Programação Orientada aos Objetos
Imoobiliaria

André Rodrigues Freitas A74619
Carlos Daniel Leitão da Silva Vieira A73974
Sofia Manuela Gomes de Carvalho A76658

21 de Maio de 2016



Conteúdo

1 Introdução

Este projeto pretende que se desenvolva uma aplicação que permita gerir os imóveis de uma agência imobiliária. Trata-se de um processo em que deve ser possível abranger várias opções desde a criação de um imóvel até ao registo da sua venda. As várias operações efetuadas aos imóveis, que se encontram sob a forma de anúncio, são efetuadas pelos utilizadores da aplicação que podem ser vendedores ou compradores.

Os compradores podem pesquisar imóveis, não precisando de estar registados na aplicação, ou marcar um imóvel como favorito, sendo necessário estar registado e autenticado. Os vendedores podem inserir, consultar e remover anúncios de imóveis, e também alterar o seu estado. Para além disso, podem ainda aceder a informação atualizada sobre estatísticas como o número de anúncios criados ou número de visualizações dos anúncios.

Tal como exsitem diferentes tipos de utilizadores, também existirão diferentes tipos de imóveis como moradias ou apartamentos que terão características diferentes uns dos outros.

2 Desenvolvimento

2.1 Classes

2.1.1 Imoobiliaria

Esta é a principal classe da agência ImOObiliária. É aqui que encontramos as listas de todos os utilizadores e de todos os imóveis, bem como as funções solicitadas na secção 4 do enunciado, que permitem fazer as diversas operações que os utilizadores escolham, tais como iniciar sessão ou marcar um imóvel como favorito. Algumas dessas funções irão usar as variáveis *veSessao* e *tipoSessao* que criamos para mais facilmente sabermos se o utilizador tem sessão iniciada e se iniciou como comprador ou vendedor.

2.1.2 Utilizador

Nesta classe, temos todas as variáveis que são comuns aos compradores e vendedores, desde o seu e-mail ao tipo de utiilizador, pois as classes Vendedor e Comprador fazem *Extends* da classe Utilizador. Possui, portanto, os métodos que alteram e que devolvem essas variáveis. Consoante a String *tipoUtilizador*, cada utilizador terá acesso a outras variáveis extra, que são as diferentes listas de imóveis dos compradores e dos vendedores.

2.1.3 Comprador

Esta classe é utilizada quando a String *tipoUtilizador* é igual a "Comprador". Como faz *Extends* da classe Utilizador, os compradores serão definidos por um e-mail, um nome, uma password, uma morada, uma data de nascimento e o respetivo tipo de utilizador (variáveis estas definidas, portanto, na classe Utilizador).

Sendo um "Comprador", terá acesso, ainda, a uma lista de imóveis onde

serão guardados os seus imóveis favoritos.

Nesta classe foram declarados todos os construtores, bem como métodos de instância que permitem pesquisar imóveis dando como parâmetro qualquer uma das variáveis de instância definidas no utilizador. É possível também verificar se um imóvel já está registado na imobiliária, se já pertence à lista de favoritos, inserir um imóvel favorito, remover um imóvel favorito, contar o número total de imóveis favoritos, entre outros. Tem também declarados os métodos *clone*, *equals* e *toString*, que devolve uma representação textual do comprador.

2.1.4 Vendedor

Quando a variável *tipoUtilizador* iguala a String "Vendedor", o utilizador tem acesso a mais duas variáveis de instância: um ArrayList de imóveis, que guarda todos os imóveis que estão vendidos, e um ArrayList de imóveis com todos os imóveis nos quais a variável estado está definida como "Em venda". Alguns dos métodos que podemos encontrar nesta classe: verificar se um imóvel está à venda, se está vendido, inserir um novo imóvel para venda na lista dos imóveis que estão à venda, inserir um novo imóvel na lista dos vendidos, consultar um imóvel, alterar o seu estado, entre outros. Estão também definidos os métodos *clone*, *equals* e *toString*.

2.1.5 Imovel

Na classe Imovel encontramos as variáveis que são comuns a todos os imóveis, independentemente do seu tipo. Assim, temos informações sobre a rua em que se situam, o preço pedido para venda e o preço mínimo que o proprietário pode aceitar para vender que não pode ser apresentado aos compradores. Cada imóvel possui também um código que o identifica e o seu tipo, como por exemplo, moradia ou apartamento, e ainda o e-mail do vendedor desse imóvel. Definimos ainda uma variável *visualizacoes* que nos fornece o número de visualizações de um anúncio de um certo imóvel.

É possível encontrar nesta classe, além dos métodos *clone* e *equals*, que

verifica a igualdade com outro imóvel, o método qu devolve sobre a forma de uma string toda a informação atual sobre um imóvel (*toString*).

2.1.6 Terreno

A classe Terreno também tem acesso a todas as variáveis do Imovel. Além destas, o Terreno é definido também pelo seu tipo de construção (apropriado para construção de habitação ou apenas para a construção de armazéns), o diâmetro das canalizações, o número de kWh máximo suportado pela rede elétrica e o acesso à rede de esgotos.

Nesta classe, encontram-se declaradas as funções *clone*, *equals* e *toString*.

2.1.7 Loja

Nesta classe, mais uma vez, temos acesso a todas as variáveis da classe Utilizador, mas também à área da loja, ao tipo de negócio dessa loja, à possibilidade de ter WC ou não e ao número da porta. Existem algumas lojas que podem ser consideradas habitáveis, mas no entanto, definimos essas lojas na classe LojaHabitavel, sendo que contêm também as variáveis da classe Apartamento.

2.1.8 Habitavel

A classe Habitavel é uma classe para os imóveis que são habitáveis. Faz, por isso, *Extends* da classe Imovel.

Fazem parte dos habitáveis as moradias, os apartamentos e as lojas que são habitáveis, fazendo cada uma dessas classes *Extends* da Habitavel para que isso seja possível.

2.1.9 Moradia

A Moradia é uma classe especial de Imovel, pois é habitável. Faz *Extends* de Habitavel que por sua vez faz *Extends* da classe Imovel, para que as moradias sejam definidas com todas as variáveis de um imóvel (o seu código, o tipo de imóvel que neste caso é "Moradia", a rua em que está situado, o estado (em venda ou vendido), o preço pedido e o preço mínimo pedido pelo vendedor (variável apenas acessível pelos vendedores), o número de visualizações e o mail).

2.1.10 Apartamento

Aqui temos definido um apartamento como um *Extends* da classe Habitavel, sendo que terá associado a si o tipo(simples, duplex ou triplex), a área total, o número de quartos, o número de casas de banho, o número da porta, o andar e a possibilidade de ter ou não garagem.

2.1.11 LojaHabitavel

Esta classe faz *Extends* da classe Apartamento, visto que as lojas habitáveis contêm as informações relativas aos apartamentos e definimos uma classe Habitavel que impediria de fazer *Extends* da Loja, pois as lojas aí definidas não são habitáveis. Sendo assim, acrescentamos as variáveis da loja a esta classe.

2.1.12 ImoobiliariaApp

Esta classe é a aplicação com menu em modo texto. Foram declarados o menu da aplicação e realizados métodos: o principal e os auxiliares. Dos métodos auxiliares fazem parte as funções *carregarMenus*, *carregarDados* e ainda outras funções que permitem ler o que for inserido por um utilizador na aplicação e aplicar as respetivas funções declaradas na classe Imoobiliaria.

2.1.13 Menu

Na classe Menu, está codificado um menu em linguagem Orientada Aos Objetos que permite ler e escolher a opções declaradas na classe ImoobiliariApp. Esta classe foi disponibilizada pelo regente da Unidade Curricular, tendo sido sugerido pelo professor do turno prático usá-la no projeto final.

2.1.14 UtilizadorExistenteException

Esta é a classe usada quando se tenta registar um utilizador já registado.

2.1.15 SemAutorizacaoException

Nesta classe temos a exceção usada quando o utilizador não tem autorização para uma operação, seja por não ter sessão iniciada ou por não ter a permissão adequada (comprador ou vendedor).

2.1.16 ImovelInexistenteException

Classe usada para a exceção em que um imóvel não existe na imobiliária.

2.1.17 ImovelExisteException

Esta classe é usada para a exceção em que se pretende inserir um imóvel numa lista onde ele já é elemento dessa mesma lista.

2.1.18 EstadoInvalidoException

Esta é a classe usada para a exceção em que o estado de um imóvel é inválido.

3 Conclusão

Em suma, este projeto não correu totalmente dentro do esperado. Fomonos deparando com vários obstáculos, não tendo conseguido ultrapassar alguns deles. Deste modo, entregamos tudo o que conseguimos realizar.

Para incluir novos tipos de imóveis como, por exemplo, uma quinta, teríamos de criar uma nova classe Quinta e definir as suas variáveis e construtores de forma semelhante aos outros tipos de imóveis.

As adversidades encontradas não nos permitiram ter um menu a funcionar corretamente nem os métodos *getTopImoveis* e *getConsultas*. Também não realizamos as tarefas pedidas na secção 5 nem na 7.