Engenharia de Telecomunicações
POO29004 – Programação Orientada a Objetos

27/03/2020

Lista 5: Modelagem de classes com associação

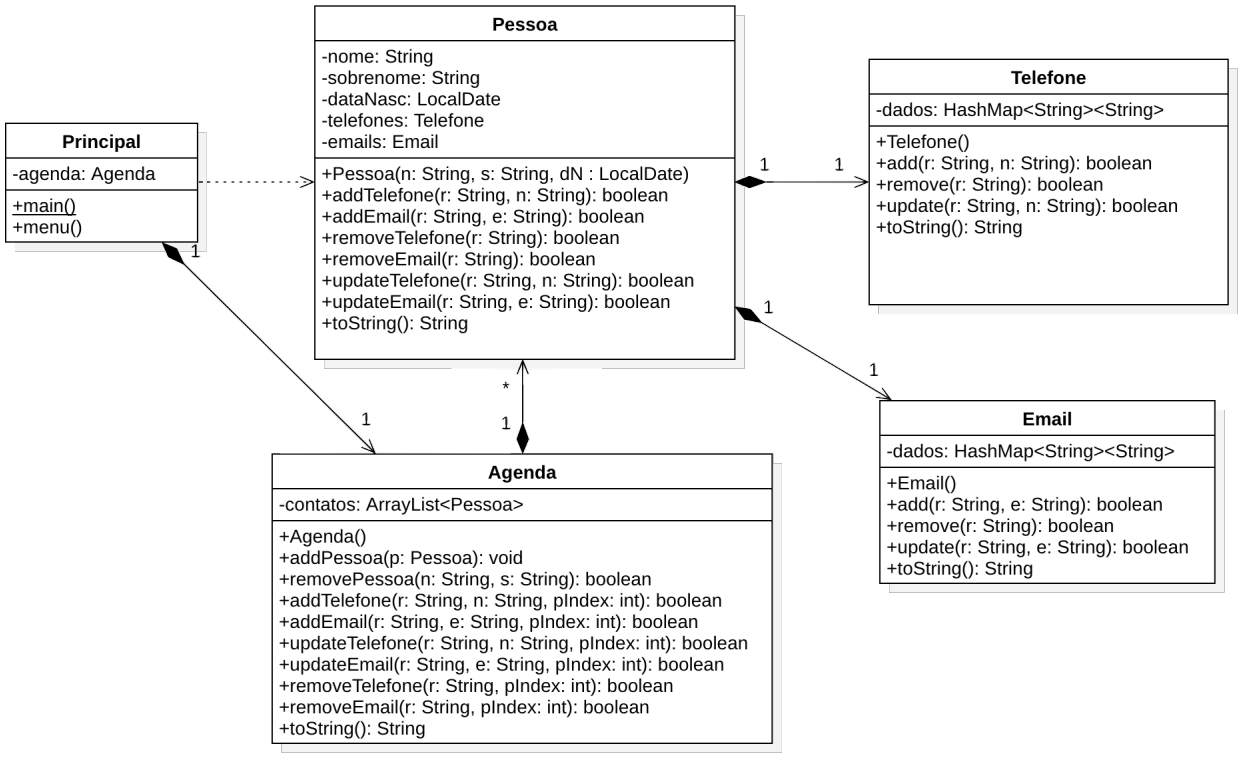
Professor: Emerson Ribeiro de Mello

http://docente.ifsc.edu.br/mello/poo

Nota:

- *Antes de iniciar, sincronize seu repositório local com o seu repositório remoto "Listas de exercícios" do Github. É possível que o professor tenha feito um commit por lá após a entrega da lista de exercício anterior;*
- *Crie um diretório com o nome* `lista-05` *na raiz do repositório e dentro deste, crie um projeto Java+gradle com o IntelliJ para cada um dos exercícios abaixo;*
- *Edite o arquivo* `Readme.md` *que está na raiz do repositório e adicione um novo item com o nome* `lista-05` *e este deverá ter um link o qual deverá apontar para o subdiretório com o nome* `lista-05`*.*

1 Agenda Telefônica em Java



- Aplicação que permita ao usuário gerir sua agenda de contatos
  - Adicionar, Remover, Atualizar
  - Listar dados de um contato, Listar todos contatos
- Para cada contato deve-se guardar o nome, sobrenome, data de nascimento, telefone(s) e e-mail(s)
- Para cada telefone ou email é necessário ter um rótulo. Ex:

  - **rótulo:** comercial
  - **valor:** email@empresa.com

- Ao cadastrar um email, deve-se garantir que é um endereço válido
- Ao listar um telefone, deve-se aplicar uma máscara para facilitar a leitura
  - Ex: +55048998761234 → +55 (48) 9 9876-1234
- Faça uso de tratamento de exceção de forma a garantir que o programa nunca finalize abruptamente diante de um dado fornecido pelo usuário.

2 Drone – Veículo aéreo não tripulado

Os drones[1][2] estão sendo empregados em diversas áreas, como em fazendas para vistoriar, vigiar e tomar algumas ações para aumentar a produtividade rural. Estes drones em questão possuem 4 rotores, 1 ou 2 câmeras de vídeo, sendo que ambas podem filmar também em infravermelho. A bateria permite uma autonomia de vôo de até 10 minutos, sendo que o consumo da bateria está relacionado com o número de manobras realizadas.

Os drones podem ser controlados a distância por um operador humano (fornecendo a todo instante comandos para movimentação) ou mesmo podem ser autônomos, seguindo um plano de vôo determinado previamente (se deslocando por um conjunto de coordenadas). Possuem um GPS e este é usado para determinar o local de onde decolou, para que depois seja usado para o retorno automático para casa; e também a sua atual posição.

Cada um dos rotores pode ser controlado de forma independente (a velocidade de rotação pode ir de 0 a 100%), permitindo assim alterar a direção e altitude do drone. As câmeras de vídeo podem ser acionadas para capturar uma foto, para iniciar ou parar a gravação de um vídeo.

Figura 1: Um drone com quatro rotores

Tarefas

1. Com base no texto descritivo acima, identifique as classes, seus atributos, principais métodos e os relacionamentos entre classes e faça um diagrama UML. Imagine que construirá uma classe para simular um drone real e que usuário poderá interagir com este drone, inclusive carregar um plano de voo. Crie um arquivo `Readme.md` e inclua nesse o arquivo `.png` com diagrama de classes UML criado.
2. Implemente em Java cada uma das classes identificadas no item anterior.

[1] veículos aéreos não tripulados
[2] https://pt.wikipedia.org/wiki/Quadrotor

3. Desenvolva um aplicativo Java (p.ex. classe `Principal`) que forneça um menu onde o usuário poderá interagir com um drone. Por exemplo:

   - Fazer a decolagem
   - Alterar altitude (subir ou descer)
   - Mover para uma direção (frente, trás, direita ou esquerda)
   - Qual sua posição atual (coordenadas GPS)
   - Carregar plano de voo e iniciar voo autônomo
   - Verificar o nível da bateria. Se o nível estiver com 10% ou menos, o drone deve iniciar o procedimento de pouso na mesma coordenada de onde decolou
   - Retornar para a base (na mesma coordenada de onde decolou)

É necessário que pense em uma lógica para a simulação das funcionalidades do drone a ser implementado e faça as simplificações necessárias para um simulador que tem a ambição de só imprimir mensagens no terminal e não ser um drone real. Por exemplo, toda vez que aumentar potência de um motor, o nível da bateria é decrementado em `XX%`. A movimentação poderia imprimir no terminal as coordenadas atuais, o nível de bateria atual, as coordenadas destino e o nível de bateria após a movimentação. As coordenadas GPS poderiam ser simplificadas como coordenadas em um plano cartesiano com três eixos `(x,y,z)` e a distância entre dois pontos[3] poderia ser usada para fazer o consumo da bateria. Para a captura de fotos bastaria imprimir no terminal "foto capturada nas coordenadas (x,y,z)", contudo seria vital guardar as fotos capturadas em memória.

Para a modelagem siga o princípio da **divisão de responsabilidades** (*Separation of Concerns* – SoC) e responsabilidade única (*Single Responsibility Principle – SRP*[4]). Ou seja, **Cada classe deve ser responsável por uma pequena parte** da funcionalidade de um software e a **responsabilidade** deve **estar completamente encapsulada** dentro da classe. Um número maior de classes é preferível a poucas classes.

> ⚠ **Atenção:**
>
> - ***Para receber nota 10:*** *Os commits para essa entrega deverão ser encaminhados (push) para o Github Classroom até às 12:30 do dia 01/04/2020;*
> - ***Para receber nota 5:*** *Os commits deverão ser encaminhados (push) para o Github Classroom até o dia 02/04/2020;*
> - *Qualquer entrega após o dia **02/04/2020** não será contabilizada e receberá nota 0;*
> - *Entrega parcial da lista receberá no máximo nota 1.*

[3] https://pt.wikipedia.org/wiki/Sistema_de_coordenadas_cartesiano#Dist%C3%A2ncia_entre_pontos
[4] https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple