

MC322 - Laboratório 6

Prof. Esther Colombini

esther@ic.unicamp.br

<http://www.ic.unicamp.br/~esther/teaching/2020s2/mc322>

PEDs:

Pedro Santos de Rezende Alves (pedrorezendesantos@gmail.com)

Renata Falguera Gonçalves (renatafalguera@gmail.com)

PADs:

Bernardo do Amaral Teodosio (b167494@dac.unicamp.br)

Fabrício de Souza Maruta (f138313@dac.unicamp.br)

Gabriel de Freitas Garcia (g216179@dac.unicamp.br)

1 Descrição Geral

Com o avanço da pandemia de SARS-COV-2 e a necessidade mundial de isolamento, atividades acadêmicas em todo o mundo migraram para um modelo temporário integralmente virtual, o que culminou em uma mudança de paradigma e necessidade de reorganização do modelo de estudo por parte dos estudantes. Neste cenários, ferramentas capazes de permitir a organização das atividades (Trello), a comunicação entre os alunos divididos por grupos de interesse (Discord, Slack) e a troca de material de apoio, se tornaram importantes aliados no processo de aprendizagem. Neste contexto, os laboratórios desenvolvidos ao longo do semestre terão por objetivo construir uma ferramenta colaboartiva de organização das atividades acadêmicas dos alunos com vistas à comunicação, troca de conteúdos e alinhamento de atividades do semestre letivo.

2 Objetivo

O objetivo desta atividade consiste na familiarização com o conceito de *Interfaces* e *Classes Abstratas* através de implementações e alterações nos conjuntos das classes desenvolvidas em laboratórios anteriores.

3 Atividade de Interfaces

3.1 Definições

Neste laboratório serão propostas alterações nas classes Grupo, Usuario e Cartao, além da criação de novas classes / interfaces, caso necessário. A classe Cartao passará a ter uma função mais ativa e representará uma tarefa ou "lembrete" de uma tarefa que precisará ser executada por usuários participantes de um grupo. A classe Usuario possuirá uma nova subclasse, **UsuarioComum**, que representará um usuário *default* do sistema. Novos métodos e atributos, assim como a modificação de métodos já implementados nos laboratórios anteriores, serão desenvolvidos com intuito de explorar o conceito e a utilização de *Interfaces* e *classes abstratas* em um sistema. O diagrama 1 apresenta as classes do sistema:

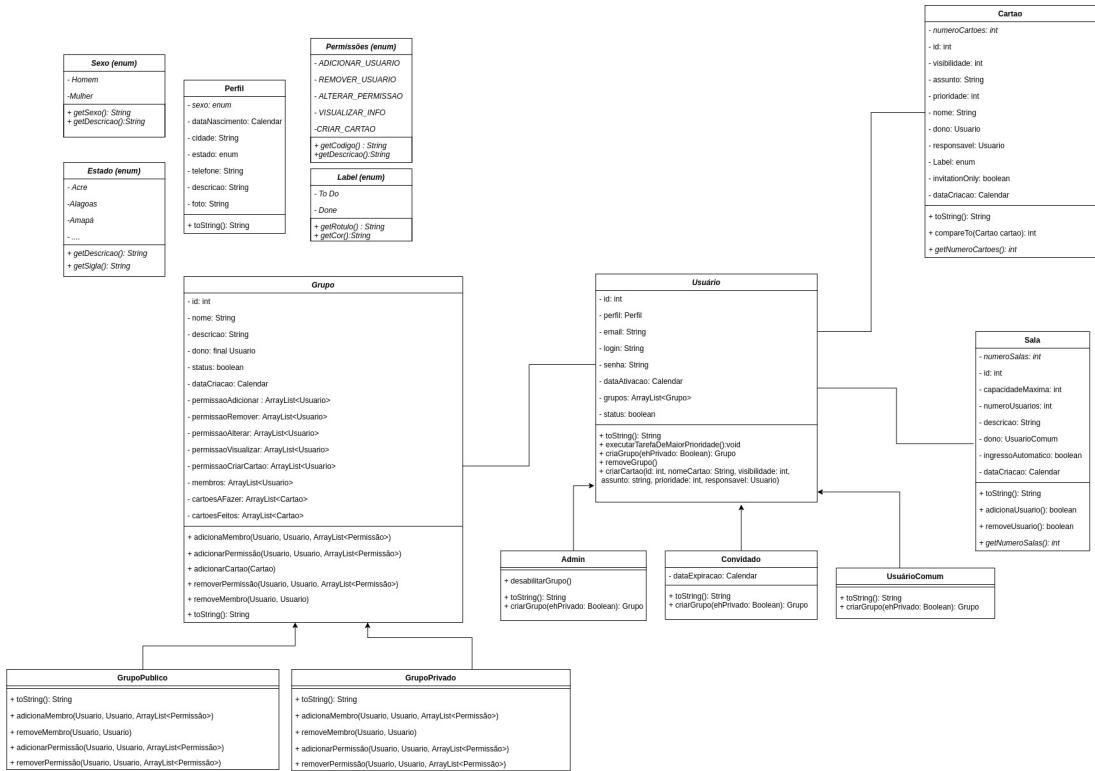


Figura 1: Diagrama UML da classe

3.1.1 Classe Grupo: Interface ou Classe Abstrata?

A primeira implementação solicitada neste laboratório é, também, um desafio de conceito. Baseado nos conceitos de interface e classe abstrata e na arquitetura hierárquica da classe **Grupo** e suas subclasses **GrupoPublico** e **GrupoPrivado**, a classe Grupo deverá:

- (a) Ser alterada e implementada como uma *Interface*?
- (b) Ser alterada e implementada como uma *Classe Abstrata*?
- (c) Ser mantida como está atualmente?

De acordo com a resposta à essa questão, a classe-mãe **Grupo**, bem como seus atributos e métodos, deverá ser reimplementada para garantir as características da abordagem escolhida e manter a estrutura polimórfica do sistema.

QUESTÃO: Qual foi a estratégia abordada: *Interface*, *Classe Abstrata* ou *manutenção da classe concreta*? Explique sua escolha.

3.1.2 Cartão

1. **Interface Comparable:** Neste laboratório, a classe Cartão deverá implementar a interface *Comparable*, padrão da linguagem Java. Para isso, o método *compareTo* desta interface deverá ser implementado na classe Cartão. O objetivo da implementação deste método é fazer com que dois cartões distintos possam ser comparados a partir de sua prioridade, permitindo a ordenação dos cartões quando necessário. O método *compareTo* deve retornar um inteiro, que indicará se o cartão *this* tem prioridade maior, menor ou igual ao cartão recebido por parâmetro no método.
2. **Cartões com responsáveis:** Um atributo *responsável*, do tipo *Usuario*, deverá ser incluído à classe Cartão. Este atributo indicará quem é o usuário responsável por tratar / resolver / executar aquele cartão.

3. **Prioridade de execução:** Um atributo *prioridade*, do tipo *int*, deverá ser incluído à classe *Cartao*. Esse atributo deverá representar o grau de urgência de execução do cartão e deve ser indicado por valores inteiros de 1 a 5, com **1 = alta prioridade** e **5 = baixa prioridade**.

3.1.3 Alterações no método *criarCartao*

O método *criarCartao()*, da classe **Usuario**, deverá ser alterado de modo a receber os parâmetros necessários para a criação de um cartão, além do id do grupo estabelecido em implementações anteriores. Os novos atributos deverão ser todos aqueles definidos no construtor da classe *Cartao* - nome, visibilidade, assunto, e etc.

Obs.: Criação de Cartões - Note que, com adição do atributo *responsavel* à classe *Cartao*, todo cartão deverá ter necessariamente um responsável. Ao criar um novo cartão, você deve se certificar que o usuário responsável por este cartão também faz parte do grupo ao qual o cartão pertence. Além disso, cartões recém-criados devem ter **TODO** como label.

3.1.4 Classe **Usuario** e representação de um **UsuarioComum**

Iremos trabalhar com a utilização de classes abstratas através de alterações na classe **Usuario** e suas subclasses. A primeira alteração será a mudança da classe **Usuario** transformando-a de uma **classe concreta** em uma *Classe Abstrata*, ou seja, a partir de agora não será possível instanciar objetos do tipo **Usuario**.

Crie a subclasse **UsuarioComum**. Esta nova subclasse deverá representar o que em laboratórios anteriores a **classe concreta Usuario** representava, ou seja, um usuário que não é admin ou convidado. Deste modo haverão 3 tipos de usários: Usuário Comum, Admin e Usuário Convidado. As 3 subclasses devem herdar e implementar os métodos abstratos da classe **Usuario** e sobrescrever os métodos concretos necessários.

A estrutura de implementação pode ser vista no diagrama 1.

Após a implementação destas classes, responda à questão:

QUESTÃO: Em outra possível abordagem, a classe **Usuario** poderia ser implementada como uma *Interface*? Caso sim, por quê? Quais alterações seriam necessárias?

3.1.5 Usuário com Perfil

A classe **Usuario** deverá ser alterada e um atributo *perfil*, do tipo **Perfil**, deverá ser incluído. Este atributo indica que todo usuário do sistema deverá ter um perfil.

3.1.6 Controle de estado dos cartões

A partir deste laboratório, os cartões de um grupo serão classificados em dois estados: **TODO** e **DONE**. Esses estados representarão, respectivamente: tarefas de um grupo que foram listadas mas ainda não foram executadas (**TODO**); e tarefas já executadas (**DONE**). A classe **Grupo** deverá ser alterada de forma a manter em listas distintas os cartões a serem feitos (**TODO**) e os cartões já executados (**DONE**). Para isso, remova o atributo *cartoes* desta classe, e adicione duas novas listas de cartões: *ArrayList<Cartao>cartoesAFazer* e *ArrayList<Cartao>cartoesFeitos*.

3.1.7 Execução de cartões

Um cartão (ou a tarefa que ele representa) pode ser executado pelo seu responsável. Executar um cartão significa, neste momento, alterar o estado do cartão de **TODO** para **DONE** em todas as instâncias em que se fizer necessário - o que inclui mudar o cartão de uma lista para outra na classe **Grupo**, alterar sua label e etc.

Porém, a execução de uma tarefa deverá obedecer alguns critérios, uma vez que deve-se dar preferência de execução para cartões que foram definidos com maior prioridade. Você deverá criar um método na classe **Usuario** chamado **executarTarefaDeMaiorPrioridade()**. Este método, dado o usuário que o chamou, deverá buscar o cartão de maior prioridade e executá-lo (considerando

a definição execução de um cartão). Observe que, para implementar este método, você deve percorrer todos os grupos da lista de grupos do usuário em busca de cartões pelos quais ele seja responsável e, destes cartões, executar aquele de maior prioridade.

- **Dica 1:** Utilize o método *compareTo* para encontrar o cartão de maior prioridade.
- **Dica 2:** Caso um usuário possua dois ou mais cartões com mesma prioridade, estabeleça um critério de desempate para definir qual cartão deverá ser executado.

3.2 Tarefa

Como tarefa:

1. Modifique classes, métodos antigos e acrescente os novos métodos e atributos propostos para as classes Grupo, Usuario, Cartao conforme apresentado no diagrama UML 1 e na descrição do laboratório.
2. Implemente a classe **UsuarioComum** conforme apresentado no diagrama UML 1.
3. Crie N grupos, privados e públicos. Para cada grupo criado:
 - Adicione ao menos 3 usuários ao grupo, um usuário de cada tipo.
 - Utilize um usuário do grupo para criar 6 cartões com diferentes níveis de prioridade e atribua como o responsável algum outro usuário do grupo.
 - Imprima os cartões criados.
 - Execute todos os cartões criados de acordo com suas prioridades.

3.3 Observações

- Não peça, em nenhum momento, dados pela entrada padrão. Construa os objetos com valores inseridos diretamente no código.
- O manejo, bem como o processo de ordenação da listas de cartões (manter listas ordenadas ou não), é um processo de sua escolha.
- Desenvolva seu código priorizando, sempre que possível, a manutenção e uso de polimorfismo.
- Onde se fazer necessário o uso de classes abstratas, a definição de métodos abstratos e concretos deverá ser feita pelo aluno.

4 Questões

Sobre a atividade realizada, responda como comentário no início do código da classe que contém o main as questões levantandas em 3.1.1 e 3.1.4.

5 Submissão

Para submeter a atividade utilize o Google Classroom. Salve os arquivos dessa atividade em um arquivo comprimido no formato .tar.gz ou .zip e nomeie-o **Lab6-000000.tar.gz** ou **Lab6-000000.zip** trocando '000000' pelo seu número de RA. Submeta o arquivo na seção correspondente para esse laboratório no Classroom da disciplina MC322.

Datas de entrega

- Dia **04/01/2021** até às 21:00h