UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Disciplina: Programação Orientada a objetos – Turma 03A
Prof. Thiago Donizetti dos Santos

PROJETO PRÁTICO – Plataforma de Locação de Propriedades

Parte 2 – Final.

Descrição:

Dando continuidade ao projeto iniciado na Parte 1, nesta segunda fase os alunos deverão evoluir o sistema **MackNb** de locação de propriedades, aplicando os principais conceitos de Programação Orientada a Objetos (POO), tais como herança, polimorfismo, encapsulamento, classes abstratas, interfaces, sobreposição e sobrecarga de métodos, além de persistência de dados (por arquivos ou banco de dados).

Primeira Entrega – Preparação das classes

Na primeira etapa, foram entregues as primeiras classes do projeto.

O que foi entregue na primeira entrega:

- **Classe usuário:** Usuários genérico. Não define se é proprietário ou não.
  - **Atributos:** Nome, email e senha.
  - **Métodos:** Construtor(es)*, imprimir dados

- **Classe propriedade:** Propriedade genérica.
  - **Atributos:** Disponível (livre ou não), título, descrição, localização, capacidade, preço por noite e proprietário
  - **Métodos:** Construtor(es)*, imprimir dados, verificar disponibilidade.
- **Classe principal:**
  - **Atributos:** Um usuário proprietário (locador), um usuário locatário, uma propriedade.
  - **Métodos:** Main (cria os usuários e a propriedade), listar usuários, listar propriedades.
- **Classe Reserva (OPCIONAL na primeira entrega):**
  - **Atributos:** propriedade que foi reservada, usuário que fez a reserva, data de check-in, data de check-out, custo total.
  - **Métodos:** Fazer reserva e calcular custo total.

UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Disciplina: Programação Orientada a objetos – Turma 03A
Prof. Thiago Donizetti dos Santos

Agora vocês irão atualizar o projeto e criar todos os métodos, adicionar implementação do POO e persistência em banco de dados.
Para a entrega final, vocês devem acrescentar herança, polimorfismo, encapsulamento, tratamento de exceções e persistência (banco de dados e/ou arquivos) bem como outros detalhes avançados.

Requisitos Funcionais:

1) Cadastro de Propriedades: Os proprietários devem poder cadastrar suas propriedades, fornecendo informações como título, descrição, localização, capacidade, preço por noite, e fotos.
2) As propriedades **devem ser derivadas de uma classe abstrata chamada Propriedade**, podendo ser do tipo Casa, Apartamento ou Sitio.
3) Cadastro de Usuários: Os usuários devem poder se cadastrar na plataforma, fornecendo informações pessoais, como nome, email e senha. Os usuários podem ser proprietários ou clientes.
4) Os usuários da plataforma (clientes e proprietários) **devem ser derivados de uma classe abstrata genérica chamada Usuario**.
5) Reservas: Os usuários registrados podem buscar propriedades disponíveis em uma determinada data e localização e fazer reservas. A plataforma deve calcular o custo total da reserva.
6) O sistema deve permitir:
    a. Cadastro e listagem de clientes e propriedades;
    b. Reservas de propriedades por clientes;
    c. Consulta de disponibilidade;
    d. Cálculo automático do custo da reserva com base nas datas;
    e. Inicialização do sistema com dados previamente cadastrados (utilizando persistência).
7) Menu: Inicialmente deve exibido um menu solicitando que o usuário escolha entre Proprietário ou Cliente.
    a. Proprietário pode cadastrar propriedades, exibir detalhes de propriedades, e verificar quais propriedades estão alugadas.
    b. Cliente: pode consultar propriedades disponíveis para aluguel e pode alugar uma propriedade.
8) Pagamentos: Não é escopo gerenciar pagamentos.

9) O Aluno deve ter previamente cadastrado alguns proprietários, propriedades, clientes, e que estes usuários já tenham alugado algumas propriedades.

Requisitos Técnicos:

- O projeto deve ser implementado em Java.
- POO: Aplicar os conceitos de herança, polimorfismo, encapsulamento, classes abstratas, interfaces, sobreposição e sobrecarga.
- Utilize classes e objetos para modelar as propriedades, usuários e reservas.
- Persistência: Deve-se utilizar arquivos (texto, CSV ou JSON) e/ou banco de dados simples (como supabase, SQLite ou H2).
- Para a persistência, deve ser usado o padrão DAO.
- Use ArrayList para armazenar informações sobre as propriedades, usuários e reservas.
- Implemente validações de entrada de dados impedindo que dados inválidos sejam gravados.
- Faça o tratamento de exceções, por exemplo, entrada de dados inválida, tipos inválidos, entrada e saída, banco de dados, arquivos, etc.
- Crie uma interface de usuário simples (console) para interação com o sistema.

<u>Modificações (e adições) nas Classes:</u>

Classe *abstrata* Usuário:

- Atributos: nome, email, senha
- Métodos: Construtor, getters e setters para os atributos, imprimir informações (método *void imprimirDados ()* abstrato para ser sobrescrito por subclasses)

Classe Proprietário (herda de Usuário):

- Atributos: ArrayList<Propriedade> propriedades
- Métodos: Construtor, getters e setters para os atributos, void cadastrarPropriedade(Propriedade p), void listarPropriedades(), void

listarPropriedadesAlugadas (), void imprimirDados() (implementação do método da classe-pai – polimorfismo de sobreposição)

Classe Cliente (herda de Usuário):

- Atributos: ArrayList<Reserva> reservasRealizadas
- Métodos: Construtor, getters e setters para os atributos, void realizarReserva(Propriedade p, Date checkIn, Date checkOut), void listarReservas(), void listarPropriedadesDisponiveis(), void imprimirDados() (implementação do método da classe-pai – polimorfismo de sobreposição)

Classe *abstrata* Propriedade:

- Atributos: Disponível (livre ou não), título, descrição, localização, capacidade, preço por noite e proprietário (referência ao objeto Proprietário)
- Métodos: Construtor, getters e setters para os atributos, verificar disponibilidade, imprimir informações (método *void imprimirDados ()* abstrato para ser sobrescrito por subclasses), calcular preço total (double calcularPrecoTotal(int dias) abstrato para ser sobrescrito pelas subclasses)

Casa (herda Propriedade)

- Atributos: boolean possuiPiscina, double preçoPorPessoa.
- Métodos: Construtor, getters e setters para os atributos, imprimir informações (implementação do método da classe-pai – polimorfismo de sobreposição), calcular preço total (implementação do método da classe-pai – polimorfismo de sobreposição – para a casa, o preço total deve ser o preço por pessoa multiplicado pelo preço por noite.)

Apartamento (herda Propriedade)

- Atributos: int andar, double taxa.
- Métodos: Construtor, getters e setters para os atributos, imprimir informações (implementação do método da classe-pai – polimorfismo de sobreposição), calcular preço total (implementação do método da

classe-pai – polimorfismo de sobreposição – para o apartamento, o preço total deve ser a taxa mais o preço por noite vezes o número de noites.)

Sitio (herda Propriedade)

- Atributos: double areaTotal.

Métodos: Construtor, getters e setters para os atributos, imprimir informações (implementação do método da classe-pai – polimorfismo de sobreposição), calcular preço total (implementação do método da classe-pai – polimorfismo de sobreposição – para o sítio, o preço total deve ser o preço por noite vezes o número de noites).

Reserva

- Atributos: propriedade reservada, cliente que fez a reserva, data de check-in, data de check-out, custo total.
- Métodos: Construtor, getters e setters para os atributos, calcular custo total (método que chama o método correspondente da propriedade alugada)

Classe Principal (Main)

- **Atributos**:
  - ArrayList<Usuários> listaUsuarios (para permitir polimorfismo com clientes e proprietários)
  - ArrayList<Propriedades> listaPropriedades (para permitir polimorfismo com diferentes tipos de propriedades)
- **Métodos**:
  - Apresentar o menu principal
  - Realizar cadastros (propriedades e usuários)
  - Executar funcionalidades via console
  - Listar propriedades, reservas e usuários
  - Persistir e carregar dados

UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Disciplina: Programação Orientada a objetos – Turma 03A
Prof. Thiago Donizetti dos Santos

Implementação do DAO para Arquivo ou Banco de Dados

CLASSES DE DOMÍNIO / MODELO

Abstratas:

1. Usuario
2. Propriedade

Subclasses concretas:

3. Proprietario
4. Cliente
5. Casa
6. Apartamento
7. Sitio
8. Reserva

INTERFACES DAO

9. UsuarioDAO
10. PropriedadeDAO
11. ReservaDAO

IMPLEMENTAÇÕES DAO – PERSISTÊNCIA EM ARQUIVO

12. UsuarioArquivoDAO – implementa UsuarioDAO
13. PropriedadeArquivoDAO – implementa PropriedadeDAO
14. ReservaArquivoDAO – implementa ReservaDAO

IMPLEMENTAÇÕES DAO – PERSISTÊNCIA EM BANCO DE DADOS

15. UsuarioBancoDAO – implementa UsuarioDAO
16. PropriedadeBancoDAO – implementa PropriedadeDAO
17. ReservaBancoDAO – implementa ReservaDAO

SUPORTE / UTILITÁRIAS

18. DAOFactory – retorna a implementação de DAO (arquivo ou banco) com base na configuração
19. ConexaoBanco – cria conexão com o banco de dados

20. ArquivoUtil – leitura e escrita de arquivos
21. Serializador – (se usar JSON) leitura/escrita de objetos
22. ConversorDeData – conversão de datas (String ↔ Date)
23. ValidadorDeEntrada – valida entradas do usuário (opcional, mas útil)

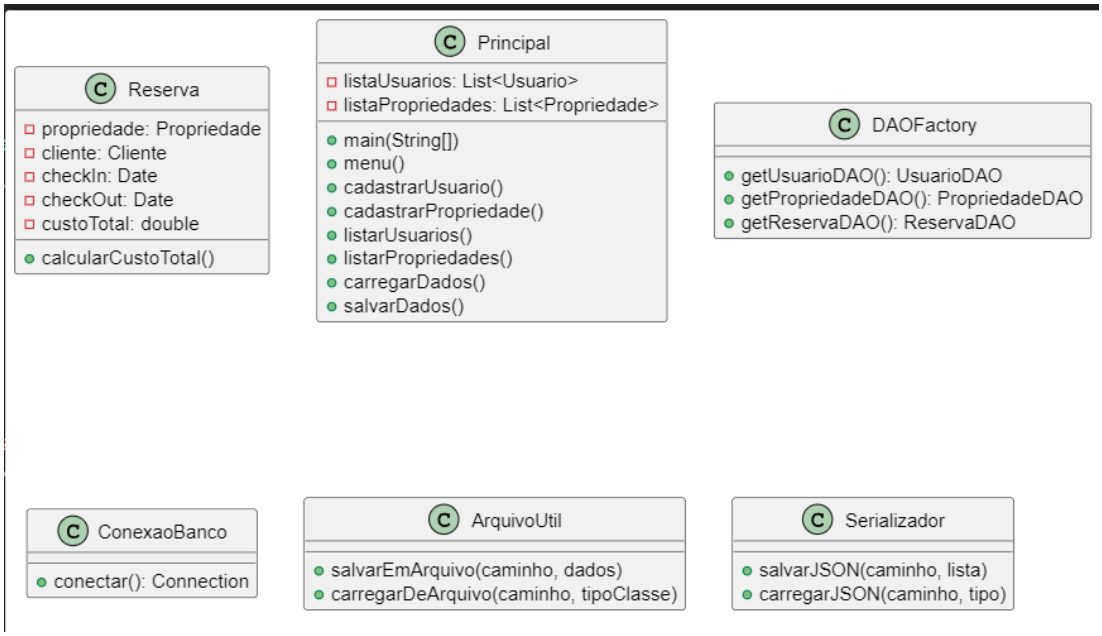
CLASSE PRINCIPAL

24. Main ou AppPrincipal – executa o sistema

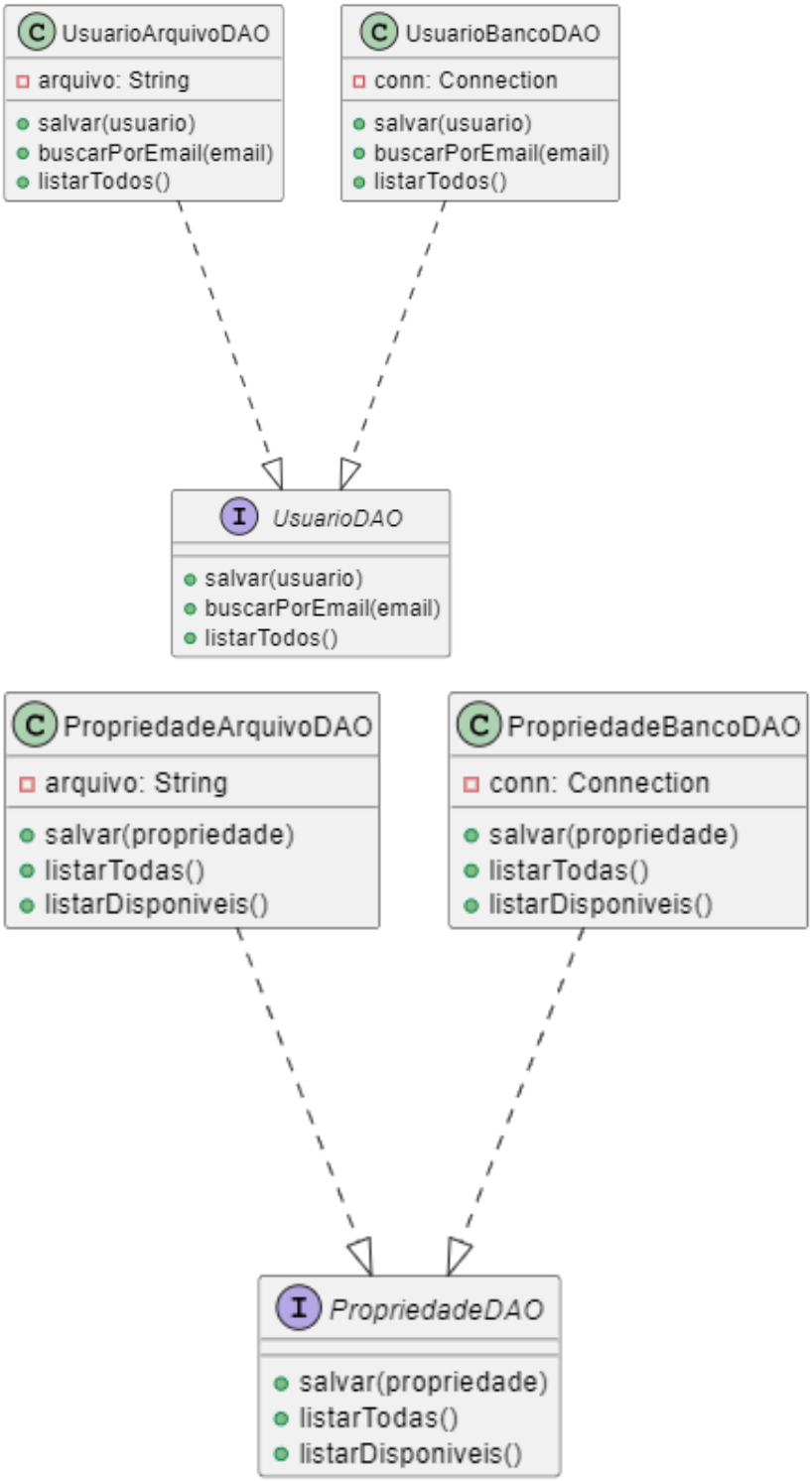
OBSERVAÇÕES:

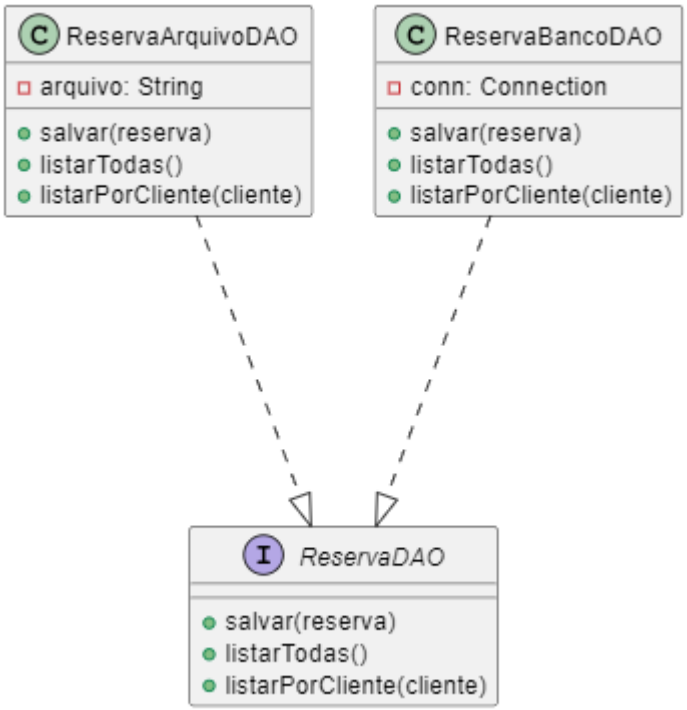
- Pelo menos uma das classes de domínio/modelo deve ter persistência em banco ou arquivo.
- Para propriedades, implemente pelo menos dois tipos, por exemplo, casa e apartamento, apartamento e sítio, casa e sítio, etc.

Sugestão de projeto:




Cliente
reservasRealizadas: List<Reserva>
realizarReserva(Propriedade, Date, Date)
listarReservas()
listarPropriedadesDisponiveis()
imprimirDados()
Proprietario
propriedades: List<Propriedade>
cadastrarPropriedade(Propriedade)
listarPropriedades()
listarPropriedadesAlugadas()
imprimirDados()
Usuario
nome: String
email: String
senha: String
imprimirDados()
Casa
possuiPiscina: boolean
precoPorPessoa: double
imprimirDados()
calcularPrecoTotal(dias: int): double
Apartamento
andar: int
taxa: double
imprimirDados()
calcularPrecoTotal(dias: int): double
Sitio
areaTotal: double
imprimirDados()
calcularPrecoTotal(dias: int): double
Propriedade
disponivel: boolean
titulo: String
descricao: String
localizacao: String
capacidade: int
precoPorNoite: double
proprietario: Proprietario
verificarDisponibilidade()
imprimirDados()
calcularPrecoTotal(dias: int): double


UsuarioArquivoDAO
arquivo: String
salvar(usuario)
buscarPorEmail(email)
listarTodos()
UsuarioBancoDAO
conn: Connection
salvar(usuario)
buscarPorEmail(email)
listarTodos()
UsuarioDAO
salvar(usuario)
buscarPorEmail(email)
listarTodos()
PropriedadeArquivoDAO
arquivo: String
salvar(propriedade)
listarTodas()
listarDisponiveis()
PropriedadeBancoDAO
conn: Connection
salvar(propriedade)
listarTodas()
listarDisponiveis()
PropriedadeDAO
salvar(propriedade)
listarTodas()
listarDisponiveis()



Sobre a entrega e apresentação:

A **ENTREGA** deve ser realizada exclusivamente pelo Moodle, na atividade correspondente até o dia 30/05/2025 às 23h59.

- Deve ser enviado um único arquivo zipado, contendo todos os códigos-fonte das classes e um relatório **(PDF ou DOCX)** que descreve quais classes foram desenvolvidas e decisões de implementação
- Cada classe deve possuir seu próprio arquivo.
- No arquivo da classe principal colocar o nome de todos os integrantes.
- Comentários em excesso (mais de 60% do código comentado) acarretará em um desconto de 20% da nota de implementação.
- Mostrar também no relatório testes de execução com ao menos o menu principal, listagem de propriedades e usuários.
- **Entregas pelo e-mail não serão aceitas**
- Entregas **atrasadas terão desconto na nota de implementação**, proporcional aos dias de atraso.
- Apenas um membro do grupo precisa enviar o arquivo

UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Disciplina: Programação Orientada a objetos – Turma 03A
Prof. Thiago Donizetti dos Santos

A **APRESENTAÇÃO** será realizada por TODOS os no dia 30/05/2025 durante o horário da aula. A ordem de apresentação será por grupos, por sorteio realizado previamente. Alunos que não apresentarem terão nota de apresentação igual à zero, mesmo que o grupo tenha apresentado.

- Os alunos farão a apresentação de forma individual, ao lado do professor que fará 4 perguntas gerais sobre o projeto. Independentemente de como as tarefas foram distribuídas entre os integrantes, todos os alunos devem conhecer e saber responder sobre qualquer parte da implementação do projeto.
- Serão avaliados se o estudante consegue responder com confiança as perguntas, e se conseguiu entender todos os aspectos do projeto, bem como consegue explicar com clareza de que forma contribuiu com o projeto.
- Alunos que não compareçam no dia da apresentação terão sua nota de apresentação igual à 0.
- O programa deve compilar e executar, caso contrário, o grupo terá nota descontada da implementação e da apresentação.

Cálculo da nota final

- **Cópias de parte ou todo no código terão nota zero.**
- Códigos feitos por IA's ou assistidos por IA's sobre os quais o grupo não saiba explicar terão nota igual a zero.

A nota final do projeto será dada pela fórmula:

0.6*NA + 0.2*NR + 0.2*NC,

onde NA denota a nota de apresentação, NR denota a nota do relatório enviado, e NC denota a nota de implementação/código-fonte.

Considerações finais:

Dúvidas sobre os requisitos funcionais e técnicos, sobre o relatório ou sobre a apresentação deverão ser enviadas ao professor até o dia 20/05/2025. Problemas em relação à clareza das informações no enunciado do projeto e nos requisitos terão até essa mesma data para serem resolvidos. A partir dessa data, será considerado que todos os alunos compreenderam os requisitos e a correção será feita considerando esse fato, não sendo aceitas reclamações e solicitações posteriores em relação ao entendimento do solicitado no projeto.