P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Projeto SO

REALIZADO POR:

ABILIO CASTRO – 8170054 | RICARDO CARDOSO – 8170278 | VITOR SANTOS - 8170312

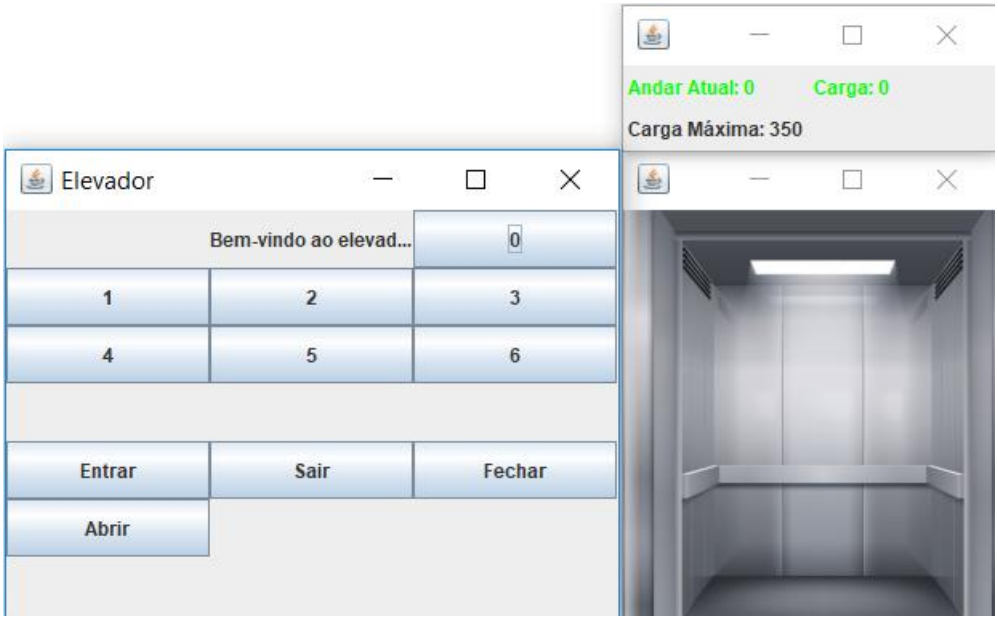
Índice

Manual de Configuração

É possível alterar o número de pisos do elevador e a carga máxima do mesmo no ficheiro “elevador” que se encontra na pasta do projeto.

Quanto à utilização da aplicação pensamos que é bastante intuitiva mas as suas funcionalidades são melhor descritas no ponto abaixo “Descrição das Funcionalidades Implementadas”

Descrição das Funcionalidades implementadas



Na janela com o nome elevador temos a botoneira, os botões com número representam os pisos (o numero de pisos é dado por um ficheiro externo), o botão “Entrar” simula a entrada de uma pessoa no elevador e incrementa 60 kilos à carga, o botão “Sair” simula a saída de uma pessoa no elevador e decrementa 60 kilos à carga, o botão “Fechar” fecha as portas do elevador e o botão “Abrir” abre as portas do elevador.

À direita da botoneira temos a representação gráfica das portas do elevador (que nesta imagem se encontram abertas).

Em cima das portas temos uma espécie de display que nos mostra o andar atual, a carga e a carga máxima (a carga máxima é dada por um ficheiro externo).

O elevador está feito de modo a que:

- Caso sejam pressionados muitos pisos, os pisos são guardados numa queue e são percorridos pela ordem que forem pressionados.
- O elevador demora 2 segundos entre cada piso.
- Depois de chegar a um piso selecionado o elevador abre as portas durante 5 segundos, caso estejam mais pisos na queue o elevador fecha as portas e segue para o próximo piso, não havendo mais pisos na queue o elevador apenas fecha as portas e fica no piso onde está.
- É impossível abrir as portas se o elevador se encontrar em movimento, caso o elevador esteja parado, o botão de “Abrir” mantém as portas abertas durante 5 segundos ou até o botão “Fechar” ser pressionado.
- É impossível entrar ou sair pessoas do elevador (adicionar ou remover carga) enquanto ele está em movimento.
- Se a carga máxima for excedida e for pressionado um piso, o elevador só iniciará movimento quando saíram pessoas suficientes de modo a que a carga seja menor que a carga máxima. (A carga é representada a verde se for inferior ou igual à carga máxima e a vermelho se for superior à carga máxima).
- Depois de cada piso é guardado num ficheiro “relatório” o pisos percorridos e a carga que levava em cada piso.

Mecanismos de sincronização e comunicação entre módulos

SharedOBJ

O SharedOBJ é uma classe que usamos para reter toda a informação do elevador relevante para o nosso programa como:

- O piso onde se encontra
- O piso onde pretende ir
- Se está em movimento ou não
- Se as portas estão abertas ou não
- A carga atual
- A carga máxima
- Um semáforo que vai ser explicado mais à frente no relatório

```
public class SharedOBJ {

    Semaphore semP = new Semaphore(1);
    private int currentFloor;
    private int pisoS;
    private boolean moving = false;
    private boolean doorsOpen = true;
    private int carga = 0;
    private int maxCarga;
```

Threads:

Durante a execução do programa são lançadas Threads das seguintes classes:

- **Motor**

  Classe responsável por mover o elevador para cima e para baixo

- **Portas**

  Classe responsável por abrir e fechar as portas

- **Botoneira**

  Classe responsável por dizer quais os botões pressionados

- **Display**

  Classe responsável pela representação gráfica das portas e do display em cima das portas

Semáforo nº1:

O primeiro semáforo está a ser usado no modulo do Motor, o objetivo deste semáforo é prevenir que seja lançadas várias Threads da classe Motor caso sejam pressionados vários pisos.

Numa parte mais inicial do projeto quando ainda não tínhamos este semáforo implementado surgiu-nos o seguinte problema:

Quando nos situávamos por exemplo no piso 4 e eram pressionados os botões para o piso 6 e piso 2, o que acontecia é que eram lançadas duas Threads do Motor uma delas tentava fazer o elevador subir e outra o elevador descer o que resulta no elevador subir para o piso 5 e de seguida descer para o 4 e assim continuava num loop infinito.

A maneira que usamos para resolver este problema foi com a utilização de um semáforo, basicamente o que faz é só deixar uma thread aceder á parte que move o elevador ao mesmo tempo e assim que essa acabasse de mover o elevador entraria a próxima.

```
public synchronized void run() {

    try {
        sem.acquire();
        shl.setPisoS(queue.poll());
        System.out.println("Entrei:" + shl.getPisoS());

    } catch (InterruptedException ex) {
        Logger.getLogger(Motor.class.getName()).log(Level.SEVERE, null, ex);
    }

    while (shl.isDoorsOpen() || shl.getCarga() > shl.getMaxCarga()) {
        try {
            Thread.sleep(2000);
        } catch (InterruptedException ex) {
            Logger.getLogger(Motor.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    shl.setMoving(true);
    if (shl.getPisoS() > shl.getCurrentFloor()) {
        movingUp();
    } else if (shl.getPisoS() < shl.getCurrentFloor()) {
        movingDown();
    } else {
        sem.release();
```

O acquire é feito no inicio do método run(), e só é feito o release no final dos métodos movingUp(), movingDown() ou no caso do elevador se encontrar no piso que foi pressionado.

Semáforo nº2:

Este é o semáforo mencionado em SharedOBJ, a funcionalidade deste semáforo é abrir as portas automaticamente quando o elevador se para de mover.

```
            try {

                shl.semP.acquire();
            } catch (InterruptedException ex) {
                Logger.getLogger(Portas.class.getName()).log(Level.SEVERE, null, ex);
            }

            try {
                shl.semP.acquire();
            } catch (InterruptedException ex) {
                Logger.getLogger(Portas.class.getName()).log(Level.SEVERE, null, ex);
            }
            openDoorsAF();
        }
    }
```

O semáforo é iniciado com apenas uma permissão, é feito o primeiro acquire e só é feito o release quando o motor para de trabalhar, ou seja o elevador já não se encontra em movimento, permitindo então passar o segundo acquire que abre as portas do elevador, depois das portas estarem abertas é feito o release que permite passar o primeiro acquire.

E foi desta maneira que arranjamos a que as portas se abram automaticamente quando o elevador para num piso.

Funcionalidades pedidas e não implementadas

Funcionalidades pedidas:

- Módulo “Main”
- Módulo “Botoneira”
- Módulo “Motor”
- Módulo “Portas”
- Funcionalidades Avançadas:
    - Botão stop “S”
    - Botão bloqueio “K”
- Outras funcionalidades:
    - Configuração do elevador através da leitura de um ficheiro
    - Registo de atividade (log)

Funcionalidades não implementadas:

- Funcionalidades Avançadas:
    - Botão stop “S”
    - Botão bloqueio “K”