Simulador De Autocarros

Trabalho realizado por Pedro Simões [30007732] e Gonçalo Lemos [30007523].

O projecto foi implementado seguinte o paradigma de programação concorrente com a utilização de threads e timers. Incluí também um extenso uso de expressões lambda e de outros aspectos do paradima funcional.

O projecto foi totalmente implementado á excepção da interface de linha de comandos necessária para o funcionário realizar certas ações. Tal função não foi implementada devido a dificuldade em gerir a CLI, pois a mesma possuí uma constante ocorrência de mensagens a serem impressas durante a execução da simulação.

O projecto foi implementado em primeiro lugar pelo Pedro Simões, com a ajuda concorrente do seu colega, Gonçalo Lemos; ambos os membros possuem um conhecimento total sobre a implementação do mesmo.

Todo o código possuí extensos comentários explicando genericamente o que cada linha está a realizar.

Ficheiro De Configuração:

O programa possui um ficheiro de configuração que é utilizado pelo simulador para defnir os valores de todas as variáveis necessárias para a simulação. Um exemplo do ficheiro de configuração com os valores default:

{

"MINIMUM_FLEET_SIZE": 4, *Representa o valor mínimo da frota total de autocarros; ou seja, é o total mínimo de autocarros necessários para a execução da simulação. Tem como valor default o 4, de acordo com o enunciado.*

"MAXIMUM_FLEET_SIZE": 10, *Representa o valor máximo da frota total de autocarros; ou seja, é o total máximo de autocarros necessários para que a execução da simulação seja possível. Tem como valor default o 10, de acordo com o enunciado.*

"MAX_CAPACITY_MINI_BUS": 24, *Representa a capacidade máxima dos autocarros do tipo 'MINI_BUS'. Tem como valor default o 24, de acordo com o enunciado.*

"MAX_CAPACITY_CONVENCIONAL": 51, *Representa a capacidade máxima dos autocarros do tipo 'CONVENCIONAL'. Tem como valor default o 51, de acordo com o enunciado.*

"MAX_CAPACITY_LONG_DRIVE": 59, *Representa a capacidade máxima dos autocarros do tipo 'LONG_DRIVE'. Tem como valor default o 59, de acordo com o enunciado.*

"MAX_CAPACITY_EXPRESS": 69, *Representa a capacidade máxima dos autocarros do tipo 'EXPRESS'. Tem como valor default o 69, de acordo com o enunciado.*

"BASE_SPEED_MINI_BUS": 5000, *Representa o delay do autocarro do tipo 'MINI_BUS' entre duas paragens. Tem como default o valor 5000 (milisegundos).*

"BASE_SPEED_CONVENCIONAL": 10000, *Representa o delay do autocarro do tipo 'CONVENCIONAL' entre duas paragens. Tem como default o valor 10000 (milisegundos).*

"BASE_SPEED_LONG_DRIVE": 12000, *Representa o delay do autocarro do tipo 'LONG_DRIVE' entre duas paragens. Tem como default o valor 12000 (milisegundos).*

"BASE_SPEED_EXPRESS": 15000, *Representa o delay do autocarro do tipo 'EXPRESS' entre duas paragens. Tem como default o valor 15000 (milisegundos).*

"NUMBER_OF_MINI_BUS": 2, *Define o número de autocarros do tipo 'MINI_BUS' na simulação. Tem de ser pelo menos um.*

"NUMBER_OF_CONVENCIONAL": 4, *Define o número de autocarros do tipo 'CONVENCIONAL' na simulação. Tem de ser pelo menos um.*

"NUMBER_OF_LONG_DRIVE": 3, *Define o número de autocarros do tipo 'LONG_DRIVE' na simulação. Tem de ser pelo menos um.*

"NUMBER_OF_EXPRESS": 1, *Define o número de autocarros do tipo 'EXPRESS' na simulação. Tem de ser pelo menos um.*

"MINIMUM_DELAY_FOR_MALFUNCTION_EXECUTION": 20000, *Indica o limite inferior do intervalo de onde o fixed rate de execução da thread de avaria vai ser escolhido. Tem de ser menor que o limite superior.*

"MAXIMUM_DELAY_FOR_MALFUNCTION_EXECUTION": 30000, *Indica o limite superior do intervalo de onde o fixed rate de execução da thread de avaria vai ser escolhido. Tem de ser maior que o limite inferior.*
"FLAT_TIRE_DELAY_FOR_MALFUNCTION_EXECUTION": 60000, *Define o tempo até a avaria do tipo 'FLAT_TIRE' seja resolvida (em milisegundos).*
"COOLING_SYSTEM_DELAY_FOR_MALFUNCTION_EXECUTION": 30000, *Define o tempo até a avaria do tipo 'COOLING_SYSTEM' seja resolvida (em milisegundos).*
"COLLISION_DELAY_FOR_MALFUNCTION_EXECUTION": 120000, *Define o tempo até a avaria do tipo 'COLLISION' seja resolvida (em milisegundos).*
"MALFUNCTION_PROBABILITY": 0.4 *Define a probabilidade de avaria de um autocarro. Tem de ser maior que zero e menor que 1, inclusive.*
}

BUS_STOP:
Esta enumeração foi implementada com o propósito de organizar e definir uma só classe (neste caso especial) todas as paragens que um autocarro pode executar, pela sua ordem, de sul a norte.

```
public enum BUS_STOP {
    CASCAIS,
    LISBOA,
    COIMBRA,
    PORTO,
    BRAGA
}
```

A ordem de declaração das seguintes constantes, ou seja, das paragens é importante pois se alterada, alterará a rota dos autocarros. A ordem de declaração das mesma corresponde á ordem de seguimento das paragens pelos autocarros se o percurso estiver a ser realizado de sul a norte, caso contrário, o inverso. Caso uma nova paragem seja adicionada, a sua posição em declaração corresponderá á sua posição no novo trajeto dos autocarros, é claro, tal alteração será suportada pela simulação automaticamente.

BUS_TYPE & MALFUNCTION_TYPE
As seguintes enumerações foram implementadas simplesmente pelo propósito de definir os tipos de autocarros e os tipos de avarias, respetivamente.

```
// Enumeration used to represent the diferent types of buses.
public enum BUS_TYPE {
    //The same as the conventional one, allowing the transport of only 24 passengers.
    MINI_BUS,
    //It can carry up to a maximum of 51 passengers.
    CONVENTIONAL,
    //It can carry up to a maximum of 59 passengers.
    // It travels slower when compared to a conventional bus.
    LONG_DRIVE,
    //Same as conventional, but can carry up to a maximum of 69 passengers.
    // It only stops in the cities of Lisbon, Porto and Braga.
    EXPRESS
}
```

```
// Enumeration used to represent the diferent types of malfunctions.
public enum MALFUNCTION_TYPE {
    COLLISION,
    COOLING_SYSTEM,
    FLAT_TIRE
}
```

BUS:

A classe BUS representa o autocarro. É essencialmente uma thread que após construção e execução, executará até que o autocarro, ou seja, o mesmo, chegue ao seu destino. Durante a execução da mesma, ou seja, durante o movimento do autocarro, o mesmo pode sofrer uma avaria, se tal acontecer o autocarro irá parar temporariamente até que a manutenção seja efetuada, usando o método sleep() e uma duração pré-definida de tempo, um delay, em milisegundos. Para evitar que uma exceção de interferência ocorre pois estaria a acontecer uma paragem da thread usando o sleep() devido a uma avaria, durante a execução do sleep() principal que simula o deslocamento do autocarro, fazemos o sleep() da simulação de deslocação um milisegundo de cada vez, usando um 'for' IntStream.range(), de modo a que cada execução desse mesmo for a verificação de avaria possa ocurrer seguramente sem que a excepção aconteça. Caso a mesma aconteça, o que não deve de acordo com a lógica aplicada para impedir-la, um bloco try-catch está implementado para avisar o utilizador que um erro de interrupção de threads aconteceu no momento de simulação da avaria.

A escolha da próxima paragem é realizada com auxílio do método GET_NEXT_STOP() que determina a próxima paragem com atenção á direção do percurso do autocarro (sul ou norte), ao sua origem e destino, e ao facto se é do tipo expresso ou não (pois os autocarros do tipo expresso não param em certas paragens).

A simulação de entrada e saída de passageiros ocorre simplesmente por escolhendo um novo número de ocupantes, maior que zero, com limite na capacidade máxima do autocarro; simulando assim a saida de certos passageiros e a entrada de outros.

SIMULATOR:

A classe SIMULATOR executa o papel de controller da simulação, lendo as variáveis necessárias do ficheiro de configuração (para isso usamos a biblioteca de parser de JSON's: json-simple-1.1.jar), realizando as devidas verificações ás mesma para termos a certezas que podem ser usadas na simulação sem causar problemas; caso, alguma das verificações falhe (ou seja, uma ou mais variáveis estejam mal definidas, o sistema irá terminar e antes disso explicar ao utilizador de uma maneira detalhada o erro de definição das mesma).

Após tais tarefas cria simplesmente os autocarros dos vários tipos, na quantidade definida no ficheiro de configuração, utilizando uma origem e um destino escohidos aleatoriamente, com atenção ao tipo de autocarro (mais uma vez), uma velocidade base entre duas paragens, uma capacidade máxima de passageiros, e claro, um nome ´que defineu unicamente o autocarro com base no seu tipo e index de criação. Em seguida, e para cada autocarro criado, inicia a sua viagem, usando o método start(), e indica a thread main que deve esperar pela execução de cada um deles para terminar (ou seja, assim cumprindo a condição de término do programa: só acaba quando todos os autocarros chegarem ao seu destino), usando o método join(). Após a gestão dos autocarros terminar, cria um timer que engloba uma TimerTask que será executada a um intervalo fixo escolhido aleatoriamente de dentro do intervalo definido no ficheiro de configuração, para a escolha de uma avaria aleatória para aplicar a um autocarro aleatório se a condição probabilística da avaria ocorrer for validada. Se tal acontecer, a avaria escolhida é aplicada ao autocarro escolhido (os dois aleatoriamente). Para terminar, a após a simulação terminar um aviso de término será dado ao utilizador e o Timer responsável pela execução de avarias será cancelado, antes do término do programa.