Amigo Oculto 8 - Participantes

Gerenciando os participantes

Cada participante pode estar em N grupos e cada grupo pode conter N participantes. Assim, temos um relacionamento chamado de N:N (N para N). A criação de um relacionamento N:N raramente é direta. A forma mais comum é criarmos uma entidade intermediária que conecta uma entidade à outra, isto é, usuários a grupos. Chamaremos essa entidade de participação. Por meio dela, transformamos o relacionamento N:N em dois relacionamentos 1:N que já sabemos implementar: o primeiro é um relacionamento 1:N entre usuários e participações (1 usuário pode ter N participações); o segundo é um relacionamento 1:N entre grupos e participações (1 grupo pode ter N participações); uma participação, porém, pertence a um único usuário e a um único grupo.

PARTICIPANTES

A entidade Participação

A entidade Participação precisa ter os seguintes atributos:

- `int idParticipação` - Este atributo identificará cada participação, isto é, o vínculo de um usuário a um grupo, de forma exclusiva.
- `int idUsuário` - Este atributo conterá o ID do usuário que participará do grupo.

- `int idGrupo` - Este atributo conterá o ID do grupo do qual o usuário participará.
- `int idAmigo` - Manteremos, neste atributo, a informação de quem será presenteado por este usuário. Este atributo é, portanto, o ID de outro usuário. Esse valor só será preenchido quando fizermos o sorteio.

Novamente, você precisará implementar os métodos da interface `Registro`, incluindo os métodos `getID()`, `setID()`, `chaveSecundaria()`, `toByteArray()` e `fromByteArray()`. Nesta entidade, o seu método de chave secundária será usado para, mais tarde, recuperarmos a participação de um usuário em um grupo. Assim, você pode retonar a concatenação do ID do usuário com o ID do grupo.

```
public String chaveSecundaria() {
  return this.idUsuario+"|"+this.idGrupo;
}
```

DESAFIO 4: ~~Altere a sua classe do CRUD genérico, para que ela não crie o índice indireto, quando a entidade retornar *null* à chamada do método `chaveSecundaria()`~~.
Obs.: *Esse desafio perdeu o sentido com a revisão do método chaveSecundária(), pois ele será necessário para recuperar a participação de um usuário específico em um grupo específico.*

Menu de participantes

O menu de gerenciamento de participantes será acessado por meio do menu de gerenciamento de grupos. Aqui estamos falando de operações que serão realizadas pelo administrador do grupo e não pelo próprio participante. As operações disponíveis ao administrador do grupo, portanto, serão:

```
AMIGO OCULTO 1.0
================

INÍCIO > GRUPOS > GERENCIAMENTO DE GRUPOS > PARTICIPANTES

1) Listagem
2) Remoção

0) Retornar ao menu principal

Opção: _
```

~~Humm..... um menu para apenas uma opção? Sim! Por enquanto, nossa aplicação é muito simples. Futuramente, você poderá ampliar as operações de gerenciamento de participantes.~~

ATUALIZAÇÃO 28/04: Resolvi acrescentar uma listagem de participantes, mas isso fica como atividade opcional para vocês. No entanto, o código é bastante simples, pois é apenas uma

parte do código de remoção (sem a remoção propriamente dita).

Bem, remover um participante é algo diferente de cancelar um convite. Estamos falando de alguém que já aceitou o convite e aqui, precisamos considerar a remoção desse participante em dois possíveis cenários: grupos ainda não sorteados e grupos já sorteados. Se o grupo ainda não tiver sido sorteado, basta remover a participação Se o grupo já tiver sido sorteado, precisaremos identificar quem saiu com esse usuário, para que possamos repassar o amigo do usuário a ser removido para quem saiu com ele.

Se ficou meio confuso, pense dessa forma: Suponha que precisemos retirar o usuário João do grupo. João deveria presentear Alice. E quem daria presente para o João seria a Maria. Assim, precisamos alterar os registros de participações, para que Maria presenteie Alice e João pode sair da história.

DESAFIO 5: Que tal impedir que o administrador retire participantes arbitrariamente. Você poderia implementar uma restrição por meio da qual o participantes só possa ser retirado de um grupo quando ele solicitar. Assim, você precisaria criar um CRUD de solicitação de saída do grupo.

CRUD

Antes de prosseguirmos, é importante que você crie um CRUD de participações (que já foi necessário na página anterior, para a aceitação de convites). Paralelamente, precisaremos de duas árvores B+ desta vez. Uma será usada para armazenar os pares de ID de grupo e ID de participação. A outra será usada para armazenar os pares de ID de usuário e ID de participação. Dessa forma, poderemos buscar as participações tanto por meio dos grupos (quais usuários participam de um grupo) quanto por meio dos usuários (quais grupos um usuário participa).

Inclusão de participantes

Não faremos a inclusão dos participantes de forma direta, isto é, não haverá uma opção no menu para inclusão de novos participantes. Essa inclusão acontece no momento em que o usuário aceita o convite. Quando ele fizer isso, retiraremos o seu convite da lista de convites pendentes e criaremos uma nova participação desse usuário no grupo para o qual ele foi convidado. Isso foi encaixado já no algoritmo de Inscrições da página anterior (que você precisará alterar).

Listagem dos participantes

O código para a listagem de participantes é bastante curto. Basicamente, precisamos escolher o grupo e, em seguida, recuperar as participações vinculadas a esse grupo. Isso pode ser feito por meio desses passos:

1. Obter a lista de IDs de grupos na Árvore B+ dos grupos usando o ID do usuário;
2. Para cada ID nessa lista,
    1. Obter os dados do grupo usando o método *read(ID)* do CRUD de grupos;
    2. Se o grupo estiver ativo, apresentar os dados do grupo na tela.
3. Solicitar do usuário o número do grupo que deseja gerenciar;
    1. Se o usuário digitar 0, retornar ao menu de grupos;
4. Usando o ID do grupo escolhido, recuperar os dados do grupo usando o método *read(ID)* do CRUD;
5. Apresentar os dados do grupo na tela, explicitando se o sorteio já foi realizado ou não;
6. Obter a lista de IDs de participações em uma das Árvore B+ das participações, usando o ID do grupo escolhido;
7. Para cada ID nessa lista,
    1. Obter os dados da participação, usando o método *read(ID)* do CRUD de participações;
    2. Usando o ID do usuário na participação, obter os dados do usuário usando o método *read(ID)* do CRUD de usuários;
    3. Apresentar o nome do usuário.

Observe que essa sequência de passos é exatamente a sequência inicial de passos do método de remoção, apresentado abaixo.

Remoção de participantes

A primeira operação da remoção de participantes é saber se o sorteio para o grupo já foi realizado. Precisaremos, antes disso, identificar de qual grupo retiraremos o participante. Faremos isso por meio dos seguintes passos (da mesma forma que fizemos nas etapas anteriores):

1. Obter a lista de IDs de grupos na Árvore B+ dos grupos usando o ID do usuário;
2. Para cada ID nessa lista,
    1. Obter os dados do grupo usando o método *read(ID)* do CRUD de grupos;
    2. Se o grupo estiver ativo, apresentar os dados do grupo na tela.
3. Solicitar do usuário o número do grupo que deseja gerenciar;
    1. Se o usuário digitar 0, retornar ao menu de grupos;
4. Usando o ID do grupo escolhido, recuperar os dados do grupo usando o método *read(ID)* do CRUD;
5. Apresentar os dados do grupo na tela, explicitando se o sorteio já foi realizado ou não;
6. Obter a lista de IDs de participações em uma das Árvore B+ das participações, usando o ID do grupo escolhido;
7. Para cada ID nessa lista,
    1. Obter os dados da participação, usando o método *read(ID)* do CRUD de participações;
    2. Usando o ID do usuário na participação, obter os dados do usuário usando o

método *read(ID)* do CRUD de usuários;
3. Apresentar o nome do usuário;
4. Se o sorteio já tiver sido realizado,
    1. Armazene (em memória principal) o par de ID do usuário a ser presenteado e ID da participação do usuário presenteador em uma estrutura de qualquer tipo (ex.: tabela *hash*), que você pode chamar de `presenteadosPor`;
8. Solicitar o número do usuário (isto é, da participação) que será removido do grupo;
    1. Se o usuário digitar 0, retornar ao menu de gerenciamento de participações;

O próximo conjunto de passos transfere o usuário a ser presenteado por este ao outro usuário:

9. Se o sorteio **já** tiver sido realizado,
    1. Identificar o usuário que seria presenteado por este usuário a ser removido (a partir do objeto da participação);
    2. Identificar na estrutura `presenteadosPor` qual é o ID da participação da pessoa que presentearia o usuário a ser removido;
    3. Usando o método *update()* do CRUD de participações, passar a essa participação (do passo anterior), o `idAmigo` daquele que seria presenteado pelo usuário a ser removido;

E, agora, a remoção da participação, independentemente de o sorteio já ter sido realizado ou não:

10. Excluir a participação usando o método *delete()* do CRUD de participações, usando o seu ID de participação;
11. Remover o par ID do grupo e ID da participação da primeira árvore B+ de participações;
12. Remover o par ID do usuário e ID da participação da segunda árvore B+ de participações;
13. Retornar ao menu de gerenciamento de participantes.

Não foi tão difícil assim, certo? Tivemos que acessar muitos CRUDs, mas as operações eram relativamente simples.