# Trabalho Prático: Elevador

Prof. Daniel Conrado

## 1 Introdução

Neste trabalho, você deve criar um novo projeto no BlueJ que simule o funcionamento de um elevador de um prédio. Para não deixar o trabalho muito complexo, vamos assumir algumas coisas:

1. O prédio possui apenas um elevador.
2. Não vamos nos preocupar com o peso total dos passageiros.
3. Haverá alguma entidade, externa ao nosso projeto, que monitora as solicitações nos andares e que movimenta o elevador.
4. Normalmente os elevadores tendem a subir, tendem a descer, ou não estão para subir nem descer. Para simplificar, nós vamos desconsiderar esse último caso e entender o elevador como sempre tendendo a subir, ou sempre tendendo a descer.

As pessoas interagem com o sistema do elevador em dois cenários. No primeiro cenário, a pessoa está em um determinado andar do prédio e fora do elevador. As únicas coisas que a pessoa pode fazer é:

- solicitar o elevador para subir.
- solicitar o elevador para descer.
- esperar pela chegada do elevador.

Nesse cenário, a pessoa interage apenas com um painel contendo dois botões (cf. Figura 1). Interno a esse painel, há um mecanismo que percebe a chegada do elevador, fazendo-o parar se necessário.

No segundo cenário, a pessoa já está dentro do elevador. Dessa forma, ela pode interagir com o painel interno do elevador (Figura 2) e:

- pedir para o elevador fechar a porta.
- solicitar parada em um ou mais andares.
- esperar até o elevador parar.

Figura 1: O painel externo do sistema de elevador.

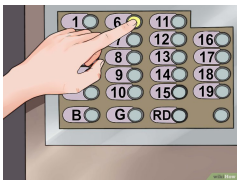


Figura 2: Painel interno do elevador.

### 1.1 Objetivo

O objetivo deste trabalho prático é criar um projeto Java que simule o funcionamento desse sistema de elevador e, com isso, explorar a ideia de como os objetos podem interagir para fornecer uma determinada funcionalidade. Você deve criar duas classes: 1) Uma classe chamada **Piso** para representar os andares do prédio com os painéis externos, permitindo às pessoas solicitarem o elevador; e 2) uma classe chamada **Elevador** para representar o elevador em si, contendo o painel para a pessoa solicitar parada nos andares.

### 1.2 Método

Desta vez, você deve criar o projeto do zero. Não fornecerei nenhum projeto de base. Contudo, eu vou descrever ao longo dos exercícios quais são os métodos e o comportamento esperado que você deve implementar. Findo o projeto, você deve enviar para mim, via Classroom, o código-fonte das classes, ou seja, os arquivos com extensão **.java**, sem estarem compactados.

## 2 Exercícios

**Exercício 1.** Comecemos pela classe mais simples. Você deve criar a classe **Piso** e escrever campos, construtores e métodos para atender o seguinte: um objeto da classe **Piso** deve possuir métodos para:

- retornar o andar do piso (método **getAndar**).
- sinalizar que alguém no piso quer chamar o elevador para subir (ou seja, quando a pessoa aperta o botão de 'subir' no painel externo). Nota: o piso não deve movimentar o elevador. Lembre-se que quem movimenta o elevador é uma entidade externa ao projeto. (Método **queroSubir**.)

- sinalizar que alguém no piso quer chamar o elevador para descer. (Método **queroDescer**.)
- retornar se alguém no piso quer subir. (Método **isQueroSubir**.)
- retornar se alguém no piso quer descer. (Método **isQueroDescer**.)
- *Imprimir na tela* uma mensagem representando a situação atual do painel. P. ex., considere um piso no andar 3. Se ninguém acionou o botão subir nem o botão descer, então a mensagem deve ser "3 △ ▽". Se o botão de subir estiver acionado, deve-se imprimir "3 ▲ ▽". Se apenas o botão de descer estiver acionado, deve-se imprimir "3 △ ▼". Finalmente, se ambos os botões estiverem acionados, deve-se imprimir "3 ▲ ▼". (método **mostrarPainel**)

Crie também um construtor que permita a criação de um objeto **Piso** somente se for informado o seu andar.

Pense com cuidado nos tipos de retorno dos métodos e também nos parâmetros (quais métodos precisam de parâmetros e quais não precisam).

**Exercício 2.** Formando uma cadeia de pisos

Para simular a estruturação de um prédio, modifique a classe **Piso** para que seja possível *conectá-los* uns aos outros. P. ex. em um prédio de 3 andares, o piso 1 deve estar conectado ao piso 2; o piso 2 deve estar conectado ao piso 1 e ao piso 3; e o piso 3, por ser o último andar, deve estar conectado apenas ao piso 2.

Um objeto da classe **Piso** deve fornecer um método para eu conectá-lo a um outro piso. Escreva o método **setPróximoPiso** para determinar o piso que está acima desse. P. ex. para que eu faça com que o **piso2** fique acima do **piso1,** eu chamo o método **setPróximoPiso** em **piso1** passando por parâmetro o **piso2**; ou seja, **piso1.setPróximoPiso(piso2);**. Faça com que esse método também conecte o **piso1** como o piso *anterior* ao **piso2**.

Crie também os métodos **getPróximoPiso** e **getPisoAnterior** para dar acesso a esses pisos. Você vai precisar de campos para armazenar o próximo piso e o piso anterior.

**Exercício 3.** Deixe por ora a classe **Piso** de lado. Voltaremos a ela depois. Crie a classe **Elevador**. Um objeto da classe **Elevador** deve fornecer métodos para:

- abrir a porta. (método **abrirPorta**)
- fechar a porta. (método **fecharPorta**)
- me dizer se a sua porta está aberta ou não. (método **isPortaAberta**)
- me dizer se a sua direção é para subir. (método **isSubindo**)
- me dizer se a sua direção é para descer. (método **isDescendo**)
- me dizer em que piso ele se encontra atualmente. (método **getPisoAtual**)

Crie um construtor para a classe **Elevador** que permita criar um objeto **Elevador** apenas pelo fornecimento do piso térreo. Ou seja, o elevador inicialmente está no primeiro piso do prédio. No construtor, defina que a porta está fechada e que sua direção é para subir (como ele está inicialmente no térreo, ele só pode subir).

**Exercício 4.** Este exercício envolve alterar ambas as classes. A ideia é fazer com que a pessoa, enquanto está dentro do elevador, consiga solicitar parada em um piso. Vamos criar um método que simula o pressionamento de um botão de piso do painel interno do elevador. Crie o método **parar**NoPiso. Ele deve receber como parâmetro um objeto da classe **Piso**. Esse método simplesmente avisa ao piso que ele deve parar o elevador quando ele chegar lá. Finja que a classe **Piso** possui o método **solicitarParada**, e que não recebe parâmetros.

Agora, escreva o método **solicitarParada** na classe **Piso**. Esse método deve apenas sinalizar que uma parada foi solicitada. Crie também o método **isParadaSolicitada** para retornar se uma parada foi solicitada ou não neste piso.

**Exercício 5.** Crie o método **mostrarPainel** para a classe **Elevador**. A ideia do método é *imprimir na tela* uma representação de como está o painel interno do elevador. O painel deve mostrar quais pisos têm uma parada solicitada e onde o elevador está no momento.

Vou deixar a saída deste método a critério criativo de vocês. Para exemplificar o que estou querendo, suponha que o prédio possua 5 andares, que tenha duas pessoas dentro do elevador no piso 2, e elas solicitaram parada nos andares 3 e 5. A saída do método **mostrarPainel** pode ser:

```
** Painel do Elevador **
 1   2  *3*  4  *5*
     ^
```

**Exercício 6.** Fazendo o elevador se mover

Agora, vamos fazer nosso elevador se mover entre os pisos. Lembre-se de que uma entidade externa vai comandar a subida e descida do elevador. Nossa responsabilidade aqui é apenas fornecer um método para o elevador subir ou descer um andar (e apenas um andar). Portanto, crie o método **mover**.

O comportamento esperado do método **mover** é o seguinte: se a direção do elevador for "subindo", então o método **mover** sobe o elevador para o andar de cima. Se a direação for "descendo", então o método desce o elevador para o andar de baixo.

Toda vez que o elevador mudar para um novo andar, ele deve avisar o novo piso de que ele chegou. O piso, por sua vez, verifica se o elevador precisa abrir a porta ou não naquele andar. Ou seja, é *do piso* a responsabilidade de parar o elevador. Daqui a pouco, falo quando que o piso deve parar ou não o elevador. Por ora, finja que os objetos da classe **Piso** possuem o método **receberElevador**, e que recebe por parâmetro um objeto da classe **Elevador**.

Há algumas condições que o método **mover** deve atender:

- Ele não deve mover o elevador enquanto ele estiver com a porta aberta.

- Se o elevador estiver no último piso, o método **mover** deve mudar a sua direção para "descendo" e descer um andar.
- Se o elevador estiver no primeiro piso, o método **mover** deve mudar a sua direção para "subindo" e deve subir um andar.

**Exercício 7.** BUT WAIT! THERE'S MORE!

Agora, você deve implementar o método **receberElevador** na classe **Piso**. Quando um piso receber o elevador, ele deve decidir se o elevador vai parar ou não. Há três situações em que o piso deve parar o elevador:

- quando o piso é uma parada solicitada por alguém dentro do elevador.
- quando alguém no piso sinalizou que quer subir e o elevador está subindo. Quando o elevador chegar, o sinal de "subir" do piso deve ser desligado.
- quando alguém no piso sinalizou que quer descer o elevador está descendo. Quando o elevador chegar, o sinal de "descer" do piso deve ser desligado.

Note que, se o piso estiver apenas com o "subir" ligado e o elevador estiver descendo, ele não para nesse piso! Para o piso fazer o elevador parar, ele deve abrir a sua porta (lembre-se que o objeto **Elevador** possui um método **abrirPorta**).

**Exercício 8.** OPCIONAL

Se você quiser, você pode criar uma interface de usuário (textual ou gráfica) para fornecer uma forma de interação com o sistema.