ULHT
Linguagens de Programação 2
LEI / LIG / LEIRT
2020/2021
Bruno Cipriano, Rodrigo Correia, Pedro Alves
v1.0.0

Guião Prático 2 - “Elevador”

Introdução

Este guião tem como objetivo ajudar-vos a progredir no exercício do sistema do “Elevador”.

Este guião é baseado no sistema do “Elevador”, que foi o TPC teórico #2. Neste guião vamos implementar uma das possíveis soluções.

Exercício 1 - Criação de projecto no IDE

1.1 - Comece por criar um projecto no seu IDE (NetBeans ou outro)

O seu projecto deve conter as seguintes classes:

- Classe principal do seu programa
- Elevador
- Pessoa
- SensorPeso
- Porta

Clarificação

- Na classe `Elevador`, a “capacidade” corresponde ao número máximo de pessoas;

- Desta forma, cada `Elevador` irá ter dois limites físicos:
    - capacidade - controlado pelo próprio `Elevador`
    - peso máximo - controlado pelo `SensorPeso`

Exercício 2 - Novos atributos

2.1) Adicione à sua classe `Pessoa` um atributo que permita representar o nome da pessoa.

(O tipo de dados deste novo atributo fica ao seu critério, mas deverá ser justificado.)

2.2) Implemente um acessor (chamado p.e. `obterNome()` ou `getNome()`, conforme preferia) que lhe permita obter o valor do atributo nome de um objecto da classe Pessoa.

Exercício 3 - Construtores

Nestas alíneas vamos implementar alguns constructores para as classes do nosso modelo.

Lembre-se que o construtor:

É um método que :

- tem o mesmo nome que a classe;
- é executado sempre que se cria uma instância / objecto da classe;
- não tem tipo de retorno (nem sequer *void*);
- deve deixar o objecto num estado “completo” - isto é, deve inicializar os atributos conforme seja necessário para o correcto funcionamento da classe.

Tenham também em conta que uma classe pode ter vários construtores, desde que os mesmos tenham argumentos diferentes.

3.1) Classe SensorPeso

Esta classe representa um sensor que é capaz de medir o peso das várias pessoas que estão dentro do elevador.

Quando criamos um objecto `SensorPeso`, que valor faz sentido atribuir ao atributo `pesoActual`?

Partindo do princípio que o objecto é criado antes de haver peso no elevador, vamos assumir que o peso inicial será zero.

Criamos então um construtor que apenas fará a atribuição do valor 0 ao atributo `pesoActual`:

```
class SensorPeso {
    int pesoActual;

    public SensorPeso () {
         pesoActual = 0;
    }
}
```

Também temos a opção de criar um construtor que receba o valor para inicializar o argumento pesoActual.

Exemplo:

```
class SensorPeso {
    int pesoActual;
    public SensorPeso () {
         pesoActual = 0;
    }
    public SensorPeso (int peso) {
         pesoActual = peso;
    }
}
```

Neste caso ficámos com uma situação em que a classe tem dois construtores. Ao criar o objecto, o Java consegue distinguir qual deles será invocado considerando os tipos e número de argumentos que forem passados.

3.2) Classe Porta

Esta classe representa a Porta do elevador. O único atributo que a classe contém é o booleano `aberta`, que terá o valor *true* caso a porta esteja aberta e *false* em caso contrário.

a) **Que valor faz sentido para inicializar este atributo?**

b) **Crie o construtor e faça a implementação respectiva.**

```
class Porta {

    /* o seu código …*/

}
```

3.3) Classe Pessoa

A classe pessoa tem dois atributos:

- peso
- nome

Como esses dados vão variar de Pessoa para Pessoa, é difícil arranjar valores por omissão que façam sentido. Assim sendo, vamos definir o construtor da classe como recebendo os valores para os atributos.

```
class Pessoa {

    int peso;
    String nome;

    public Pessoa(String nome, int peso) {

        this.peso = peso;

        /* Faça a inicialização do nome... */

    }

}
```

Neste exemplo usou-se a palavra reservada **`this`** de forma a desambiguar entre o argumento “peso” do método e o atributo “peso” da classe.

(Uma alternativa era ter um nome diferente para o argumento, mas isso nem sempre resulta em nomes claros e que comuniquem a quem lê o código o significado do atributo).

3.4) Classe Elevador

Esta classe tem uma maior variedade de atributos, sendo alguns deles de classes do nosso próprio modelo/sistema.

Pretendemos que o nosso constructor deixe o objecto num estado completo. Ou seja:

- vamos inicializar o atributo `capacidade`
    - (usando o argumento `capacidade`)

- vamos inicializar o atributo `porta` com um **novo objecto** da classe `Porta`
    - (que inicialmente está “aberta”)

- vamos inicializar o atributo sensor com um **novo objecto** da classe `SensorPeso`

- vamos inicializar o atributo que corresponde à lista dos `ocupantes`
    - (que inicialmente está vazia)
    - Será semelhante ao que fizemos no exercício do `Autocarro` - ao início, a lista de passageiros estava vazia.

a) Na classe `Elevador`, vamos criar um constructor que deverá receber dois argumentos:

```
Elevador (int capacidade, int pesoMaximo) { }
```

b) Vamos começar por inicializar os **atributos capacidade e pesoMaximo**, usando os valores do argumentos `capacidade` e `pesoMaximo`.

Exemplo:

```
public Elevador (int capacidade, int pesoMaximo) {
     this.capacidade = capacidade;

     /* Faça a inicialização do atributo pesoMaximo */
}
```

c) De seguida, vamos inicializar o atributo **sensor**

Este atributo é do tipo (classe) `SensorPeso`. Por essa razão, para o inicializar, temos de criar um objecto `SensorPeso`.

Vamos considerar que o peso actual quando se cria o Elevador é zero. Sendo assim, podemos invocar o construtor de `SensorPeso` que não recebe o argumento `pesoActual`.

```
public Elevador (int capacidade, int pesoMax) {

    this.capacidade = capacidade;

    /* Faça a inicialização do atributo pesoMaximo */

    sensorPeso = new SensorPeso();

    // Devem preencher o código em falta aqui
    // (têm de inicializar a Porta e os ocupantes)

}
```

d) Faça a inicialização do atributo porta

Deverá ser um objeto da classe Porta.

e) Faça a inicialização do atributo ocupantes

Deverá ser uma “lista de pessoas” vazia (`ArrayList<Pessoa>`). É semelhante ao que foi feito no Guião do Autocarro para representar os passageiros.

4) Novos métodos

4.1) Na classe `Elevador`, vamos criar um método `mostrarLimites()` que irá apresentar no ecrã os limites do elevador, quer em número de passageiros, quer em peso máximo.

Este método deverá apresentar os dados da seguinte forma:

```
Peso Máximo: … (kg)
Capacidade Máxima: … (pessoas)
```

Exemplo:

```
/* Este método não vai retornar nada, apenas vai mostrar dados no
ecrã. Assim sendo, qual deve ser o seu tipo de retorno ? */

public _____ mostrarLimites() {

}
```

4.2) Ainda na classe `Elevador`, vamos criar um método `mostrarOcupantes()` que irá apresentar no ecrã os nomes dos ocupantes do elevador.

Caso a lista de ocupantes esteja vazia, a função deverá apresentar a mensagem:
"O elevador encontra-se vazio neste momento."

Nota: tente implementar este método sem usar o método `get()` da classe ArrayList.

5) Criação de um objecto Elevador

5.1) Na classe principal do nosso programa, devemos ter o implementado o método `main()`.

5.2) Nesse método vamos criar um objecto `Elevador`, com capacidade para:

- 210 (kg) de peso
- 3 pessoas

Exemplo:

```
public static void main(String[] args) {

    // Criar o objecto Elevador e inicializá-lo
    // (usando o construtor que recebe os dois limites)

}
```

5.3) Ainda no método main, vamos **invocar** (chamar) o método `mostrarLimites ()` sobre o objecto `Elevador` que acabámos de criar, para verificar se os dados estão correctos.

```
public static void main(String[] args) {
    // Criar o objecto e inicializá-lo
    // Elevador elevador = …

    // Invocar o método mostrarLimites() sobre o objecto criado
}
```

5.4) Vamos executar o projecto.

Neste momento, se tudo estiver bem, o *output* deverá ser o seguinte:

```
Peso Máximo: 210 (kg)
Capacidade Máxima: 3 (pessoas)
```

Caso não tenha obtido o *output* indicado, verifique as suas implementações.

5.5) Ainda na função `main()`, vamos invocar o método `mostrarOcupantes ()` sobre o nosso objecto Elevador, e executar novamente o projecto.

O *output* desta segunda execução deverá ser:

```
Peso Máximo: 210 (kg)
Capacidade Máxima: 3 (pessoas)
O elevador encontra-se vazio neste momento.
```

Caso não tenha obtido o *output* indicado, verifique as suas implementações.

6) Adicionar Pessoas ao Elevador

Neste exercício vamos controlar o acesso ao `Elevador`.

Para este exercício é necessário ter na classe `Elevador` o método:

```
boolean entrar(Pessoa p)
```

6.1) Na classe principal, e no método `main(String[] args)` vamos criar quatro objectos `Pessoa`:

- o Zeca, com peso 80 kg
- a Joana, com peso 60 kg
- o Afonso, com peso 78 kg
- a Rita, com peso 62 kg

Exemplo:

```
public static void main(String[] args) {

    (...)

    Pessoa p1 = new Pessoa("Zeca", 80);

    // Devem adicionar aqui o código para criar as outras 3 Pessoas

}
```

b) De seguida, vamos adicionar as pessoas, uma a uma, e verificar se realmente entraram no elevador.

Como é que sabemos se uma pessoa entrou no Elevador?

Lembre-se que o método `boolean entrar(Pessoa p)` retorna `false` quando não é possível adicionar a pessoa ao `Elevador`.

Exemplo:

```
public static void main(String[] args) {

    (...)

    Pessoa p1 = new Pessoa("Zeca", 80);
```

```
    (...)

    boolean r1 = elevador.entrar(p1);

    String nome = p1.obterNome ();

    if(r1) {
        String texto = nome + " entrou no elevador.";
        System.out.println(texto);
    }
    else {
        String texto = nome + " não entrou no elevador.";
        System.out.println(texto);
    }

}
```

Nota: este código assume que existe uma variável chamada **`elevador`** que aponta para um objecto da classe `Elevador` (criado no item **5.2** deste guião).

c) Podíamos repetir o processo anterior para as quatro pessoas. No entanto, isso iria levar a uma excessiva repetição de código.

Vamos tentar fazer o mesmo com um `ArrayList<Pessoa>` e um ciclo:

Exemplo:

```
import java.util.ArrayList;

(...)

public static void main(String[] args) {

    (...)

    Pessoa p1 = new Pessoa("Zeca", 80);
```

```
    (...)

    List<Pessoa> pessoas = new ArrayList<Pessoa>();

    pessoas.add(p1);

    // Completem com o código para adicionar o resto das pessoas
    // ao ArrayList
    // Devem respeitar a ordem: Zeca, Joana, Afonso, Rita

    (...)

    for(int i=0; i<pessoas.size(); i++) {

        Pessoa p = pessoas.get(i);

        boolean r1 = elevador.entrar(p);

        // Completem com o código que verifica se a pessoa entrou
        // e apresenta a mensagem correspondente

        (...)

    }
}
```

(Atenção ao `import`.)

d) Vamos executar o projecto.

Neste momento, se tudo estiver bem, o *output* deverá ser o seguinte:

```
Peso Máximo: 210 (kg)
Capacidade Máxima: 3 (pessoas)

Zeca entrou no elevador.
Joana entrou no elevador.
Afonso não entrou no elevador.
Rita entrou no elevador.
```

O que é correcto:

- Inicialmente o elevador tem espaço para três pessoas e 210 kg;
- Ao entrar o Zeca, passou a ter espaço para mais duas pessoas e 130 kg;
- Ao entrar a Jana, passou a ter espaço para mais uma pessoa e 70 kg;
- Apesar de ainda haver espaço para mais uma pessoa, quando o Afonso tentou entrar não foi possível porque ele pesa mais do que 70 kg;
- Quando a Rita tentou entrar, isso foi possível, pois ela pesa menos do que 70 kg.

Notem que este *output* seria diferente caso as pessoas fossem introduzidas no Elevador por outra ordem.

e) Indique qual seria o *output* do programa caso se tentassem adicionar as pessoas pela seguinte ordem:

1º Zeca
2º Afonso
3º Joana
4º Rita

Exercício 7 - Verificar estado do objecto Elevador

7.1) Na classe `Elevador` vamos criar um método `mostrarEstado()` que vai apresentar os seguintes dados do Elevador:

- capacidade (número de pessoas)
- peso máximo (kg)
- ocupação actual (número de pessoas)
- peso actual (kg)

7.2) Voltar à classe `Main` e ao método `main`. No final desse método vamos pedir à instância da classe `Elevador` para mostrar o seu estado, invocando para isso o seu método `mostrarEstado()`.

7.3) Vamos executar novamente o projecto.

Devemos obter o seguinte output:

```
Zeca entrou no elevador.
Joana entrou no elevador.
Afonso não entrou no elevador.
Rita entrou no elevador.
Capacidade: 3 pessoas
Peso máximo: 210 (kg)
Ocupação actual: 3
Peso actual: 202 (kg)
```

Caso não tenha obtido o *output* indicado, verifique as suas implementações.

FIM