UNIVERSIDADE CATÓLICA DE ANGOLA

FACULDADE DE ENGENHARIA

Curso: Engenharia Informática

Cadeira: Fundamentos de Programação III

Ano Lectivo: 2021/22

RELATÓRIO

Simulador de Aterragem e Decolagem de um Aeroporto

Autor:

ID 1000026185

Nome Gilberto Alexandre Adão de Jesus

Turma AINF

Data 14/02/2022

Docente Engº Pedro Mbote

Resumo

O simulador visa a fazer um gerenciamento cauteloso e seguro do tráfego de aviões em um aeroporto simulado, de modos que se evitem acidentes aéreos e haja um melhor controlo de quais aeronaves entram ou saem do aeroporto.

Luanda – 2021/22

1. Introdução

O aeroporto é um lugar onde é feito o tráfego de n-aviões estejam eles vindo para aterrar ou saindo a decolar, desta feita foi encarregado desenvolver uma aplicação de simulação de decolagem e aterragem que visa a ter uma ideia de como é feito o gerenciamento do tráfego de aviões nos aeroportos para que não haja acidentes aéreos e perdas desnecessárias de vidas.

Portanto, para o desenvolvimento da aplicação foi pré-definido um cenário com poucas pistas (3 pistas), que podem ser usadas tanto para aterragem como decolagem, o cenário é o seguinte:

- São três pistas, todas podem ser usadas para decolagem, apenas pode ser feito uma aterragem normal nas duas primeiras pistas, para uma aterragem emergente de um avião deve ser feito por obrigação na última pista, caso houver mais aviões podem usar todas as pistas dando prioridade aos de aterragem emergente.
- Cada pista apenas pode ser usada ou para decolagem ou para aterragem, não para decolagem e aterragem ao mesmo tempo.
- Diz-se aterragem emergente, a aterragem de um avião cujo tempo de combustível seja igual a zero, aviões por decolar não precisam, pois pressupõe-se que tenham reabastecido o suficiente para voar até um certo tempo aceitável.
- Existem filas de espera para cada uma das pistas, para uma aterragem normal existem duas filas de espera para cada uma das pistas (as duas primeiras pistas), para a decolagem apenas uma fila de espera para cada uma das pistas.
- Precisa-se de um algoritmo eficiente o suficiente para evitar enchentes nas filas de espera, tanto para aterragem quanto para decolagem.
- A cada instante até 3 aviões podem chegar para as filas de espera de decolagem e até 3 aviões podem chegar para as filas de espera de aterragem.
- Os números de identificação para decolagem são sequências de números pares e para decolagem números ímpares.
- Para maior percepção do que acontece, saídas de resumo descrevendo o processo é uma escolha elegante.
- E no final de tudo a geração de ficheiros contendo o resumo dos processos feitos durante a execução do programa.

2. Implementação

2.1. Classe Avião

No princípio da implementação, pelos dados referidos no enunciado, sabe-se que um Avião é uma parte essencial na existência do aeroporto, visto que são os aviões que mantém a existência e a dinâmica do aeroporto, foi dado atributos para cada Avião, um ID, uma Quantidade de Unidade de Tempo que representa o tempo restante até a reserva do combustível estar vazia, um Tempo de Espera e o Próximo Avião na fila de espera para proximamente ser possível implementar as filas encadeadas.

Para tal foi implementada uma Classe Avião com os atributos citados acima:

- ID – Atributo do Tipo Inteiro
- Unidade de Tempo do Combustível – Atributo do Tipo Inteiro
- Tempo de Espera na Fila – Atributo do Tipo Inteiro
- O Próximo Avião da Fila – Atributo do Tipo Avião

De modos á serem realizados operações com os aviões, foi criado um método para retornar o tempo que um avião esperou desde o momento que entrou na fila até o momento de saída da fila, para tal, inicialmente pensou-se em usar o System.currentTimeMillis() para tal, e realizar uma diferença entre a entrada e a saída, mas usei uma estratégia de modos que seja feita a contagem a cada instante que se mantém na fila, enquanto se manter na fila o seu tempo de espera incrementa.

2.1.1. Método toString()

Para poder ver as informações do Avião, usamos o método toString() que retorna uma String com o id e a unidade de tempo de combustível.

Parâmetro: não retorna nada.

Retorna: retorna uma String contendo a informação do avião.

2.2. Classe Fila

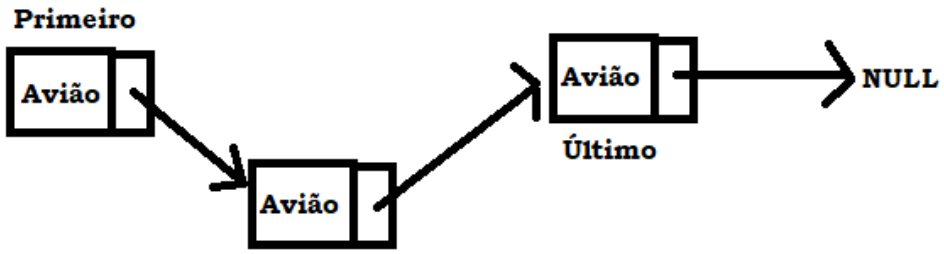
Para a implementação da fila, precisamos simplesmente de três (3) atributos, o primeiro avião da fila, o último avião da fila e o tamanho actual da fila que é incrementado ou decrementado no momento da inserção ou da remoção de qualquer um dos aviões da fila, caso não estiverem vazias.

Para tal foi implementada uma Classe Fila com os atributos citados acima:

- Primeiro Avião da Fila – Atributo do Tipo Avião
- Último Avião da Fila – Atributo do Tipo Avião
- Tamanho da Fila ou Quantidade de elementos da Fila – Atributo do Tipo Inteiro

Logo á seguir, para que a aterragem ou decolagem fosse por ordem de chegada e não por ordem aleatória, embora a aterragem emergente seja um caso particular, que

obrigatoriamente precisa-se retirar de uma posição qualquer de forma emergente, como não se sabe quantos aviões podem chegar para cada uma das Filas, achou-se conveniente implementar uma Fila Dinâmica usando referência de modos á manter o mínimo de memória usada por cada Fila.

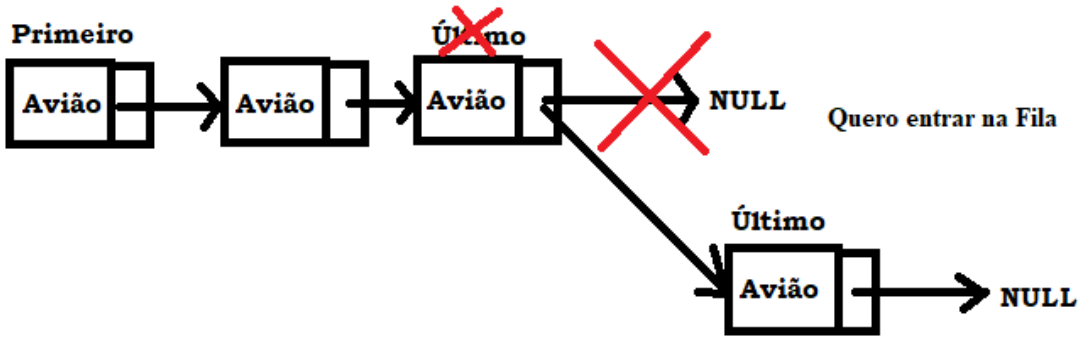


2.2.1. Método de Inserção

Foi implementado um método para a inserção dos aviões na Fila, que consiste em usar referência, com um dos atributos da Classe Avião, o próximo, é possível identificar e definir o próximo avião na fila, sendo que o último elemento da fila sempre aponta para nada (null), como a estrutura de dados fila é uma estrutura que consiste em Inserir no final da fila e remover no início, ao inserir na fila, o actual último avião aponta para o novo avião inserido e o novo avião inserido se torna o novo último, mas em caso de termos uma fila vazia, o novo avião inserido se torna o primeiro e o último avião.

Parâmetro: recebe como parâmetro o avião que deseja inserir na fila.

Retorno: não retorna nada.

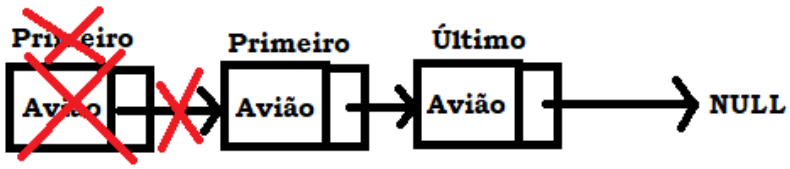


2.2.2. Método de Remoção

Em caso de apenas existir inserção, a fila não teria como ser esvaziada. Para tal, pensou-se que se precisa de um método de remoção de aviões da fila, sendo assim pensado, para poder realizar a remoção é bastante simples, na Estrutura de Dados Fila é removido no início, portanto neste caso tornar o próximo do primeiro no novo primeiro.

Parâmetro: não recebe qualquer parâmetro.

Retorno: retorna o avião removido, caso retornar null é porque a fila está vazia.



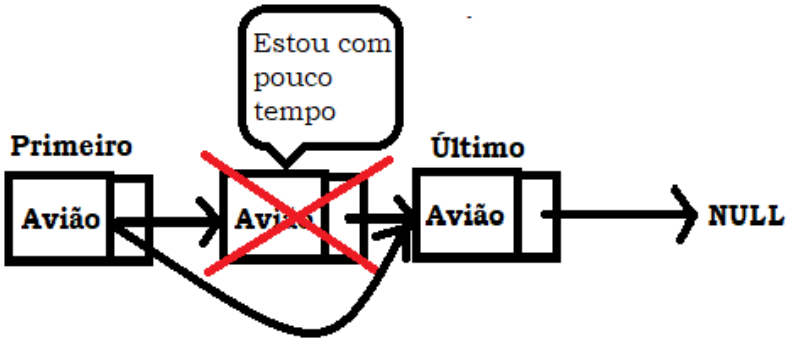
Vou sair da Fila

2.2.3. Método de Remoção Emergente

Como referido anteriormente, existem aviões que têm a Unidade de Tempo do Combustível igual á 1, a estes aviões lhes é emergente sua aterragem, por definição das filas não é permitido retirar nenhum avião de um lugar diferente de início ou fim, mas como é uma emergência, desvia-se logo das regras. Portanto, usei conceitos de lista encadeada de modos á procurar se existe um avião com escassez de Unidade de Tempo de Combustível e se caso existir removê-lo da fila.

Parâmetro: não recebe qualquer parâmetro.

Retorno: retorna o avião removido, caso retornar null é porque não tem nenhuma emergência na fila.

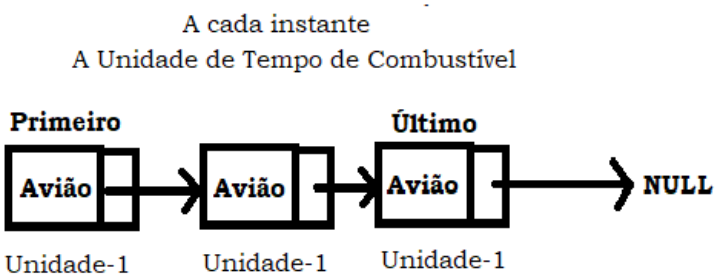


2.2.4. Método de Decremento da Unidade de Tempo de Combustível

Para manter o dinamismo dos instantes decidiu-se criar um método que a cada instante decrementa a Unidade de Tempo de Combustível de todos os aviões na fila. Para tal, pesquisa-se a fila toda, desde o início até chegar ao fim da fila.

Parâmetro: não recebe qualquer parâmetro.

Retorno: não retorna nada.

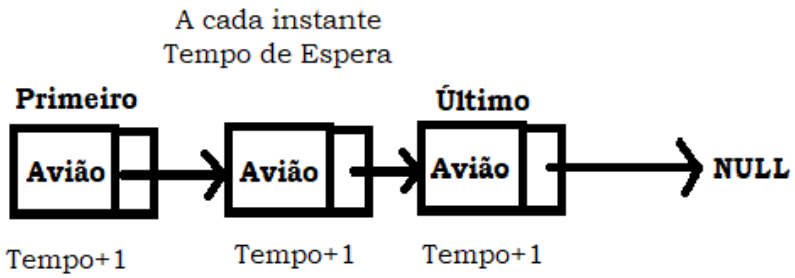


2.2.5. Método de Incremento do Tempo de Espera

Com vista a fazer contagem do Tempo de Espera de cada avião e o dinamismo permanecer, decidiu-se incrementar o Tempo de Espera de cada avião da fila a cada instante, neste caso pesquisar toda fila.

Parâmetro: não recebe qualquer parâmetro.

Retorno: não retorna nada.



2.2.6. Método toString()

Para mostrar o conteúdo de uma fila, criou-se um método para mostrar de forma ordenada as informações de cada Avião existente na fila, chamando o método toString() de cada Avião.

Parâmetro: não recebe qualquer parâmetro.

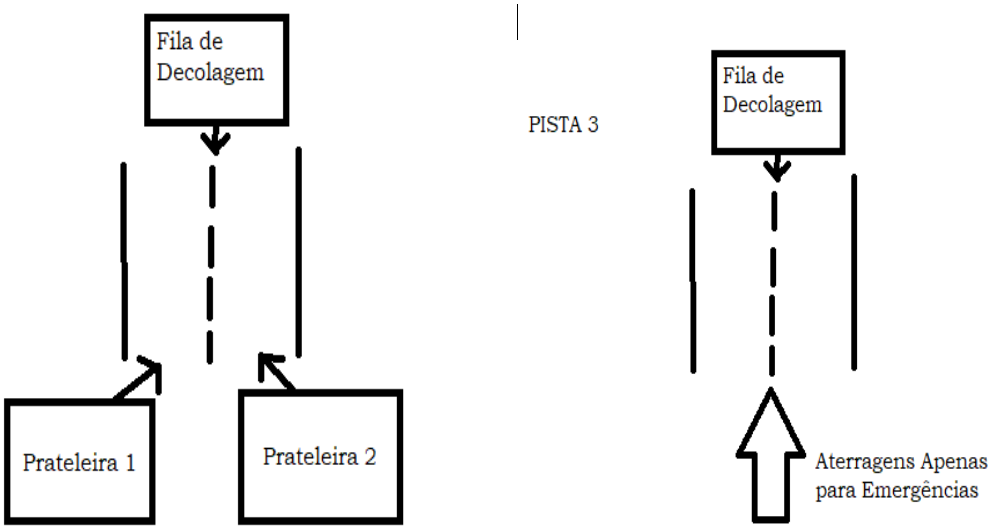
Retorno: retorna uma String contendo os aviões que estão na fila.

2.3. Classe Pista

De modos á manter a gestão correcta de cada uma das pistas, criar uma Classe Pista é a decisão mais correcta á ser feita, sendo que como foi pré-definido que cada uma das pistas terá duas filas de espera para aterragem (prateleiras) e uma fila de espera para decolagem, embora a última fila não precise de prateleiras, apenas precisa de uma fila de espera para decolagem, os atributos são na verdade três (3) Filas.

Para tal foi implementada uma Classe Pista com os atributos citados acima:

- Fila de Espera para Aterragem 1(Prateleira 1) – Atributo do tipo Fila
- Fila de Espera para Aterragem 2(Prateleira 2) – Atributo do tipo Fila
- Fila de Espera para Decolagem – Atributo do tipo Fila



2.3.1. Método de Inserção nas Prateleiras

A partir daqui, surgiu um problema, como decidir em quê prateleira inserir cada avião gerado? Uma alternativa que poderia ajudar na resolução do problema seria deixar a escolha aleatória e achou-se inconveniente, por eu achar que poderia tomar decisões demasiado arriscadas e causar congestionamento nas filas, descartou-se a ideia, surgiu uma outra possível solução, para estar alternando entre uma e outra e manter o mesmo tamanho de forma dinâmica. Decidi que caso a Prateleira 1 não tenha um tamanho maior do que a Prateleira 2, então os aviões vão para a Prateleira 1, no caso contrário vão ser inseridos na Prateleira 2.

Parâmetro: recebe um avião.

Retorno: não retorna nada.

2.3.2. Método de Decremento e Incremento

Para não estar pegando fila por fila na Classe que chamaria a Classe Pista, decidiu-se criar um método para decrementar as Unidades do Tempo de Combustível (Excluindo a fila de espera para decolagem) e incrementar o Tempo de Espera de todos aviões a cada instante, de cada uma das filas da pista sejam elas de decolagem.

Parâmetro: não recebe nada.

Retorno: não retorna nada.

2.3.3. Método de Decisão de Decolagem e Aterragem

Como precisa-se que se mantenha a mesma quantidade de aviões nas filas de espera para cada uma das pistas, agora chega a hora de decolar ou aterrar, numa pista não pode haver

decolagem e aterragem ao mesmo tempo, isso seria um perigo, para a sua resolução, criou-se um método na Classe Pista que tratasse este assunto internamente, já que a aleatoriedade não foi uma opção, como normalmente há muito mais probabilidades de que as prateleiras da pista fiquem sempre com maior número de aviões que as filas de espera para decolagem, portanto, então as prateleiras teriam a prioridade pelo menos até que o tamanho das filas de espera para decolagem ficassem iguais ou maiores que a soma dos tamanhos das prateleiras da pista. Neste caso equilibraria o número de aviões entre as prateleiras e a fila de espera para decolagem.

Parâmetro: não recebe nada.

Retorno: retorna um avião que será removido de uma das filas, se retornar null é porque a pista não tem nenhum avião em nenhuma das filas de espera.

2.3.4. Método de Pesquisa de Emergências

É de facto uma grande dor de cabeça ter de tratar com emergências, mas como na Classe Fila já foi implementado um método que resolve isso, é de grande importância referir que na pista existem duas prateleiras, portanto é necessário que se pesquise nas duas prateleiras se tem ou não um avião que precisa de uma aterragem emergente, chamado o método implementado na fila para cada uma das prateleiras.

Parâmetro: não recebe nada.

Retorno: retorna o avião removido, caso retornar null é porque não tem nenhuma emergência em nenhuma das prateleiras.

2.3.5. Método para Salvar Conteúdo das Filas em Ficheiros

Para que dados sejam guardados de forma eficiente, decidiu-se criar um método para guardar o conteúdo de todas filas da última execução, para que fosse possível visualizá-las mesmo tendo já feito o fecho do programa. Para tal, como forma de manter a organização dos ficheiros de cada fila, cria-se uma pasta referenciando o número da pista contendo os ficheiros com os conteúdos de cada fila, para posteriormente poder ser usado.

Parâmetro: Recebe o número da Pista para criar a pasta.

Retorno: não retorna nada.

2.3.6. Método para Guardar Strings nos Ficheiros

Para poder salvar precisamos saber se o ficheiro já existe ou não, usando o método exists() da Classe File, caso não existir, é criado um novo ficheiro de mesmo nome chamando o método createNewFile() da Classe File, para podermos escrever cada elemento da fila precisamente num ficheiro usando o método toString(), usando a Classe FileWriter para escrever em um ficheiro.

Parâmetro: recebe como parâmetro duas Strings, uma contendo o nome do ficheiro e outra contendo o que se deseja gravar.

Retorno: não retorna nada.

2.3.7. Método toString()

Para que mostrasse o conteúdo de cada fila de espera da pista, foi necessário criar um método que chame o método de cada fila que mostra o conteúdo de todas filas da pista.

Parâmetro: recebe como parâmetro o número da pista.

Retorno: retorna uma String contendo toda informação das filas da pista.

2.4. Classe para Operação em Geral

Todo o aeroporto precisa de uma torre de controle, para que se defina como cada avião deve proceder, para que se definam os aviões em estado emergente, os avisos de chegada e saída, desta feita, uma Classe foi implementada onde seria executado tudo o que foi pedido no enunciado de forma conjunta. Os controles de tráfego também são opções relevantes para o controlo das pistas e a intercalação das aterragens e decolagens, é aqui nesta Classe onde todos processos principais ocorrem.

2.4.1. Método Principal

Como é uma simulação de aterragens e decolagens, tudo foi feito dentro de um loop que a cada 30-35 aviões gerados realiza uma parada indefinida para que o usuário decida se quer continuar a simulação ou não, para que houvesse uma visualização rápida usou-se uma Thread.sleep() com argumento 1000 milissegundos, representando 1 segundo, para adormecer o programa durante um instante e visualizar a execução do programa em tempo real, de modos que a simulação pareça o mais próximo do cenário que se encontra no enunciado. Como disse anteriormente, todas as instruções e decisões que o programa fará estarão dentro deste mesmo loop que terminará apenas quando aparecer a paragem indefinida e o usuário digitar a letra ‘n’, foi uma ideia, até porque no enunciado nunca menciona quando parar a execução, mas surgiu uma ideia que seria bastante engenhosa como foi mencionado.

Serão gerados aleatoriamente um certo número de aviões, este certo número de aviões estará no intervalo entre 0 e 3, que serão distribuídos entre as 3 pistas(2 pistas apenas para as aterragens), neste caso, como temos aviões de decolagem e aterragem, isso implica que no máximo podem ser gerados até 6 aviões no máximo. Para a geração de números aleatórios usei o Math.random().

Na inserção, foi feito algo importante, para evitar as possíveis cheias nas filas de decolagem, como a última fila é a que não é usada para aterragem normal, é usado apenas para emergências de aterragem e decolagens normais, portanto decidi primeiramente preencher a fila de espera para decolagem da pista 3 e evitar preencher de forma irresponsável os aviões nas filas de espera.

Na decolagem e na aterragem, primeiramente antes de tudo precisa verificar se na pista 1 e na pista 2 há algum avião com escassez de Combustível, caso não tiver, faz as escolhas para a aterragem ou decolagem normal, caso não tiver nenhum avião na fila de

espera, é mostrado uma mensagem. Sempre verifica primeiro para a pista 3, a seguir na pista 2 e no final a pista 1. Sob o pretexto de tornar o código mais dinâmico, menor e preciso, decidiu-se criar um Array de Pistas, para que seja possível acessar facilmente, visto que as pistas são bastante limitadas.

Durante cada instante, prateleiras de todas pistas existentes, cada avião das prateleiras, a sua Unidade de Tempo de Combustível é decrementada e para todos os aviões de todas pistas são incrementados o seu Tempo de Espera. E no final de cada instante é mostrado o relatório periódico de cada instante. Caso não desejares continuar, todo resumo será passado para um ficheiro, de formas á visualizar os processos feitos durante a execução do programa.

2.5. Classe Executar

Foi implementada uma Classe contendo o método main que permitirá executar o programa, fazendo a chamada da Classe de Operação Geral.

3. Testes Realizados

```
girozetto@Giro-Zetto:~/Desktop/pfp3$ javac Executar.java
girozetto@Giro-Zetto:~/Desktop/pfp3$ java Executar
Chegaram 0 Aviões ás Filas de Espera de Decolagem
Chegaram 3 Aviões ás Prateleiras de Aterragem
Pista 3 sem nenhum Avião nas Filas de Espera
>>O Avião {ID: 3 } - { Unidade de Tempo: 18 } Aterrou na Pista 2<<
>>O Avião {ID: 5 } - { Unidade de Tempo: 5 } Aterrou na Pista 1<<



+++++Conteúdos das Filas de Cada Pista+++++
--------Pista 1--------
Quantidade de AviÃµes: 1
********Prateleira de Aterragem 1*********
Total de Aviões: 1
{ID: 1 } - { Unidade de Tempo: 11 }
********Prateleira de Aterragem 2*********
Total de Aviões: 0
*******Fila de Espera para Decolagem******
Total de Aviões: 0
-----------------------------------------------
```

Como pode ser verificado na figura acima, bastou apenas compilar a Classe Executar que já compila todas as outras, no início é feito de forma imediata os vôos e como pode ser verificada por terem sido apenas gerados aviões de aterragem, apenas irá preencher as duas primeiras pistas.

```
++++++++++++++++++++++++++++++++++++++++++
Até ao momento Decolaram: 0.0 Aviões
Tempo médio para Decolagem: 0.0 Segundos
Até ao momento Aterraram: 2.0 Aviões
Tempo médio para Aterragem: 0.0 Segundos
Número de Aviões que aterraram sem Reserva de Combustível: 0
```

Apartir daqui vê-se o resumo periódico, na qual foram aterrados 2 aviões. Algo interessante é que como foram aterrados 2 aviões no mesmo instante que entraram na fila, não esperaram nada, portanto seu tempo de espera é zero.

```
Chegaram 1 Aviões ás Filas de Espera de Decolagem
Chegaram 0 Aviões ás Prateleiras de Aterragem
<<O Avião {ID: 6 } - { Unidade de Tempo: -1 } Decolou da Pista 3>>
Pista 2 sem nenhum Avião nas Filas de Espera
>>O Avião {ID: 1 } - { Unidade de Tempo: 10 } Aterrou na Pista 1<<
```

Pode ser observado que os aviões de decolagem têm Unidade de Tempo sempre igual á -1 e seus ID's são números pares por obrigação. Na implementação, foi mencionada uma

possível estratégia de resolver enchentes, pode ser observado tudo agora neste teste. E caso a pista não estiver com algum avião.

```
++++++++++++++++++++++++++++++++++++++++
Até ao momento Decolaram: 1.0 Aviões
Tempo médio para Decolagem: 0.0 Segundos
Até ao momento Aterraram: 3.0 Aviões
Tempo médio para Aterragem: 0.33333334 Segundos
Número de Aviões que aterraram sem Reserva de Combustível: 0
```

Como pode ser verificado, pelo facto de apenas um avião estar esperando um instante até ser aterrado, o tempo de espera incrementa e o tempo médio modifica e os aviões são retirados da fila de espera.

```
Chegaram 2 Aviões ás Filas de Espera de Decolagem
Chegaram 3 Aviões ás Prateleiras de Aterragem
!!!!!!!Emergência: O Avião {ID: 13 } - { Unidade de Tempo: 2 } Aterrou na Pis
ta 3 Sem Reserva de Combustível<<
<<O Avião {ID: 10 } - { Unidade de Tempo: -1 } Decolou da Pista 2>>
>>O Avião {ID: 15 } - { Unidade de Tempo: 6 } Aterrou na Pista 1<<
```

A figura acima mostra, um caso de um um instante em que aterrou emergentemente, decolou e aterrou normalmente, todas em pistas diferentes, isso coloca o programa como sendo bastante eficiente.

```
++++++++++++++++++++++++++++++++++++++++
Até ao momento Decolaram: 10.0 Aviões
Tempo médio para Decolagem: 0.3 Segundos
Até ao momento Aterraram: 7.0 Aviões
Tempo médio para Aterragem: 0.14285715 Segundos
Número de Aviões que aterraram sem Reserva de Combustível: 1
```

As estatísticas começaram a se tornar cada vez mais precisas e começou á haver bastantes mudanças apenas por ter passado uns instantes no programa. E inclusive os aviões com aterragens de emergências também são contados.

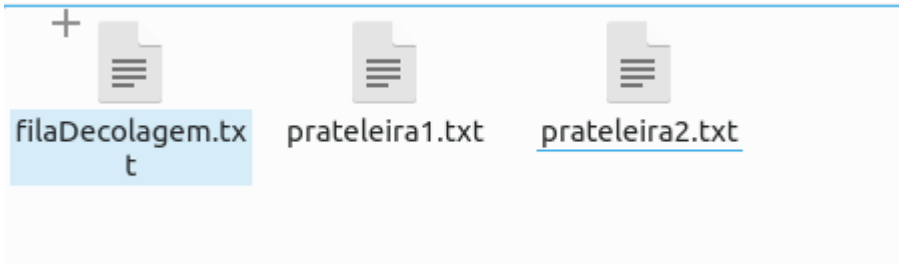
```
++++++++++++++++++++++++++++++++++++++++
Até ao momento Decolaram: 13.0 Aviões
Tempo médio para Decolagem: 0.6923077 Segundos
Até ao momento Aterraram: 13.0 Aviões
Tempo médio para Aterragem: 0.46153846 Segundos
Número de Aviões que aterraram sem Reserva de Combustível: 2



Deseja continuar a Simulação?[s/n]
```

Após ter gerado entre 30 – 35 aviões, é perguntado se deseja continuar a execução da simulação ou fechar, em caso de fechar apenas serão guardadas as informações nos ficheiros.

Como explicado anteriormente, serão criadas estas Pastas e um arquivo, nas pastas contém o ficheiro para cada fila, mas contendo apenas os conteúdos de cada fila. Já este ResumoGeral.txt contém o último resumo mostrado na tela.



Cada um destes arquivos é uma fila e estão dentro de cada pasta Pista, contendo todos aviões que permanecem na fila desde a última execução.

4. Considerações Finais

No decorrer da implementação do projecto em Java, houveram bastantes problemas referentes á remoção de um elemento no meio da Fila ou em qualquer uma outra posição da Fila, o problema não esteve na verdade na implementação do método esteve em se era permitido ou não realizar esta operação, tentei trazer para a vida real, o problema e concluí que a aterragem de emergência é uma exceção á regra. Até porque são aviões que já não podem estar nas prateleiras por muito tempo. Além disso, a resolução do resto das questões foi usar conhecimentos já adquiridos anteriormente e estratégias lógicas e com um alto índice de eficiência, embora seja para simular, mas deu uma grande satisfação terminar o trabalho até porque o aprendizado foi amplo, inclusive me deu o conhecimento de como funciona a gestão de aviões de um aeroporto.