

## MC322 - Laboratório 3

**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 do aluno com a construção de classes que possuam atributos *final*, além do uso de *ArrayList* e Herança.

## 3 Atividade

### 3.1 Definições

Adicionaremos as classe referente aos Grupos e o Admin nesta atividade. As classes devem conter os atributos e métodos descritos do diagrama da figura 1.

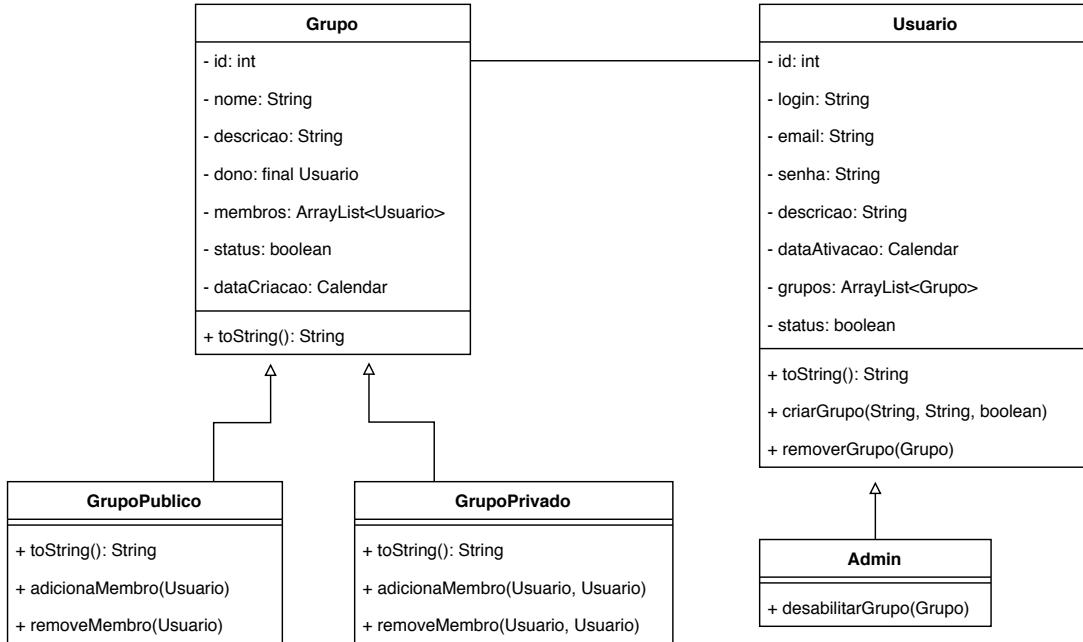


Figura 1: Diagrama UML da classe

Nesta parte do laboratório, adicionaremos 4 novas classes ao projeto: **Grupo**, **GrupoPublico**, **GrupoPrivado** e **Admin**. Por meio dessas classes vamos demonstrar o funcionamento de herança em Orientação a Objetos.

- Grupo é a classe mãe das quais as demais classes herdarão.
- GrupoPublico e GrupoPrivado são classes filhas e portanto são mais especializadas. Estas são classes de um tipo de Grupo. Cada uma possui objetivos distintos já que GrupoPublico será utilizada para um grupo visível a todos os usuários e GrupoPrivado representará um grupo visível apenas para quem for convidado ou autorizado a participar do mesmo.
- Admin é uma especialização de um Usuário que pode desabilitar grupos.

Como tarefa:

1. Implemente a nova classes conforme apresentado no diagrama UML 1. Os *getters*, *setters* e construtores foram omitidos mas eles precisam ser criados para cada atributo, a não ser quando estabelecido explicitamente. Não crie *setters* para o *ArrayList de membros*, ao invés disso instancie um *ArrayList* vazio em todos os construtores que achar necessário criar.
2. Implemente o método *adicionaMembro(Usuario)* em GrupoPublico. Este método deve adicionar o objeto *usuario* passado por parâmetro na lista *membros*. Só podem ser adicionados membros se o grupo estiver com status de ativo. Neste momento, qualquer usuário pode adicionar membros a um grupo público.
3. Implemente o método *removeMembro(Usuario)* em GrupoPublico. Este método deve remover o objeto *usuario* passado por parâmetro na lista *membros*. Só podem ser removidos membros se o grupo estiver com status de ativo. Neste momento, qualquer usuário pode remover membros de um grupo público.
4. Implemente o método *adicionaMembro(Usuario)* em GrupoPrivado. Este método deve adicionar o objeto *usuario* passado por parâmetro na lista *membros*. Só podem ser adicionados membros se o grupo estiver com status de ativo. Apenas o dono do grupo pode adicionar membros a um grupo privado.
5. Implemente o método *removeMembro(Usuario)* em GrupoPublico. Este método deve remover o objeto *usuario* passado por parâmetro na lista *membros*. Só podem ser removidos membros se o grupo estiver com status de ativo. Apenas o dono do grupo pode remover membros a um grupo privado.
6. Sobrescreva o método *toString()* das classes *GrupoPrivado* e *GrupoPublico* de forma que imprima todas as informações, incluindo a informação de cada membro do grupo.

7. O atributo dono da classe Grupo deve ser *final*. Isso significa que, uma vez definido, o dono do grupo não poderá ser alterado.
8. No método *main*, crie 4 objetos da classe *Usuario*, 1 objeto da classe *Admin* e um objeto de um *GrupoPublico* e outro do *GrupoPrivado*. Defina os donos dos grupos e adicione ao menos 2 membros a cada um dos grupos. Ao longo do código de teste (no main) invoque os métodos para os objetos e demonstre que:
  - O dono de um grupo não pode ser modificado
  - Apenas um admin pode mudar o status de um grupo
  - Apenas o dono de um grupo privado pode inserir ou remover usuários
9. Ao longo dos testes apresente os dados dos grupos criados, inclusive a lista atual de membros.

s

### 3.2 Observações

- Como o atributo *dono* da classe *Grupo* é *final*, é necessário que o construtor da classe receba pelo menos esse atributo. Além disso, deixa de fazer sentido a criação de um *setter* para esse atributo (nem é possível);
- Como as operações de *adicionarMembro* e *removerMembro* em *GrupoPrivado* só podem ser executadas pelo dono do grupo, estes métodos precisam receber, além do usuário a ser removido, quem foi o usuário que chamou o método.
- Não peça, em nenhum momento, dados pela entrada padrão. Construa os objetos com valores inseridos diretamente no código.

## 4 Questões

Sobre a atividade realizada, responda como comentário no início do código da classe que contém o main.

- Tente modificar o valor da dono de um grupo (que é um atributo final). Crie um *setter* se necessário. Foi possível fazer a modificação? Explique.
- Agora, no método main, crie uma variável *final* do tipo *Grupo*, e instancie ela com os valores que preferir. Tente modificar algum atributo do objeto através de um *setter*, como o atributo referente ao id. Foi possível modificar esse atributo, mesmo com o objeto sendo *final*? Por quê?
- Se ao invés de usar *ArrayList* para definir a lista de membros da classe *Grupo* tivéssemos usado um array, o que mudaria na implementação? Poderíamos continuar adicionado membros como fizemos? Haveria alguma limitação? Discuta as desvantagens dessa solução.
- Qual o principal benefício da herança?
- Adicione final na classe *Grupo*. O que aconteceu com o código? Por que isso aconteceu? Em vez de *Grupo* ser final e se definirmos *GrupoPublico* como final?
- Por que definimos os métodos *adicionaMembro* e *removeMembro* nas classes filhas e não na classe mãe (*Grupo*)?

## 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 **Lab3-000000.tar.gz** ou **Lab3-000000.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 **9/11/2020** até às 21:00h