

**Instituto Politécnico de Setúbal**

Escola Superior de Tecnologias de Setúbal

Ano Letivo: 2015/2016

Docente: Rossana Santos

**Simulação de uma Interface Modal**

Manual Técnico

Sistemas Operativos



**Trabalho realizado por:**

140221002 – Luís Mestre

140221017 – André Bastos

Turma 2º INF-ES-03/04

**Índice**

[**Introdução** 2](#_Toc440826313)

[**Objetivos** 3](#_Toc440826314)

[Tempo de espera dos passageiros pelo transporte; 3](#_Toc440826315)

[Número de passageiros que embarcaram dentro do horário; 3](#_Toc440826316)

[Número de passageiros que embarcaram fora do horário; 3](#_Toc440826317)

[Minutos de atraso por cada comboio e autocarro; 3](#_Toc440826318)

[**Código e Aplicação** 4](#_Toc440826319)

[**Ficheiros** 4](#_Toc440826320)

[**Classes** 4](#_Toc440826321)

[**Classes Necessárias:** 4](#_Toc440826322)

[**Comportamento das threads e os seus métodos:** 5](#_Toc440826323)

[**Comboio/Autocarro** 5](#_Toc440826324)

[**Avião** 5](#_Toc440826325)

[**Relógio** 6](#_Toc440826326)

[**Resultados obtidos:** 7](#_Toc440826327)

[**Autocarros:** 7](#_Toc440826328)

[**Comboios** 8](#_Toc440826329)

[**Resultados dos aviões retidos:** 8](#_Toc440826330)

[**Limitações do programa:** 9](#_Toc440826331)

# **Introdução**

Este manual tem como objetivo a descrição detalhada e explicação de todas as técnicas e métodos usados no projeto Interface Modal (IM) no âmbito da disciplina de Sistemas Operativos.

A aplicação trata-se de simular uma rede de transportes públicos coletivos que têm um horário para cumprir, para que consiga ter o máximo de lugares preenchidos. Esta simulação é feita com uma IM que contem terminais para comboios e para autocarros. Este encontra-se nas imediações de um aeroporto.

A aplicação foi desenvolvida em linguagem de programação JAVA, no Software NetBeans, no sistema operativo Windows, utilizando técnicas adequadas de sincronização/comunicação entre processos e também o uso de Threads, que permitem múltiplas execuções independentes no mesmo ambiente de processo.

Além de tudo o que seja necessário para o correto funcionamento da aplicação, o manual técnico irá constar os seguintes pontos:

* Apresentação da aplicação, introdução teórica e justificação de todas as opções de implementação;
* Exemplo de funcionamento da simulação (com “print screens”);
* Listagem das funções implementadas mais importantes e respetiva explicação, algoritmo e código fonte associado;
* Resultado da simulação para vários cenários e estudo de melhoramento do funcionamento da empresa;
* Análise das limitações do programa;
* Código fonte comentado legível.

# **Objetivos**

O desenvolvimento da aplicação Interface Modal teve em conta vários objetivos, entre eles são fazer a simulação correr e acabar de acordo com os horários dos transportes, sendo o resultado mais esperado ser os aviões estarem vazios no final da simulação.

No final da simulação iremos mostrar também os seguintes resultados:

## Tempo de espera dos passageiros pelo transporte;

O tempo de espera dos passageiros, que começa a contar a contar desde o momento que o avião aterrou.

## Número de passageiros que embarcaram dentro do horário;

São os passageiros que entraram dentro do horário que está previsto ser cumprido pelo transporte (comboio ou autocarro).

## Número de passageiros que embarcaram fora do horário;

São os passageiros que entraram no tempo de tolerância do transporte (comboio ou autocarro).

## Minutos de atraso por cada comboio e autocarro;

Os minutos de atraso são normalmente os minutos que o transporte fica à espera pelo avião que está a chegar com passageiros para o mesmo, ou seja, o tempo de tolerância cumprido.

Acrescentamos também o resultado dos aviões que ficaram retidos no aeroporto com passageiros, e também os que não tiveram hipóteses de aterrar devido aos que ficaram retidos no aeroporto. Mostramos os grupos que ficaram no avião, para que transporte pretendiam embarcar e o seu destino.

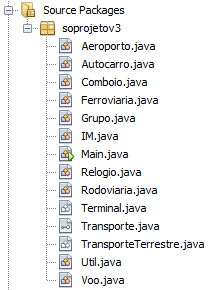
# **Código e Aplicação**

## C:\Users\Luís Mestre\Desktop\Screenshot_1.png**Ficheiros**

Temos os ficheiros que representam os Autocarros, Comboios e Aviões desta simulação.

NOTA: a explicação da estrutura de cada ficheiro está explicado no manual de utilizador.

## **Classes**

Pelo que tínhamos interpretado do enunciado decidimos criar as classes:

**Autocarro, Comboio, Voo**, que representam os transportes da simulação. Estas classes também implementam a interface Runnable visto que elas é que vão ditar os resultados e comportamento da simulação. As classes Comboio e Autocarro são classes filhas da classe TransporteTerrestre, por terem atributos bastante semelhantes e em comum. Tendo os Transportes Terrestres e os voos um comportamento semelhante e uns métodos em comum fizemos com que implementassem esses métodos da interface Transporte;

**Aeroporto**, que irá ter os aviões da simulação “estacionados” e as classes **Rodoviaria, Ferroviaria** que representam os terminais onde os autocarros e comboios iram “estacionar”, sendo elas classes filhas da classe Terminal;

**Grupo**, que irá representar os grupos que estão nos aviões;

**Interface Modal**, que irá ter a Rodoviaria e Ferroviaria e irá ser o meio de envio de passageiros entre os aviões que estão no aeroporto e os comboios/autocarros que estão “estacionados” nos terminais dentro do IM.

### **Classes Necessárias:**

Criámos as classes Relogio e Util por nos serem convenientes para a simulação e para o bom funcionamento da simulação. Em suma:

O Relogio implementa a interface Runnable. Esta classe vai nos dizendo o tempo atual da simulação e vai verificando se todas as threads acabaram o seu comportamento. Esta também vai iniciar também as threads autocarros, comboios e aviões de acordo com a respetiva hora de chegada programada.

O Util é uma classe que tem métodos que são úteis para o decorrer e execução da simulação.

## **Comportamento das threads e os seus métodos:**

### **Comboio/Autocarro**

Ambos os transportes têm um comportamento semelhante, em suma o que eles fazem como comportamento de thread é o seguinte:

- Primeiro tentam estacionar no seu respetivo terminal e na sua plataforma ou linha de embarque, caso consiga estaciona no terminal e fica em wait, caso não consiga não estaciona (visto que a plataforma ou linha de embarque que quer ocupar está ocupada) fica em wait e depois é acordado e tenta novamente estacionar.

- Depois quando chegar a hora de partida programada é acordada e vai verificar se precisa de esperar por mais algum avião, caso tenha que esperar o transporte espera até o avião chegar ou até cumprir o tempo de tolerância por ficar à espera pelo avião. Depois de partir ele é removido do sítio onde está estacionado, avisa os transportes que estão à espera que ele saiu, é guardado a hora de partida e por último é guardado as informações do transporte para os resultados.

#### **Métodos Principais:**

**- ocuparLugar()**: verifica se um transporte pode estacionar usando o método canPark(), em caso de puder ele estaciona no terminal e adormece, caso não possa ele simplesmente adormece sem efetuar o estacionamento;

**- hasMorePassengers()**: verifica se existe algo avião que ainda não aterrou e se o avião está programado para chegar durante o tempo de tolerância do transporte, caso haja então vai verificar se existem passageiros para esse mesmo transporte e caso haja verifica se tem lugares suficientes para os passageiros, retorna true se forem cumpridos essas condições, retorna false caso contrário;

#### **Métodos auxiliares:**

**- canPark()**: verifica se o transporte pode ou não estacionar na sua linha de embarque ou plataforma, retorna true caso possa estacionar, retorna false caso contrário;

**- removeTransporte()**: método que retira o transporte do terminal e guarda a hora de partida do transporte;

### **Avião**

O avião, ao contrário dos comboios e dos autocarros, não pode fazer wait em nenhuma ocasião exceto quando for para aterrar. Pois pode haver o caso de querer aterrar e a porta de embarque a que quer estacionar esteja ocupado, aí terá de esperar até que o avião saia da porta de embarque. Em suma, o comportamento do avião é o seguinte:

- Primeiro o avião tenta aterrar na porta de embarque designada, após este aterrar ele tenta entregar os passageiros aos seus respetivos transportes.

- Depois de conseguir entregar todos os passageiros antes de a simulação acabar ele sai do aeroporto e avisa a quem esteja à espera para aterrar que ele saiu. Caso a simulação tenha acabado e o avião não conseguiu entregar os passageiros todos então ele guarda as informações dos respetivos passageiros que não conseguiram embarcar no seu respetivo transporte.

#### **Métodos principais:**

**- ocuparLugar()**: tal como nos autocarros e nos comboios, verifica se o avião pode aterrar usando o método canLand(), em caso de puder ele aterra na porta de embarque respetiva e adormece, caso não possa ele simplesmente adormece sem aterrar;

- **isEmpty()**: verifica se o avião ficou ou não vazio, caso tenha ficado vazio retorna true, caso contrário retorna false;

- **entregarPassageiros(Grupo grupo)**: método da classe IM que recebe um grupo de um avião e tenta entregar os passageiros ao seu respetivo transporte, caso consiga tira os passageiros do avião e “atira-os” para o transporte;

- **removeTransporte(int nPortaEmbarque)**: método da classe Aeroporto que retira o avião do aeroporto.

#### **Métodos auxiliares:**

- **canLand()**: método que verifica se um avião pode aterrar, caso consiga aterrar retorna true, caso contrário retorna false;

- **hasAutocarrosComboiosEnded()**: método da classe Relogio que verifica se todos os autocarros e comboios acabaram o seu comportamento, isto é, se já todos partiram.

### **Relógio**

O relógio, tal como foi referido anteriormente, foi uma classe implementada por necessidade para uma melhor estrutura e organização na simulação.

O comportamento desta thread é o seguinte:

- Começa por iniciar as Listas de todos os transportes que irão estar presentes na simulação. Depois de ter as listas de todos os transportes a cada minuto da simulação vai vendo qual transporte pode iniciar comportamento de acordo com a sua hora de chegada programa.

- De seguida vai acordando cada transporte (comboio ou autocarro) quando chegar a hora de partida de cada um dos que estão estacionados.

- Depois de todos os transportes terem acabado o seu comportamento, ele acorda os aviões que não aterraram e avisa os aviões que estão retidos no aeroporto (caso existam) e avisa que a simulação acabou. No final mostra o resultado de todos os transportes da simulação.

#### **Métodos Principais:**

- **iniciarListas()**: método que inicia as listas dos transportes que vão participar na simulação;

- **iniciarTransporte()**: método que percorre as listas e inicia os transportes quando atingir a sua respetiva hora de chegada;

- **wakeUpSleepyHeads()**: método da classe IM que acorda o transporte que está estacionando (autocarro ou comboio) caso esteja na sua respetiva hora de partida;

- **hasAllTransTerrestreEnded()**: método que verifica se todos os Autocarros e Comboios acabaram o seu comportamento, retorna true caso tenham todos acabado, retorna false caso contrário;

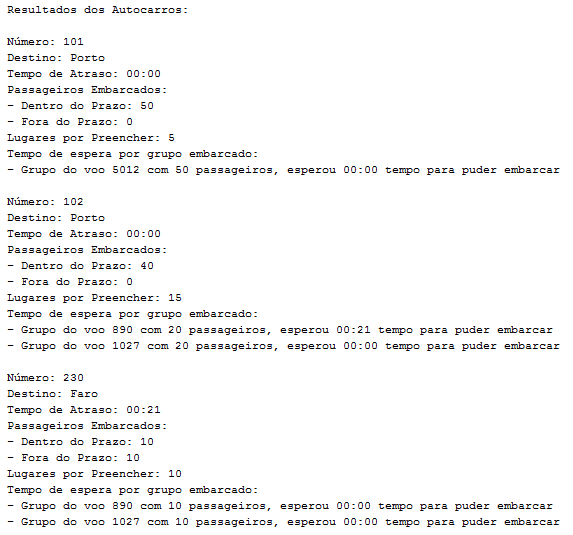
- **hasAllVoosEnded()**: método que verifica se todos os aviões acabaram o seu comportamento, retorna true caso tenham todos acabado, retorna false caso contrário;

- **resultados()**: método da classe IM que imprime todos os resultados de cada transporte quando a simulação acabar.

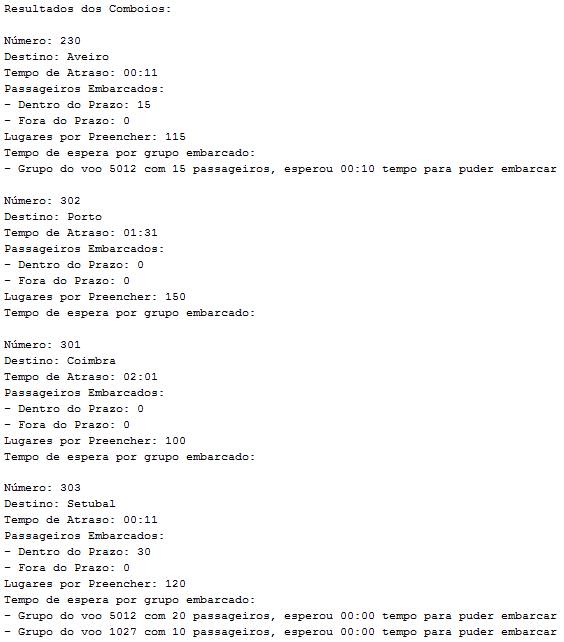
# **Resultados obtidos:**

Para obtermos estes resultados iniciamos uma simulação com os ficheiros que nos foram fornecidos pela professora da cadeira. Os resultados foram os seguintes:

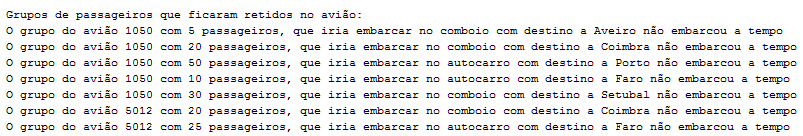
## **Autocarros:**



## **Comboios**



**Resultados dos aviões retidos:**  
(podem não haver, sendo o melhor resultado não haver nenhum retido)



# **Limitações do programa:**

No final foram encontradas algumas pequenas falhas sobre a aplicação que não foram corrigidas ou melhoradas a tempo da data de entrega.

- Por vezes o tempo de atraso dos transportes nos resultados aparece com 1 minuto extra, tal como se pode ver nos “print-screens” que pusemos nos resultados obtidos;

- Quando se tem mais do que um transporte para estacionar/aterrar na mesma linha/porta de embarque ou plataforma nem sempre quem estaciona é quem ficou primeiro à espera.