POO (MiEI/LCC)

2015/2016

Enunciado do Trabalho Prático

v.1.1

Conteúdo

1 ImOObiliária

É proposto aos alunos de POO o desenvolvimento de uma aplicação que permita fazer a gestão de imóveis. Pretende-se que a aplicação desenvolvida dê suporte a todo o ciclo de vida de um imóvel numa agência imobiliária. O processo deve abranger desde a criação do imóvel no sistema (sob a forma de anúncio), até ao registo da venda. Será necessária a existência de dois tipos distintos de utilizadores, nomeadamente, os **vendedores** e os **compradores**.

Cada perfil de utilizador deve apenas conseguir aceder às informações e funcionalidades respectivas.

- Os compradores poderão:
  - pesquisar imóveis dado um conjunto de características ou o identificador do mesmo, não necessitando de estar obrigatoriamente registados na aplicação;
  - Marcar um imóvel como favorito, sendo, neste caso, necessário estar registado e autenticado no sistema.
- Os vendedores poderão:
  - inserir, consultar e remover anúncios de imóveis, bem assim como alterar o seu estado (em venda, reservado, vendido).
  - aceder a informação atualizada sobre as estatísticas respectivas (numero de anúncios criados, número de visualizações dos anúncios, número de vendas de imóveis).

2 Actores

Propõe-se a existência de dois tipos distintos de actores no sistema, que partilham a seguinte informação:

- email (que identifica o utilizador);
- nome;
- password;
- morada;
- data de nascimento.

Para além disso, cada actor tem um conjunto de dados específicos, como explicado de seguida.

2.1 Comprador

O comprador representa a pessoa que irá fazer a compra do imóvel. Caso se registe no sistema o comprador poderá definir a seguinte informação adicional:

- Lista de imóveis favoritos.

2.2 Vendedor

O vendedor é a entidade responsável pela gestão dos anúncios de imóveis para venda. O comprador deverá, assim, conter a seguinte informação adicional:

- Portfólio de imóveis em venda;
- Histórico de imóveis vendidos;

3 Imóveis

Os imóveis representam as entidades em venda no sistema. Como é sabido, existem diversos tipos de imóveis, como moradias (com diversos tipos, como isoladas, geminadas, banda, gaveto, etc.), apartamentos (existindo como simples, duplex, triplex, etc.), lojas comerciais, e terrenos para construção de moradias.

Todos os imóveis terão a si associadas a rua em que se situam, o preço pedido e o preço mínimo aceite pelo proprietário (que não deverá ser apresentado aos compradores).

3.1 Moradia

Uma moradia representa um imóvel para habitação familiar. Estas possuem diversas características relevantes, devendo ser consideradas, no mínimo:

- o tipo (isolada, geminada, banda, gaveto)
- a área de implantação
- a área total coberta
- a área do terreno envolvente
- o número de quartos e de WCs
- o número da porta

3.2 Apartamento

Um apartamento representa um imóvel inserido num prédio, como tal sem jardim. São consideradas no mínimo, as seguintes características:

- o tipo (Simples, Duplex, Triplex)
- a área total
- o número de quartos e WCs
- o número da porta e o andar
- se possui, ou não, garagem

3.3 Loja

Uma loja representa um espaço destinado a diferentes tipos de negócio. Deve ficar registada informação quanto à sua área, se possuem, ou não, WC, qual o tipo de negócio viável na loja, número da porta.

Existem, no entanto, algumas lojas que possuem parte habitacional. Para estas deverá ser registada a informação guardada para os apartamentos.

3.4 Terreno

Um terreno representa um espaço com área disponível para construção. É de notar que existem terrenos apropriados para construção de habitação, ou apenas para construção de armazéns. É ainda importante saber qual o diâmetro das canalizações (em milímetros), assim como os kWh máximo suportados pela rede elétrica, se instalados, bem como se existe acesso à rede de esgotos.

4 Requisitos básicos

Identificam-se, de seguida, os requisitos básicos que o programa deverá cumprir, indicando-se ainda o método da classe `Imoobiliaria` que os deverá implementar[1]. O programa deverá ainda fornecer uma interface de utilizador que permita acesso às funcionalidades.

- O estado da aplicação deverá estar pré-populado com um conjunto de dados significativos, que permita testar toda a aplicação no dia da entrega.

```
public static Imoobiliaria initApp()
```

- Registar um utilizador, quer vendedor, quer comprador:

```
public void registarUtilizador(Utilizador utilizador)
                            throws UtilizadorExistenteException
```

- Validar o acesso à aplicação utilizando as credenciais (email e password);

```
public void iniciaSessao(String email, String password)
                            throws SemAutorizacaoException
public void fechaSessao()
```

Vendedores (é necessário estarem previamente autenticados):

- Colocar um imóvel à venda;

```
public void registaImovel(Imovel im)
                            throws ImovelExisteException,
                                   SemAutorizacaoException
```

[1] Oportunamente serão fornecida uma classe de testes, pelo que a implementação deverá respeitar as assinaturas dos métodos indicados.

- Visualizar uma lista com as datas (e emails, caso exista essa informação) das 10 últimas consultas aos imóveis que tem para venda;

```
public List<Consulta> getConsultas()
                            throws SemAutorizacaoException
```

- Alterar o estado de um imóvel, de acordo com as acções feitas sobre ele;

```
public void setEstado(String idImovel, String estado)
                            throws ImovelInexistenteException,
                                   SemAutorizacaoException,
                                   EstadoInvalidoException
```

- Obter um conjunto com os códigos dos seus imóveis mais consultados (ou seja, com mais de N consultas).

```
public Set<String> getTopImoveis(int n)
```

Todos os utilizadores:

- Consultar a lista de todos os imóveis de um dado tipo (Terreno, Moradia, etc.) e até um certo preço.

```
public List<Imovel> getImovel(String classe, int preco)
```

- Consultar a lista de todos os imóveis habitáveis (até um certo preço)[2].

```
public List<Habitavel> getHabitaveis(int preco)
```

- Obter um mapeamento entre todos os imóveis e respectivos vendedores.

```
public Map<Imovel, Vendedor> getMapeamentoImoveis()
```

Compradores registados:

- Marcar um imóvel como favorito.

```
public void setFavorito(String idImovel)
                            throws ImovelInexistenteException,
                                   SemAutorizacaoException
```

- Consultar imóveis favoritos ordenados por preço.

```
public TreeSet<Imovel> getFavoritos()
                            throws SemAutorizacaoException
```

[2] A definição de `Habitavel` será fornecida atempadamente.

5 Leilões

A qualquer momento um vendedor pode colocar um seu imóvel em leilão. O processo de leiloar um imóvel consiste na definição de um período (em horas) durante o qual o imóvel está em venda ao público. Durante esse período, os compradores podem fazer propostas cada vez mais altas, até ao momento do fecho do leilão. Nesse momento, caso o preço da última oferta seja superior ao preço mínimo que o proprietário aceita pelo imóvel, este passa ao estado de reservado, até que a compra seja efectivada.

Esta funcionalidade deve poder ser simulada automaticamente pela aplicação. Para tal, para cada comprador deve ser possível declarar a intenção de participar num leilão, qual o valor máximo que está disposto a dar, qual o valor dos incrementos a efectuar, e qual o intervalo entre licitações.

No final é indicado qual o utilizador que vence o leilão, assim como o valor da sua venda.

É solicitada a API mínima para a **simulação** de leilões:

- Abertura de leilão:

```
public void iniciaLeilao(Imovel im, int horas)
                            throws SemAutorizacaoException
```

- Adicionar comprador ao leilão:

```
public void adicionaComprador(String idComprador,
                              double limite,
                              double incrementos,
                              double minutos)
                            throws LeilaoTerminadoException
```

- Encerrar um leilão:

```
public Comprador encerraLeilao()
```

6 Relatório

O relatório deve descrever o trabalho realizado para desenvolver a aplicação solicitada. No mínimo, devem ser abordados os seguintes pontos:

- Capa com identificação da Unidade Curricular e do grupo.
- Breve descrição do enunciado proposto.
- Descrição da arquitectura de classes utilizada (classes, atributos, etc.) e das decisões que foram tomadas na sua definição.
- Descrição da aplicação desenvolvida (ilustração das funcionalidades).
- Discussão sobre como seria possível incluir novos tipos de imóveis na aplicação.

7 Salvaguarda do estado da aplicação

O programa deve permitir que em qualquer momento se possa guardar em ficheiro a informação existente em memória sobre os utilizadores, imóveis, registos, etc. A gravação deve ser feita de forma a permitir que o estado que foi gravado seja recuperado novamente. Na altura da entrega do projecto deve ser também entregue um estado (guardado em ficheiro) que possa ser carregado durante a apresentação. Este estado deve conter dados significativos, e que permitam testar toda a aplicação.

8 Patamares de classificação do projecto

Este projecto de POO tem previstos dois patamares de dificuldade em função dos requisitos anteriormente identificados:

1. Requisitos básicos identificados na Secção 4: nota máxima 16 valores;
2. Itens anteriores mais gestão e simulação de leilões (Secção 5): nota máxima 20 valores

9 Cronograma

A entrega do projecto far-se-á de forma faseada, nas seguintes *milestones*:

1. Definição dos grupos via Blackboard. **Data Limite**: 31 de Março
2. Entrega de projecto BlueJ com uma versão inicial das declarações das classes (pelo menos com as variáveis de instância). **Data Limite**: 29 de Abril
3. Entrega final de código e relatório de projecto. **Data Limite**: 20 Maio