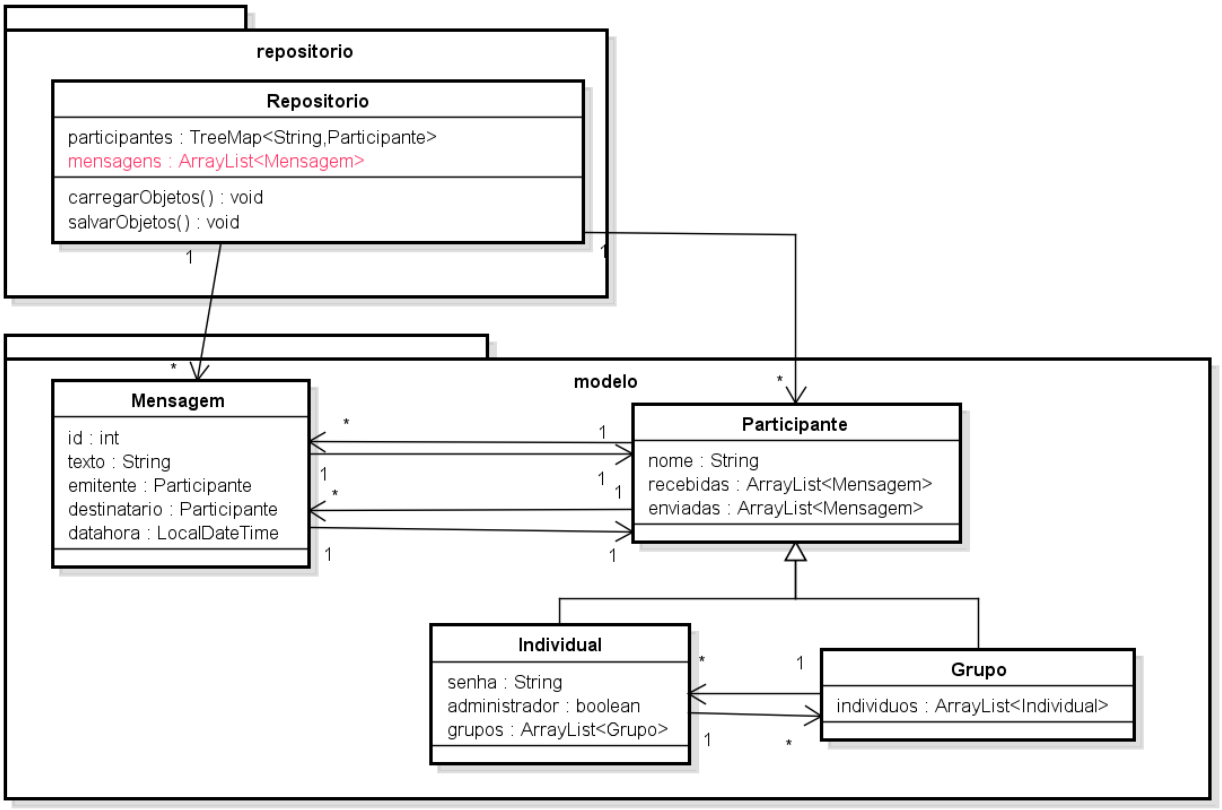
IFPB – CSTSI/POO – 2023.1
Prof. Fausto V. M. Ayres
Projeto 2
(atualizado em 30/06)

Objetivo: Desenvolver o Sistema 4TALK para que pessoas troquem mensagens entre si.

Classes do modelo de negócio e do repositório



Regras de negócio implementadas pela classe Fachada:

R1 Participantes possuem nome único, que não pode ser vazio
R2 Mensagens possuem id sequencial (1,2,3...), gerado pelo sistema, e data-hora do computador
R3 Um indivíduo não pode ter senha vazia
R4 Uma mensagem não pode ter texto vazio
R5 Uma mensagem tem um emitente (Individual) e um destinatário (Participante) que podem ser iguais ou diferentes
R6 Quando o destinatário de uma mensagem for um grupo, todos os indivíduos do grupo recebem cópias da mensagem (mesmo id e texto)
R7 Uma conversa entre um indivíduo e um participante (indivíduo ou grupo) é uma lista de mensagens, enviadas entre eles, organizada em ordem cronológica
R8 Uma mensagem excluída do sistema deve ser removida do seu emitente e do seu destinatário e, se o destinatário for um grupo, as suas cópias devem ser removidas por todos os indivíduos desse grupo
R9 Um administrador é um indivíduo que pode espionar as mensagens do sistema e obter outras informações secretas
R10 Existe um administrador nato com nome="admin"/senha="admin"

Classe Fachada

public static void	criarIndividuo(nomeindivíduo, senha) – cria um indivíduo no repositório, caso inexista no repositório
public static Individual	validarIndividuo(nomeindivíduo,senha) – retorna o indivíduo, se existir no repositório

public static void	criarAdministrador(nomeadministrador, senha) – cria um administrador no repositório, caso inexista no repositório
public static void	criarGrupo(nomegrupo) – cria um grupo no repositório, caso inexista no repositório
public static void	inserirGrupo(nomeindividuo, nomegrupo) – localiza o indivíduo e o grupo no repositório e cria o relacionamento entre eles
public static void	removerGrupo(nomeindividuo, nomegrupo) – localiza o indivíduo e o grupo no repositório e remove o relacionamento entre eles
public static void	criarMensagem(nomeindivíduo, nomedestinatario, texto) – localiza o indivíduo e o participante destinatário no repositório, cria a mensagem no repositório e a relaciona com eles dois. Se o destinatário for um grupo, envia cópias da mensagem (com mesmo id) do grupo para os membros desse grupo (exceto para emitente que criou a mensagem original), concatenando ao texto o nome do emitente, na forma "nome/texto". Adicionar as copias no repositório
public static ArrayList<Mensagem>	obterConversa(nomeindivíduo, nomedestinatario) – localiza o indivíduo e o participante destinatário no repositório e retorna todas as mensagens enviadas do indivíduo ao destinatário e enviadas do destinatário ao indivíduo, em ordem cronológica
public static void	apagarMensagem(nomeindivíduo, id) – localiza no repositório o indivíduo e a mensagem emitida por ele, remove os relacionamentos entre a mensagem e o emitente e destinatário e remove a mensagem do repositório. Se a mensagem foi enviada a um grupo, suas copias devem ser também removidas do repositório bem como seus relacionamentos
public static ArrayList<Mensagem>	listarMensagensEnviadas(nomeindivíduo) - localiza no repositório o indivíduo e retorna as mensagens enviadas por ele
public static ArrayList<Mensagem>	listarMensagensRecebidas(nomeparticipante) - localiza no repositório o participante e retorna as mensagens recebidas por ele
public static ArrayList<Individual>	listarIndividuos() - retorna os indivíduos do repositório
public static ArrayList<Grupo>	listarGrupos() - retorna os grupos do repositório
public static ArrayList<Mensagem>	espionarMensagens(nomeadministrador, termo) - localiza no repositório o administrador e retorna as mensagens do sistema, que contem o termo fornecido (termo vazio retorna todas as mensagens do sistema)
public static ArrayList <String>	ausentes(nomeadministrador) – localiza no repositório o administrador e retorna os nomes dos participantes que não criaram mensagens

Obs: Estes métodos devem lançar exceção de acordo com as regras de negócio.

Considerações finais:

- Não se pode alterar as assinaturas dos métodos da Fachada
- Serão fornecidas classes de teste e classes swing (IU).
- A chave do mapa *participantes* é o nome do participante
- ~~A chave do mapa *mensagens* é o id da mensagem~~
- O método carregarDados() lê de arquivo texto todos os objetos do repositório e o método salvarDados() grava em arquivo texto todos os objetos do repositório, usando os arquivos:
  - Os indivíduos são armazenados no arquivo **individual.csv** (nomeindividuo;senha;administrador)
  - Os grupos são armazenados no arquivo **grupo.csv** (nomegrupo;p1;p2;...;pN) onde p1,p2,...,pN são os nomes dos indivíduos de cada grupo.
  - As mensagens são armazenadas no arquivo **mensagem.csv** (id;texto,nomeemitente;nomedestinatário;datahora), onde datahora está no formato "dd/MM/yyyy HH:mm:ss"

Pontuação:

1. Implementar as classes do modelo - 30pts
2. Implementar a classe Repositorio - 10pts
3. Implementar a classe Fachada - 30pts
4. Apresentação obrigatória (avaliação individual) – 30pts

Grupo de máximo de 3 alunos