UFPI/CCN/DIE – Programação Orientada a Objetos – 2023/2
Aluno: _____ Matr.:_____

1ª Avaliação

Um grupo de pessoas se organiza em uma associação para realização de trabalhos sociais. Essas associações possuem encontros regulares, em que é necessário controlar a frequência de seus associados, além de cobrar taxas mensais para custear tanto o funcionamento da associação, bem como de suas ações solidárias.

As taxas são criadas no início do ano e possuem um nome, um valor anual, a quantidade de parcelas, sua vigência (ano) e a informação se ela é administrativa ou social. O valor mensal de uma taxa é dado pelo valor anual dividido pela quantidade de parcelas. Se uma taxa for cadastrada tendo apenas 6 parcelas, com valor anual de R$120, então sua taxa mensal é de R$20,00. Não deve ser possível pagar um valor menor que uma parcela. A não ser que seja o valor final para fechamento da taxa anual.

O pagamento das taxas pode ter como valor mínimo uma parcela, mas pode aceitar valores maiores, ficando o resto para ser abatido de futuras mensalidades. É possível criar várias taxas durante um ano (vigência), até com o mesmo nome de outras taxas, porém, de outras vigências. Assim, a tentativa de criar uma taxa com um mesmo nome de uma taxa já cadastrada para uma mesma vigência deve gerar exceção. Mas se for para vigência diferente, em que não há uma taxa com esse nome, deve ser permitido.

Um associado tem como atributos *numero* positivo (int), *nome* (String), *telefone* (String), *nascimento* (long), *dataAssociacao* (long). Existem associados remidos, que são aqueles que não pagam mais taxas **administrativas**. Um dado adicional para tais associados é a sua data de remissão (long), que é a data em que ele passou a não precisar mais pagar taxas administrativas. Se um associado foi remido em outubro de um ano, ele deve ter pago todas as taxas até um mês antes de outubro, que nesse caso é setembro.

Uma associação possui basicamente um número e nome. Além disso, tem uma relação de associados, além de uma relação de reuniões realizadas, cada reunião com indicação dos participantes da reunião, que devem ser associados da associação na data da reunião.

Nenhum dado pode ser salvo sem os seus dados preenchidos. Se isso acontecer, deve-se gerar uma exceção ValorInvalido, indicando que algum dado de alguma entidade não foi completamente preenchido, tais como dados de um associado, dados de reunião, dados de taxa, etc. São considerados não preenchidos strings vazias ou nulas, e no caso de números, valor menores que zero. Tudo isso deveria gerar a exceção de ValorInvalido.

Durante uma reunião é registrada a frequência dos participantes, bem como é construída uma ata, que resume tudo o que foi discutido durante a reunião. A ata é apenas um texto relatando os acontecimentos naquela data.

Você deve criar um sistema para controle de associações. Deve ser possível criar diferentes associações. Esse sistema terá como classe principal a classe **MinhaAssociacao** (exatamente com esse nome!) que deve implementar a InferfaceAssociacao descrita aqui. Ela deve ser implementada em um pacote chamado “associacao” dentro do pacote com o seu nome, tal qual mostramos nas aulas.

Nesta prova teremos 2 atividades. A atividade 1 será a implementação de 2 dos métodos descritos abaixo, sendo um dos métodos do grupo de métodos **adicionar** e o outro sendo de algum dos métodos restantes. Sua implementação deve ser clara o suficiente para permitir que seja avaliado sua correção. Se você criar ou usar um método que realiza uma série de ações e que seja importante para a correção, ele deve ser detalhado.

Assinatura dos métodos construtores. Ressaltamos que podem existir mais atributos e mais classes na sua implementação, mas devem existir pelo menos esses aqui detalhados!

```
public Associacao(int num, String nome);
public Associado (int numero, String nome, String telefone,   long dataAssociacao,
long nascimento);
public AssociadoRemido(int numero, String nome, String telefone, Date
dataAssociacao, Date nascimento, Date dataRemissao);
public Taxa(String nome, int vigencia, double valorAno, int parcelas, boolean
administrativa); //Se true, administrativa. Se false, outras.
public Reuniao(Date data, String ata);

package associacao;
public interface InterfaceAssociacao {
// Calcula a frequência de um associado nas reuniões ocorridas durante um
determinado período, retornando um número entre 0 e 1 (ex: 0,6, indicando que o
associado participou de 60% das reuniões.
public double calcularFrequencia(int numAssociado, int numAssociacao, Date inicio,
Date fim) throws AssociadoNaoExistente, ReuniaoNaoExistente,
AssociacaoNaoExistente, ValorInvalido;
// Registra a frequencia de um associado em uma reunião. Não deveria registrar
participacao em reuniões acontecidas antes da sua filiação na associação.
public void registrarFrequencia(int codigoAssociado, int numAssociacao, Date
dataReuniao)throws AssociadoNaoExistente, ReuniaoNaoExistente,
AssociacaoNaoExistente, FrequenciaJaRegistrada, FrequenciaIncompativel,
ValorInvalido;
// Registra o pagamento de uma taxa, em uma associação, dentro uma determinada
competência, para um associado. O valor a ser pago não pode ser menor que uma
parcela, embora não precise ser exatamente duas parcelas. Uma parcela de R$20,00
por mês aceita um pagamento de R$30,00, sendo uma parcela completa e um pedaço da
próxima. Associados remidos não deveriam mais realizar pagamentos de taxas
administrativas vigentes em datas antes da sua remissão, gerando exceção de
AssociadoJaRemido se houver tentativa de se pagar algo para esse caso. Caso o valor
a ser pago seja menor que o mínimo (não sendo o ultimo do ano!) ou gerando
pagamento maior que a taxa anual, gerar exceção de ValorInvalido. Lembrar de
verificar valores negativos.
public void registrarPagamento(int numAssociacao, String taxa, int vigencia, int
numAssociado, Date data, double valor) throws AssociacaoNaoExistente,
AssociadoNaoExistente, AssociadoJaRemido, TaxaNaoExistente, ValorInvalido;
// Calcula o total de pagamentos realizado por um associado, em uma associação,
para uma taxa, que possui uma vigência, dentro de um certo período de tempo.
public double somarPagamentoDeAssociado (int numAssociacao, int numAssociado,
String nomeTaxa, int vigencia, Date inicio, Date fim) throws
AssociacaoNaoExistente, AssociadoNaoExistente, TaxaNaoExistente;
// Calcula o total de taxas previstas para um dado ano, em uma associação.
public double calcularTotalDeTaxas (int numAssociacao, int vigencia) throws
AssociacaoNaoExistente, TaxaNaoExistente, ValorInvalido;
// Adiciona uma associação a ser gerenciada. Valida todos os campos para evitar
dados não preenchidos.
public void adicionar(Associacao a) throws AssociacaoJaExistente, ValorInvalido;
// Adiciona um associado a uma associação. Valida todos os campos para evitar dados
não preenchidos.
public void adicionar(int associacao, Associado a) throws AssociacaoNaoExistente,
AssociadoJaExistente, ValorInvalido;

// Adiciona uma reunião a uma associação. Valida todos os campos para evitar dados
não preenchidos.
public void adicionar(int associacao, Reuniao r) throws AssociacaoNaoExistente,
ReuniaoJaExistente, ValorInvalido;
// Adiciona uma taxa a uma associação. Valida todos os campos para evitar dados não
preenchidos.
public void adicionar(int associacao, Taxa t) throws AssociacaoNaoExistente,
TaxaJaExistente, ValorInvalido;

}
```