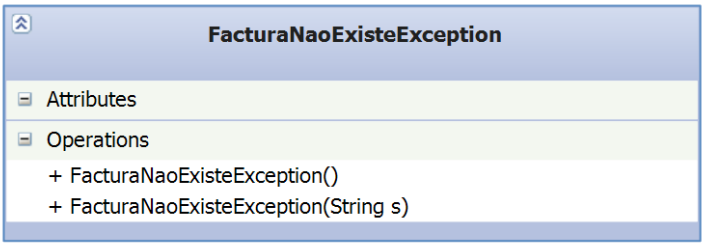
Classe ConsumoNaoEncontradoException

Implementa a excepção ConsumoNaoEncontradoException que é usado na classe Consumo.



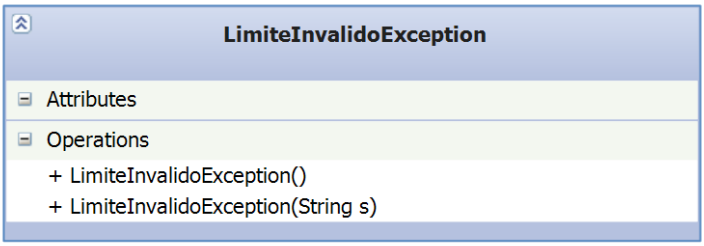
Classe FacturaNaoExisteException

Implementa a excepção FacturaNaoExisteException que é usado na classe Facturas



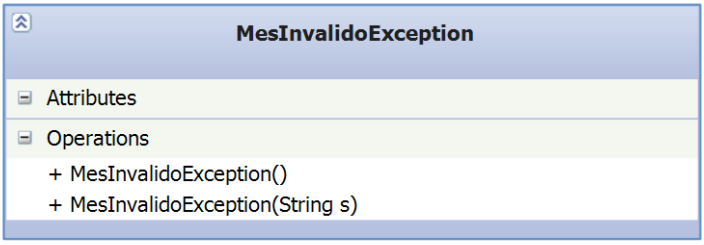
Classe LimiteInvalidoException

Implementa a excepção LimiteInvalidoException que é usado na classe Escalao



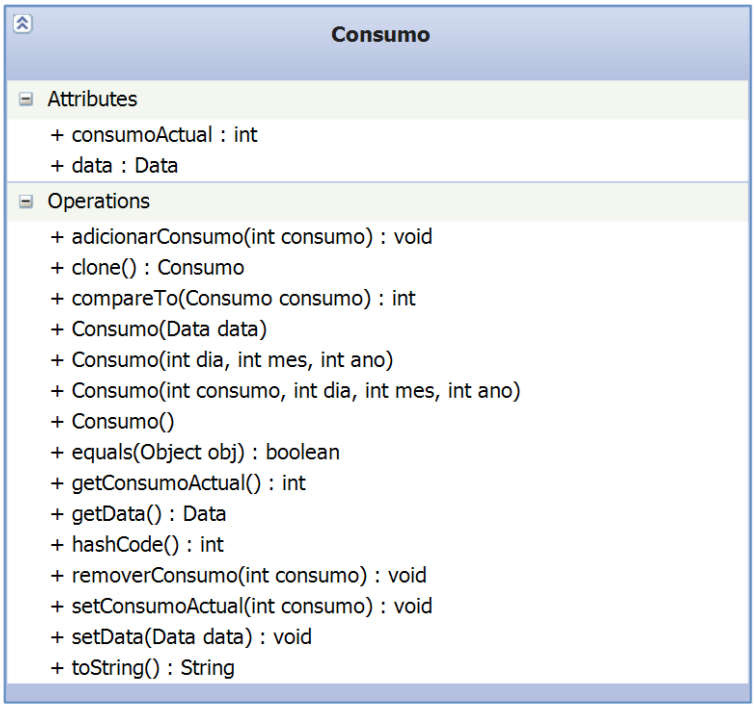
Classe MesInvalidoException

Implementa a excepção MesInvalidoException que é usado na classe Data



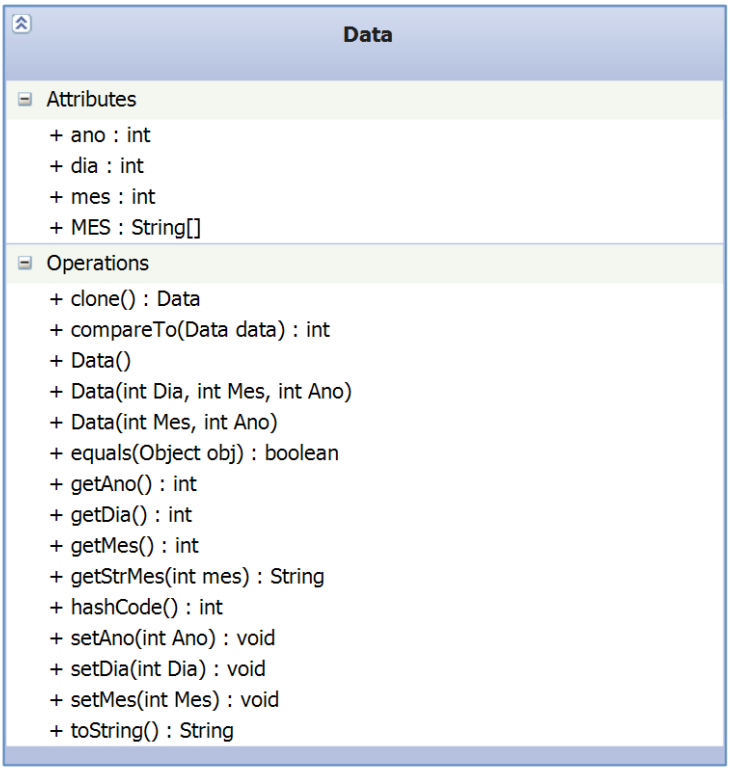
Classe Consumo

Classe responsável por guardar um consumo específico relativamente a um determinado mês.



Classe Data

Responsável por guardar a data do período actual de facturação respectivo a um consumo mensal por cliente.



Métodos a destacar:

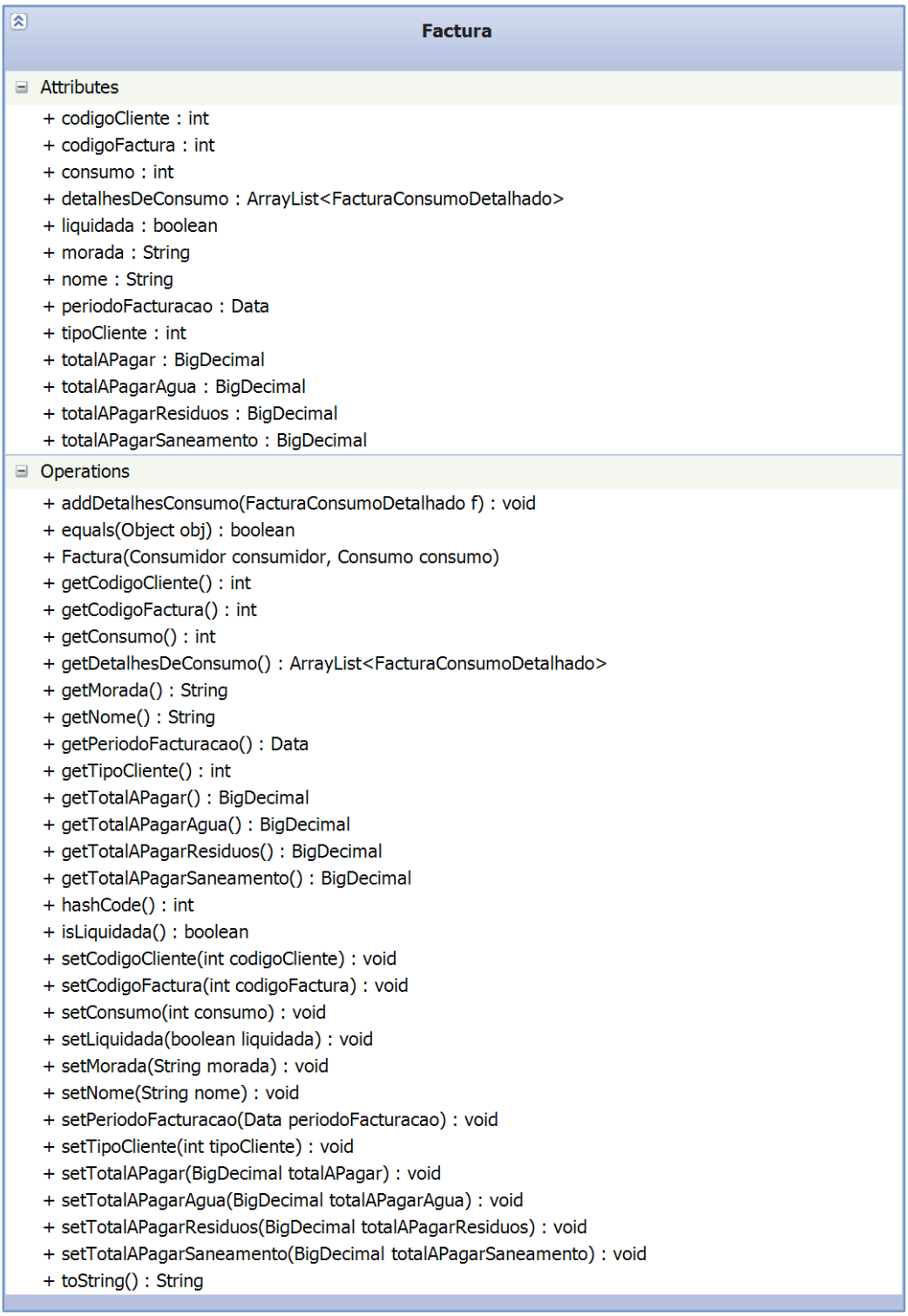
-getStrMes: obtém a string referente a um mês. Caso o mês seja inválido, lança a excepção MesInvalidoException.

Classe Escalao

Guarda um escalão específico sendo atribuído o limiteSuperior, o limiteInferior, e o respectivo preço.

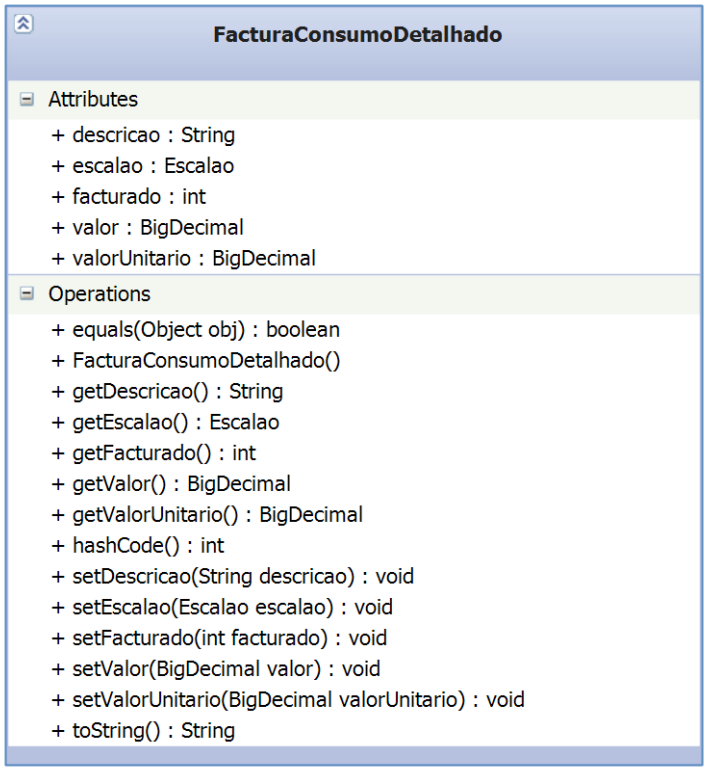
Classe Factura

Implementa o interface IFactura contendo todos os dados do consumidor, o valor a pagar respectivamente a cada serviço bem como os detalhes dos consumos de cada serviço delimitado por escalões.



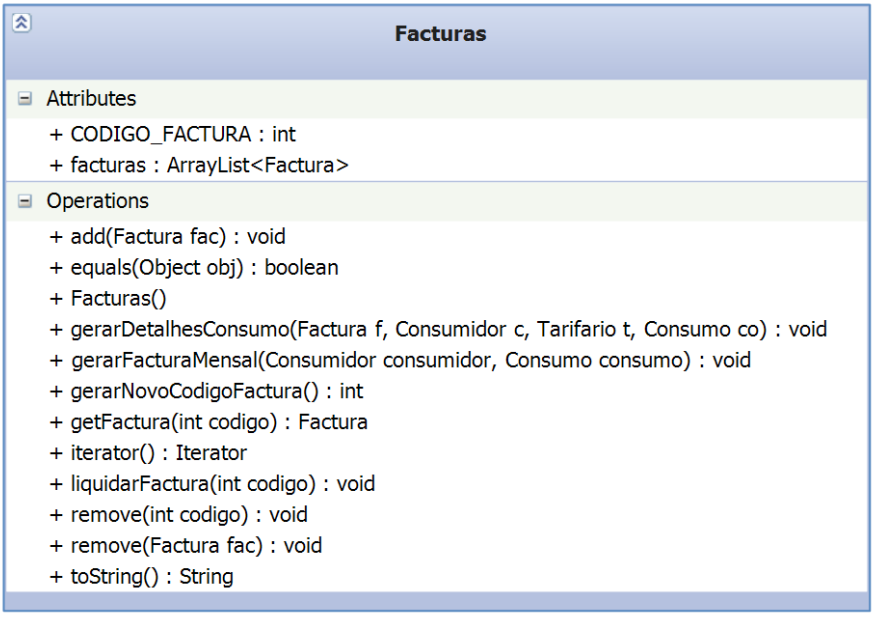
Classe FacturaConsumoDetalhado

É responsável por guardar o consumo respectivo a um serviço delimitado pelo seu próprio escalão. Inclui também o valor unitário e o valor total a pagar de acordo com o escalão do serviço especificado.



Classe Facturas

É responsável pela gestão de todas as facturas relacionadas com um determinado consumidor, bem como gerar a respectiva factura referente a um consumo. Optou-se por incluir o gestor de facturas dentro da própria classe consumidor por uma questão de eficiência porque podia-se ter criado um gestor único para as facturas o que tornaria o varrimento das facturas muito mais lento, visto que teria que varrer todas as facturas de todos os clientes sempre que se quisesse obter dados de uma factura se bem que em termos de acessibilidade seria melhor.



Métodos a destacar:

-**gerarFacturaMensal:** gera uma factura para um determinado consumidor e respectivo consumo. A factura a ser gerada e os respectivos escalões são determinados pelo tipo de consumidor(Doméstico, Comercial, Instituição).

-**liquidarFactura**: permite liquidar uma factura baseado no seu código.

-**gerarDetalhesConsumo**: gera o consumo de cada serviço em detalhe baseado no seu escalão bem como calcular o respectivo preço.

Calculo do consumo facturado: Para calcular o consumo facturado por cada escalão do respectivo serviço, é efectuado um varrimento do limite superior até ao limite inferior desse mesmo escalão e calcula-se a diferença do limite superior actual menos o consumo total. Caso a diferença seja negativa é porque existe consumo facturado para esse mesmo limite, sendo o valor facturado incrementado em 1. Se por alguma razão o consumo for superior ao consumo máximo permito, ocorrerá uma excepção. O que não deverá acontecer visto que é praticamente impossível atingir tais valores num mês.

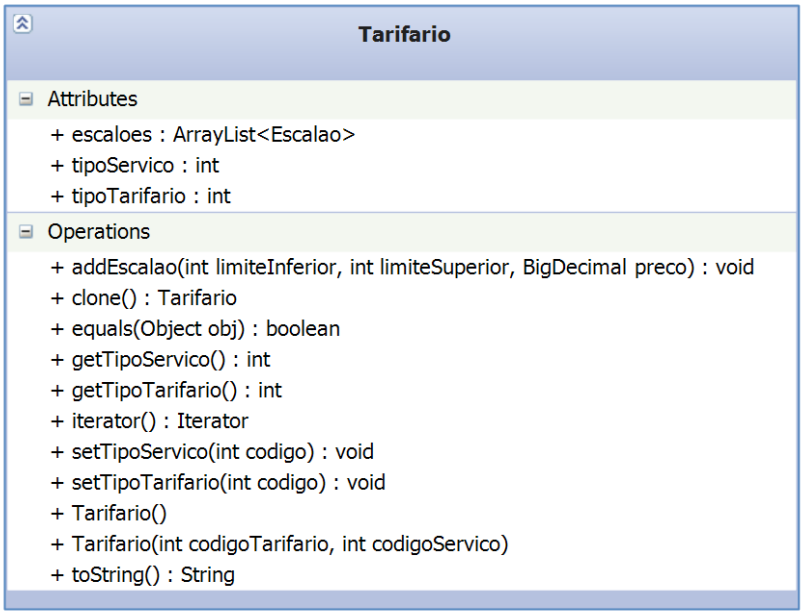
Caso exista consumo facturado para um determinado escalão, será calculado então o respectivo preço, sendo a transacção final o valor facturado, multiplicado pelo valor unitário.

Optou-se por usar BigDecimal por se tratar de valores monetários que devido aos arredondamentos é a melhor opção em comparação a usar double ou float devido aos erros cometidos ao se fazer operações com este tipo de dados. Neste caso os valores serão sempre arredondados em duas casas decimais usando o método setScale da classe BigDecimal.

Para finalizar, a transacção final calculada irá então ser adicionada ao total a pagar bem como ao total a pagar por cada serviço em separado.

Classe Tarifario

Contem informações acerca do tipo de tarifário, serviço, bem como os seus respectivos escalões.



Interfaces do programa

Interface ITarifario

Contem a estrutura necessária para a implementação básica de um tarifário

Interface IFactura

Contem a estrutura necessária para a implementação básica de uma factura.

Conclusão

Embora o programa não fosse deveras difícil, sem dúvida que a parte mais difícil será sempre encontrar uma estrutura eficiente e limpa para que se possa implementar o programa em questão, bem como modifica-lo futuramente sem grandes dificuldades. Pensamos que a estrutura implementada foi bem adequada, embora algumas partes provavelmente poderiam ser implementadas de maneira diferente ou até melhorar alguns aspectos no que toca à gestão de erros/excepções nalguns métodos implementados. No entanto pensamos que os requisitos impostos para este trabalho pratico foram atingidos tendo em conta a matéria leccionada durante o semestre e que sem dúvida que daqui para a frente só poderá melhorar.