UFFS - Universidade Federal da Fronteira Sul
Professor Valdemar Lorenzon Junior
Programação I

Sistema Imobiliário com Programação Orientada a Objeto em Java

Arthur Kolankiewicz
João Victor Carvalho
Tainara Bernardi

26/11/2024

Índice

I - Detalhamento baseado nos acontecimentos que ocorreram no desenvolver da aplicação

I.I - Versionamento e acompanhamento do código.

Para realizar a aplicação, optamos pelo versionamento do github, por sua acessibilidade e facilidade de organização. A plataforma não nos deu muitos desafios provando o motivo de ser escolhida.

I.II - Organização

Por fins organizacionais, decidimos em equipe separar o trabalho em 3 classes, assim distribuindo-a uma para cada contribuinte. Isto, no nosso ponto de vista, foi fundamental para entendimento, possibilitando um conhecimento individual profundo de cada estudante em cada região do sistema em específico, conseguindo direcionar melhor cada função e até em caso de bugs sabermos onde está o erro e facilitar a sua correção.

I.III -

II - Objetivos que buscamos alcançar

II.I - Obviamente, atender os requisitos pedidos pelo professor

Ao iniciarmos a execução, nos reunimos para decidir o que precisávamos ter em nosso sistema. Correlacionado a isso, veio à tona a importância de atingir todas as metas, listando-as e debatendo ponto a ponto como, onde e quando faríamos.

II.II - Experiência do usuário

A todo instante, em cada classe, e muito bem testado, tentamos tratar cada detalhe para que ficasse da forma mais entendível possível. Um dos problemas que encontramos ao decorrer do desenvolvimento foi a incerteza de como quem fosse usar o sistema o usaria, tentando assim, tratar da melhor forma possível todas as entradas e valores que mesmo que não fazem sentido, podem ser inseridos sem intenção ou o *próprio professor querendo apertar um pouco a nota (Ironia)*.

III - Classe Imovel

III.I - AbsImovel

Esta classe é o que está popularizada como interface. Ela está presente no projeto mais por cumprimento de requisitos, confesso que a importância dela neste projeto não é das mais cruciais, porém ela está presente.

III.II - Imovel

A classe principal da aplicação, diga-se de passagem, por motivos de todo o restante da aplicação depender do bom funcionamento desta para atingir o principal intuito do programa. Aqui se encontra:

- Inicialização da função ***getDadosImoveis()*** que será subscrita na hora de mostrar informações.
- ***filtrarImoveis()*** que permite ao usuário filtrar o que deseja através de seus interesses, como localização ou preço.
- ***printDadosImovelComLayout()*** formata as informações que serão expostas em tela com intuito de melhor visibilidade.
- ***avaliarImovel()*** que após avaliação de algum imóvel, o próprio método já fará o cálculo e mostrará na tela.
- ***listarImoveis()*** que ao chamado, mostra em tela todos imóveis cadastrados no sistema e com auxílio da printaDadosImovelComLayout(), deixa bonito e legível.

Seus nomes são auto explicativos, e garantem a visualização dos imóveis, o acesso deles, a formatação de layout que o utilizador irá tirar suas impressões, e a possibilidade de reter outras opiniões sobre aquele imóvel, respectivamente.

III.III - Subclasses Casa, Apartamento, Comercial e Pavilhão

Derivada da classe pai Imóvel, apesar da preposição, as subclasses não são menos importantes para a qualidade da aplicação. Cada uma delas contém características únicas com base em seus nomes, alguns simples exemplos como piscina em Casa, Garagem em Apartamento, Tipo de negócio para Comercial e área de manobra para Pavilhão caracterizam suas devidas funcionalidades, diferenciando-as uma a uma, o que futuramente, permitirá a utilização de ***instanceof,*** para filtrar ou distinguir as classes.

III.IV - Métodos estáticos do Imovel

Neste contexto, métodos estáticos, mais resumidamente, são responsáveis por manter as informações de um atributo em todos

os objetos. Um deles que daremos destaque é o ***cont,*** que garante com que cada imóvel terá um id único, um chave de acesso única, facilitando a organização para chamadas de determinado imóvel. Além disso, contamos com ***listaImoveis*** e ***listaImóveisFiltrados***, que garantirão um formato de persistência de dados, como se fosse um banco de dados.

IV - Usuário

IV.I - Usuario

Principal classe do pacote (classe Pai). Além dos atributos,e os getters e setter, podemos dividir os métodos nela em dois grupos:

Operações de Usuário

- cadastrarUsuario: Cadastra um novo usuário, validando campos obrigatórios e se o email já está cadastrado. Instância a subclasse correspondente ao tipo de usuário (vendedor, comprador ou administrador).
- fazerLogin: Realiza o login do usuário, retornando o objeto correspondente se as credenciais estiverem corretas.
- editarUsuario: Permite que um usuário (ou um administrador) edite informações de um usuário. O administrador pode editar qualquer usuário; um usuário comum só pode editar suas próprias informações.
- deletarUsuario: Permite que um usuário (ou um administrador) delete um usuário do sistema. O administrador pode deletar qualquer usuário; um usuário comum só pode deletar sua própria conta.
- informacaoUsuario: Exibe informações do usuário logado. Administradores têm opções adicionais para listar todos os usuários ou filtrar por tipo.

Métodos Auxiliares

- emailJaCadastrado: Verifica se um email já está cadastrado no sistema.
- listarTodosUsuarios: Exibe todos os usuários cadastrados no sistema.
- listarUsuariosPorTipo: Exibe usuários filtrados por tipo (vendedor, comprador, administrador).
- buscarUsuarioPorId: Busca e retorna um usuário pelo seu ID.

IV.II - Subclasse Cliente

Classe que representa um cliente do sistema, estendendo a classe Usuario. Esta classe permite que o cliente adicione avaliações, calcule a média dessas avaliações e liste as avaliações feitas.

Métodos

- adicionarAvaliacao: Adiciona uma nova avaliação à lista de avaliações. Valida se a avaliação está entre 0 e 5. Após a adição, recalcula a média das avaliações.
- calcularMedia: Método privado que calcula a média das avaliações armazenadas. Se não houver avaliações, a média é definida como 0. Este método é chamado sempre que uma nova avaliação é adicionada.
- getMedia: Retorna a média das avaliações feitas pelo cliente.

IV.III - Subclasses Administrador, Vendedor e Comprador

Classe Administrador

Estende a classe Usuario e representa um administrador do sistema. A classe é usada para implementar funcionalidades de gestão de usuários e imóveis, embora não possua métodos específicos definidos.

Classe Vendedor

Estende a classe Cliente e representa um vendedor de imóveis. Suas funcionalidades incluem:

- adicionarImovel: Permite que o vendedor adicione novos imóveis, coletando informações detalhadas sobre o tipo de imóvel (apartamento, casa, pavilhão ou comercial).
- listarImoveis: Lista todos os imóveis cadastrados pelo vendedor.
- deletarImovel: Remove um imóvel da lista do vendedor.
- editarImovel: Permite a edição dos detalhes de um imóvel existente.

Classe Comprador

Estende a classe Cliente e representa um comprador de imóveis. Os principais métodos incluem:

- alugarImovel: Permite que o comprador alugue um imóvel, verificando se o vendedor existe e se o imóvel está disponível.
- comprarImovel: Permite que o comprador compre um imóvel, seguindo o mesmo princípio de validação.

VI - Agendamento

VI.I - Agendamento

Principal classe do pacote, implementa a interface InterfaceAgendamento e gerencia os agendamentos de visitas e vistorias. Além dos atributos e seus respectivos getters e setters, os métodos podem ser divididos em dois grupos:

Operações de Agendamento

- verificarDisponibilidade(int idImovel, Date dataVisita): Verifica se já existe um agendamento para um imóvel em uma data

específica. Retorna true se o imóvel estiver disponível e false caso contrário.
- agendarVisita(int idImovel, Date dataVisita): Agenda uma visita para um imóvel. Chama o método verificarDisponibilidade para garantir que não há conflito de agendamento. Caso esteja disponível, cria um objeto da subclasse Visita e o adiciona à lista de agendamentos. Caso contrário, exibe uma mensagem de erro.
- agendarVistoria(int idImovel, Date dataVisita): Agenda uma vistoria para um imóvel, criando um objeto da subclasse Vistoria e adicionando-o à lista de agendamentos.

Métodos Auxiliares

- getIdImovel(): Retorna o ID do imóvel agendado.
- getDataVisita(): Retorna a data da visita ou vistoria agendada.

VI.II - Subclasse Visita

Classe do pacote **Agendamento** que representa uma visita agendada para um imóvel. A classe **Visita** é uma subclasse de **Agendamento** e é responsável por gerenciar a criação e validação de visitas a propriedades. Assim como outras subclasses do pacote, oferece métodos para verificar a disponibilidade e realizar agendamentos seguindo critérios específicos. Seus métodos e funcionalidades podem ser divididos em dois grupos:

Operações de Agendamento

- verificarDisponibilidadeVisita(Imovel imovel, Date dataVisita): Verifica se o imóvel está disponível para a realização de uma visita em uma data específica. Retorna *true* caso o imóvel esteja apto para receber a visita e *false* caso contrário. A verificação considera:
    - Garantir que o imóvel está marcado como disponível.
    - Certificar-se de que a data da visita não é anterior à data atual.

- agendarVisitaComValidade(Imovel imovel, Comprador comprador, Date dataVisita):
  Realiza o agendamento de uma visita ao imóvel, chamando o método verificarDisponibilidadeVisita para garantir que não há impedimentos. Caso esteja disponível, a visita é agendada e registrada na lista de agendamentos. Além disso, após a realização da visita, o comprador tem a opção de:
  - Avaliar o imóvel visitado.
  - Avaliar o vendedor responsável pela propriedade.

Métodos Auxiliares

1. getIdImovel()
   Retorna o ID do imóvel vinculado à visita.
2. getDataVisita()
   Retorna a data programada para a visita.

Essa estrutura permite que as visitas sejam agendadas de forma organizada e eficiente, prevenindo conflitos de agendamento e incentivando feedback pós-visita para melhorar a experiência do comprador e a qualidade do serviço prestado.

VI.III - Subclasse Vistoria

Classe do pacote Agendamento que representa uma vistoria agendada para um imóvel. A classe Vistoria é uma subclasse de Agendamento e é responsável por gerenciar o agendamento de vistorias de forma organizada e eficiente. Ela herda a funcionalidade base da classe Agendamento e oferece um método especializado para a criação de vistorias. Seus métodos e funcionalidades podem ser organizados da seguinte forma:

Operações de Agendamento

- agendarVistoria(int idImovel, Date dataVisita):
  Método que sobrescreve o comportamento padrão do agendamento de uma vistoria. Ele chama o método correspondente na classe base Agendamento para registrar a vistoria, garantindo a reutilização da lógica centralizada de agendamento. Esse método é responsável por registrar a vistoria no sistema, associando o ID do imóvel e a data especificada.
- verificarDisponibilidadeVistoria(Imovel imovel, Date dataVistoria)
  Método responsável por validar a possibilidade de agendar uma vistoria para o imóvel em uma data específica. As verificações incluem:
  - Disponibilidade do imóvel: O imóvel deve estar marcado como disponível para a vistoria.
  - Data válida: A data da vistoria deve ser posterior ou igual à data atual.

  Retorna `true` se o imóvel e a data atenderem aos critérios e `false` caso contrário, exibindo mensagens claras para os casos de indisponibilidade ou data inválida.

Métodos Auxiliares

Os métodos auxiliares são herdados da classe Agendamento e incluem:

- getIdImovel(): Retorna o ID do imóvel relacionado à vistoria.
- getDataVisita(): Retorna a data programada para a vistoria.

Essa classe simplificada foca na reutilização da lógica já implementada em Agendamento, enquanto mantém a capacidade de expansão para funcionalidades específicas de vistorias no futuro.

VI.IV - InterfaceAgendamento

Interface que define os métodos necessários para a implementação dos agendamentos. Contém os métodos para verificar disponibilidade de data, agendar visitas e agendar vistorias. Essa interface é implementada pela classe Agendamento e serve como contrato para as subclasses Visita e Vistoria.