Projeto de Programação Orientada aos Objetos
Grupo 57

Diogo Braga A82547 João Silva A82005
Ricardo Caçador A81064

31 de Janeiro de 2019

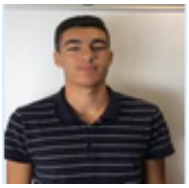
Resumo

Este documento apresenta o projeto de Programação Orientada aos Objetos, do curso de Engenharia Informática da Universidade do Minho.

O projeto baseia-se na criação de uma plataforma de disponibilização aos contribuintes da informação referente às faturas que foram emitidas em seu nome.

(a) Diogo Braga

(b) João Silva



(c) Ricardo Caçador

Conteúdo

1 Introdução

Este trabalho tem por base a criação de uma aplicação denominada de **JavaFatura**. Nesta é possível emitir e consultar faturas, entre muitas outras variantes como, por exemplo, consultar toda a informação dos Contribuintes, associar despesas dos mesmos, ou então validar os diferentes géneros de faturas.

A Secção 2 indica a arquitetura de classes proposta no projeto, a Secção 3 aborda as estruturas de dados utilizadas, a Secção 4 apresenta um manual de utilização da aplicação, e a Secção 5 aborda as principais decisões tomadas pelo grupo. O relatório termina com conclusões na Secção 6, onde é também apresentada uma análise final do projeto.

2 Arquitetura de Classes Proposta

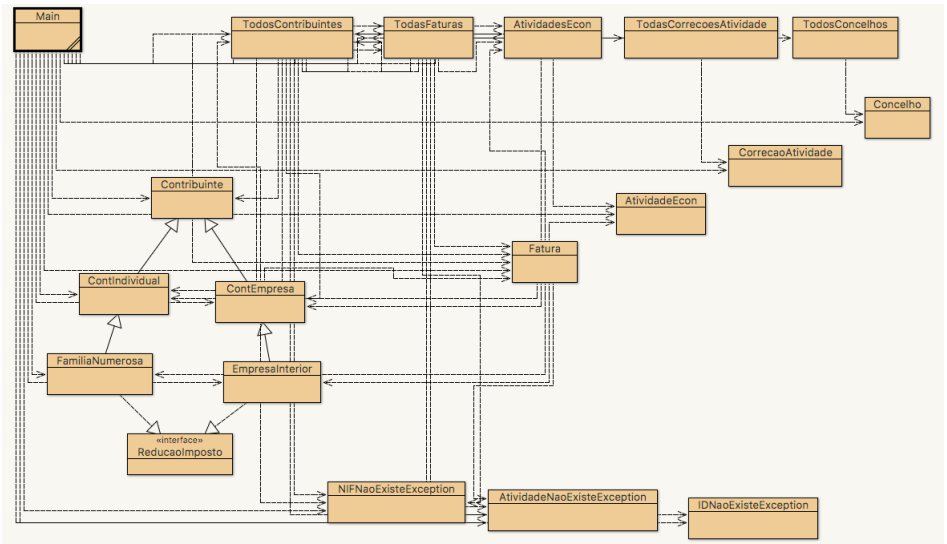


Figura 2: Arquitetura de Classes

3 Estruturas de Dados

- Classe **Main** que alberga todas as funções necessárias para o funcionamento do programa.
- Classe **TodosContribuintes** que contêm um **Map** em que as Keys são os **Nif**'s dos Contribuintes e os Values são as instâncias da classe **Contribuinte**:
  - Map <Integer,Contribuinte>**todos**
- Classe **TodasFaturas** que contêm uma **List** em que os elementos da mesma são as instâncias da classe **Fatura**:
  - List <Fatura>**faturas**
- Classe **AtividadesEcon** que contêm um **Map** em que as Keys são as **Atividade**(s) **Economica**(s) e os Values são as instâncias da classe **AtividadeEcon**:
  - Map <String,AtividadeEcon>**atividades**
- Classe **TodasCorrecoesAtividade** que contêm um **Map** em que as Keys são os **Id**'s das Faturas e os Values são as instâncias da classe **CorrecaoAtividade**:
  - Map <Integer,CorrecaoAtividade>**correcoes**
- Classe **TodosConcelhos** que contêm uma **List** em que os elementos da mesma são as instâncias da classe **Concelho**:
  - List <TodosConcelhos>**concelhos**
- Classe **Contribuinte**, na qual as variáveis de instância são:
  - int **nif**
  - String **email**
  - String **nome**
  - String **password**
  - double **euros** (o que o contribuinte individual gastou/ o que a empresa faturou)
- Classe **Fatura**, na qual as variáveis de instância são:
  - int **id**
  - int **nif_empresa**
  - String **nome**
  - LocalDate **data**
  - int **nif_cliente**
  - String **descricao**
  - String **natureza** (atividade económica)

  - double **valor**
  - boolean **validez** (verifica se a fatura já foi validada)

- Classe **AtividadeEcon**, na qual as variáveis de instância são:
  - String **atividade**
  - double **coeficiente** (coeficiente ligado a cada atividade económica para efeitos de dedução fiscal)

- Classe **CorrecaoAtividade**, na qual as variáveis de instância são:
  - int **id_fatura**
  - String **antiga_atividade**
  - String **nova_atividade**

- Classe **Concelho**, na qual as variáveis de instância são:
  - String **nome**
  - double **coeficiente**

- Subclasse **ContIndividual** de Contribuinte, na qual as variáveis de instância acrescidas são:
  - int **agregado_familiar**
  - List <Integer>**nifs_agregado**
  - double **coeficiente_fiscal**
  - List <String>**codigos** (códigos das atividades económicas que contam para efeitos de dedução)

- Subclasse **ContEmpresa** de Contribuinte, na qual as variáveis de instância acrescidas são:
  - List <String>**codigos** (códigos das atividades económicas que contam para efeitos de dedução)

- Subclasse **FamiliaNumerosa** de ContIndividual que implementa ReducaoImposto na qual as variáveis de instância acrescidas são:
  - int **dependentes** (número de dependentes de um agregado familiar)

- Subclasse **EmpresaInterior** de ContEmpesa que implementa ReducaoImposto na qual as variáveis de instância acrescidas são:
  - double **concelhoInterior**
  - double **coeficiente** (coeficiente ligado a cada empresa para efeitos de dedução fiscal)

- Interface **ReducaoImposto** em que o método presente é:
  - double **reducaoImposto** ()

- Classe para Exceção **NIFNaoExisteException**.

- Classe para Exceção **IDNaoExisteException**.

- Classe para Exceção **AtividadeNaoExisteException**.

4 Manual de Utilização

Bem-vindo ao JavaFatura! Após abrir o programa um contribuinte individual ou de empresa pode efetuar o login ou, no caso de ainda não existir na base de dados, registar-se.

No caso do contribuinte **individual**, após efetuar o login, o mesmo terá duas opções: verificar despesas, onde poderá ver todas as despesas feitas por si mesmo ou pelo seu agregado familiar, validar faturas e corrigir a natureza da atividade económica das mesmas se possuir faturas em empresas com mais de uma atividade económica.

Os contribuintes **empresariais** poderão, após fazer o seu login, efetuar 5 ações diferentes dentro do nosso JavaFatura como: emitir faturas a determinados contribuintes individuais, ver o total faturado pela empresa num período de tempo, a lista de todas as faturas da empresa ordenadas por data ou por valor, a lista das faturas de um cliente num intervalo de tempo à escolha e a lista das faturas de um cliente ordenadas por valor decrescente.

Existe ainda a possibilidade de entrar na aplicação sendo o **administrador** (através do NIF igual a 1), onde as funcionalidades tem propósitos bem diferentes das funcionalidades dos contribuintes individuais ou de empresa. O **administrador** pode executar 4 ações, tais como: adicionar novas atividades económicas, ver quem são os 10 contribuintes mais gastadores de todo o sistema, saber quais são as N empresas que mais faturam e por último rastrear todas as correções efetuadas pelos contribuintes relativamente à natureza da fatura.

5 Decisões Importantes

Ao longo da realização do projeto proposto de POO surgiram-nos alguns momentos de indecisão onde tivemos de tomar de algumas decisões relativas ao futuro do projeto, tais como:

- Colocar um parâmetro **ID** em cada fatura para possibilitar ao **Contribuinte Individual** escolher qual das faturas quer validar/corrigir, consoante os ID das faturas que lhe correspondem.
- Retirar o parâmetro **Dedução** dos **Contribuintes empresariais** proposto no enunciado, pois não encontramos uma maneira de o relacionar com o modelo da nossa aplicação. Apenas as **Empresas do Interior** têm este parâmetro pois nelas é utilizado para calcular a redução de imposto.
- Utilizar **TreeSet**'s como estrutura de dados juntamente com **Comparators** nos requisitos básicos em que é necessário haver ordenação.
- Com a saída de novos requisitos relativos à **Família Numerosa** e à **Empresa do Interior**, decidimos criar estas duas classes que implementam uma interface com o método **reducaoImposto()**. Achamos que foi a maneira mais correta e simples de adicionar os requisitos pedidos sem alterar muito código.

6 Conclusão

Tendo em conta o que se pretendia com a criação desta plataforma e os objetivos definidos pelo grupo, consideramos que a plataforma cumpre tudo o que é necessário de forma simples e compreensível a todos os utilizadores.

Importante referir que utilizámos encapsulamento total, e além disso referir o facto do nosso programa ser reutilizável pois realizámos as implementações de tal forma que quem fizer manutenção do código não tenha que fazer grandes alterações, tendo em conta a utilização de hierarquias de classes e interfaces.

Com a realização deste trabalho aprofundamos o nosso conhecimento de POO, colocando em prática tudo o que aprendemos nas aulas teóricas e práticas. Através dos vários obstáculos que fomos encontrando na realização do projeto expandimos também o nosso conhecimento noutras áreas da linguagem Java, como as bibliotecas que trabalham com ficheiros.