

Programação Orientada aos Objetos, POO

**Relatório do projeto prático**

**IMOOBILIARIA**

**Grupo de Trabalho 34**

Ana Esmeralda Fernandes A74321

Diogo Alexandre Machado A75399

Miguel Dias Miranda A74726



Rui Filipe Castro Leite A75551

Mestrado Integrado em Engenharia Informática

Maio de 16

Índice

[Breve descrição 3](#_Toc450851839)

[Arquitetura das classes utilizadas 3](#_Toc450851840)

[ImoobiliariaAPP 3](#_Toc450851841)

[Imoobiliaria 3](#_Toc450851842)

[Utilizador 4](#_Toc450851843)

[Comprador 4](#_Toc450851844)

[Vendedor 4](#_Toc450851845)

# Breve descrição

Foi proposto a elaboração de uma aplicação em JAVATM que permita a simulação e gestão de uma Imobiliária, desde o registo de Vendedores e Compradores e de vários tipos de Imóveis (nomeadamente, Moradias, Apartamentos, Lojas e Terrenos).

Para cada utilizador existem funcionalidades que vão de acordo com o seu estatuto (comprador/vendedor), possibilitando a criação de um processo que descreve o ciclo de vida de um imóvel, desde o seu anúncio até à sua venda.

No desenvolvimento da aplicação foram tidos em conta alguns dos princípios fundamentais do JAVATM:

* Encapsulamento, através da criação de métodos que permitem o acesso e modificação dos atributos de uma classe;
* Herança, através do uso de superclasses que engloba métodos comuns a determinadas classes;
* Polimorfismo, invocando métodos com a mesma designação em diferentes subclasses.

# Arquitetura das classes utilizadas

## ImoobiliariaAPP

A ImoobiliariaAPP é a classe responsável por tratar da interação entre o utilizador e a parte computacional da aplicação. Foi aqui que se implementaram os menus apresentados no Terminal e os interpretadores de comandos, bem como os métodos que permitem pré-popular a Imoobiliaria (quer através da leitura de um ficheiro de dados (ObjectStream) ou através do registo de determinados utilizadores e imóveis declarados no método initApp()).

## Imoobiliaria

A classe Imoobiliaria é quem realmente define o objeto de uma imobiliária. Existem três variáveis de instância essenciais:

* Map<String, Utilizador> utilizadores – mapeamento entre *email* e utilizador (contém todos os utilizadores na aplicação);
* Map<String, Imovel> imoveis – mapeamento entre ID (referência) e imóvel (contém todos os imóveis na aplicação);
* Map<String, Leilao> leiloes – mapeamento entre ID do imóvel e o leilão (contém todos os leilões a decorrer).

Além destas variáveis existem outras de controle, como String atualUser (*email* do utilizador com sessão iniciada), boolean online (indica se existe algum utilizador online), int count (índice quantos imóveis foram até ao momento inscritos – necessário para a criação de um ID de imóvel), Vendedor admin (administrador da aplicação, quem pre-popula a aplicação).

Nesta classe existem os métodos essenciais de computação, que permitem aceder a informações da imobiliária, tais como:

* Aceder aos utilizadores, imóveis, leilões e *email* do utilizador com sessão iniciada;
* Dado um email/ID devolver o Utilizador/Imóvel correspondente;
* Iniciar/fechar sessão e registar um utilizador;
* Aceder à lista dos imóveis com determinadas especificações (até um preco, até N consultas ou que implementam a *interface* Habitável);
* Aceder à lista dos favoritos do (Comprador)atualUser;
* Gravar/ler o estado do programa;

## Utilizador

A classe Utilizador trata-se de uma superclasse abstrata cujas variáveis de instância são comuns a Vendedores e Compradores, como o Nome, Email, *Password*, Morada e Data de Nascimento. Permite, portanto, que estas duas classes sejam tratadas em comum, estabelecendo a relação de herança entre elas.

### Comprador

De forma a dar resposta ao que foi solicitado, a classe Comprador possui um TreeSet<String> com os ID’s dos imóveis marcados como favorito. A escolha desta estrutura de dados foi devida ao facto de não ser possível a existência de ID’s iguais, pela facilidade e rapidez de acesso do TreeSet<> e pelo facto de não ser necessária uma correspondência entre chave-valor.

O Comprador tem acesso a:

* Definir um imóvel como favorito;
* Participar num leilão;

### Vendedor

A classe Vendedor possui umTreeSet<String> com os ID’s dos imóveis em portefólio (i.e., imóveis à venda) e no histórico (imóveis vendidos). Existe também um ArrayList<Consulta> com todas as consultas feitas por todos os utilizadores da aplicação aos imóveis em portfolio do respetivo Vendedor. A escolha do ArrayList<> deve-se ao facto de se tornar fácil a inserção, aquando de uma qualquer consulta, e da impressão ordenada no ecrã.

O Vendedor tem acesso a:

* Definir o estado de um determinado imóvel (“À venda”, “Reservado”, “Vendido”);
* Registar um imóvel;
* Aceder a todas as consultas feitas aos seus imóveis em venda;
* Aceder a um conjunto dos imóveis com mais de N consultas;
* Lançar um leilão;