Engenharia de Telecomunicações — Prof. Emerson Ribeiro de Mello
POO29004 – Programação Orientada a Objetos — mello@ifsc.edu.br

Projeto prático 01: Simulador de um elevador

03/07/2024

Plágio não é tolerado

Você deve ser o único(a) responsável por fazer a entrega para essa atividade. Todo o código ou texto deverá ser produzido exclusivamente por você, exceto trechos de códigos que possam ter sido fornecidos como parte do enunciado.

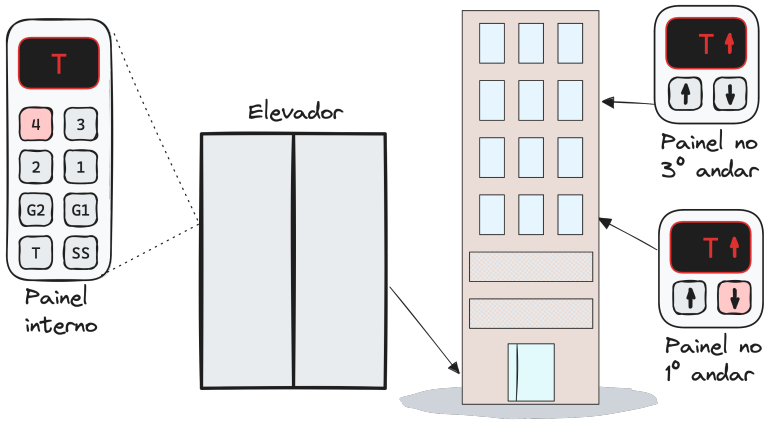
Você pode discutir com outros estudantes com o intuito de esclarecer pontos, isso é até incentivado, porém você não poderá copiar trechos de códigos, textos ou soluções de qualquer fonte (e.g. colegas da mesma turma ou de turmas anteriores, repositórios de códigos na Internet ou soluções providas por serviços como Copilot e ChatGPT).

1 Elevador

Na Figura 1 é apresentada uma ilustração de um elevador em um edifício. O elevador possui um painel de controle e este tem um botão para cada andar do edifício, além de um visor que indica o andar atual em que se encontra. Quando um botão é pressionado, o botão acende e o elevador inicia o movimento para o andar desejado. O usuário, dentro do elevador, pode visualizar o andar em que o elevador se encontra por meio do visor.

Em cada andar do edifício há um painel com dois botões e um visor que indica a direção do elevador quando em movimento (para cima ou para baixo), bem como o andar em que o elevador se encontra. O usuário pode chamar o elevador para subir ou descer. Assim, ao pressionar um botão, esse botão acende e o elevador inicia o movimento para o andar onde foi chamado.

Figura 1: Representação de um elevador



Com base no texto descritivo acima, bem como na representação gráfica do elevador, você deverá implementar um simulador de um elevador. A solução a ser desenvolvida por você deverá respeitar o princípio da divisão de responsabilidades (SOC, *Separation of Concerns*) e da responsabilidade única (SRP, *Single Responsibility Principle*). Ou seja, não deve-se criar uma única classe de forma que uma instância dessa classe assuma responsabilidades que seriam de outros objetos (lembre-se do exemplo Agenda de Contatos apresentado na aula de associação entre classes).

Apesar do exemplo indicar um edifício com 8 andares, entende-se que o número de andares do prédio é um dado variável e que deve ser passado como parâmetro para a aplicação. Assim, a aplicação deve ser capaz de simular o funcionamento de um elevador em um prédio com qualquer número de andares. Pode assumir que o primeiro andar é o andar 0 e que o último andar é o andar $n - 1$, onde n é o número total de andares do prédio.

A simulação consiste em permitir que o usuário possa chamar o elevador para subir ou descer em um andar ou que possa escolher um andar de destino quando estiver dentro do elevador. Deve-se permitir que o usuário possa visualizar o andar em que o elevador se encontra e a direção em que o elevador está se movendo, estando o usuário dentro do elevador ou em um andar. Assim, a cada ação realizada pelo usuário, o sistema deve imprimir na tela a situação do elevador, e dos painéis em cada andar. Veja um exemplo de saída da aplicação na Listagem 1. Os valores entre parênteses ao lado de cada botão indicam se o botão está aceso (1) ou apagado (0).

Listagem 1: Exemplo de saída da aplicação

```
Elevador | Andar atual: 0 | Direção: subindo
  Botões: 0 (0) 1 (0) 2 (0) 3 (0) 4 (0) 5 (0) 6 (0) 7 (0) 8 (1)
1o Andar: Elevador: subindo | Andar: 0 | Botões: subir (0) descer (1)
3o Andar: Elevador: subindo | Andar: 0 | Botões: subir (0) descer (0)
```

Na classe que possui o método `main`, você deve instanciar as classes necessárias para simular o funcionamento do elevador. Nessa classe, permita ao usuário interagir com elevador (chamando-o para subir ou descer em um andar, ou escolhendo um andar de destino quando estiver dentro do elevador).

Assuma que o deslocamento do elevador de um andar para outro demora um segundo. Você pode fazer uso do método `Thread.sleep(1000)` para simular esse tempo de deslocamento. Assuma que o elevador se movimenta de um andar para o outro de forma instantânea, ou seja, não há aceleração ou desaceleração. A cada iteração do laço que simula o funcionamento do elevador, você deve atualizar a situação do elevador e dos painéis em cada andar, e imprimir na tela.

Entregas e requisitos para desenvolvimento da solução

- ☐ Arquivo `.gitignore` adequado ao projeto
- ☐ Projeto Java com gradle, indicando e provendo corretamente as dependências de bibliotecas externas, caso haja necessidade
- ☐ Diagrama de classes UML (perspectiva de implementação) salvo em um arquivo chamado `modelagem.png` na raiz do repositório
- ☐ Respeitar o encapsulamento de dados, responsabilidade única e divisão de responsabilidades
- ☐ Fazer uso correto de constantes e não ter constantes literais espalhadas pelo código
- ☐ Comportamento correto das classes modeladas
- ☐ Comportamento correto da classe que possui método `main`.
- ☐ Garantir que seja possível compilar e executar o projeto após clonar o repositório via `git clone`. Na correção, o professor executará a aplicação com o comando: `./gradlew -q run`. Caso não seja possível executá-lo, então o projeto receberá o conceito 0.
- ☐ Arquivo `Readme.md` na raiz do repositório indicando quais funcionalidades foram implementadas e quais não foram.