Enunciado
Trabalho Prático

Algoritmos e Estruturas de Dados
NOVA FCT
Bernardo Toninho

10 Outubro, 2024

Contents

1 Introdução **2**

2 Sistema de Suporte de uma Rede Ferroviária **2**

3 Objectivo **2**

4 Conceitos **2**

5 Especificação do Sistema **3**
5.1 Sintaxe 4
5.2 Tipo dos dados 4
5.3 Sintaxe das Operações a Implementar 4
5.3.1 Inserção de linha 4
5.3.2 Remoção de linha 5
5.3.3 Consulta das estações de uma linha 5
5.3.4 Consulta das linhas de uma estação 5
5.3.5 Inserção de horário 5
5.3.6 Remoção de horário 6
5.3.7 Consulta dos horários de uma linha 6
5.3.8 Comboios por estação 7
5.3.9 Melhor Horário 7
5.3.10 Terminar aplicação 7

6 Desenvolvimento **7**
6.1 Entregas e faseamento 8
6.2 Relatório 8

7 Requisitos do Mooshak **9**

8 Avaliação **9**
8.1 Testes de funcionalidade 9
8.2 Verificação de autoria 9
8.3 Datas importantes 10

Changelog

Actualizações e modificações ao enunciado serão reportadas aqui.

10 Outubro: Publicação Inicial

1 Introdução

Este documento descreve o trabalho prático da disciplina de Algoritmos e Estruturas de Dados do 2º ano da Licenciatura em Engenharia Informática. Os alunos podem e devem tirar todas as dúvidas com os docentes da disciplina.

Trabalho em grupo de 2 alunos a desenvolver respeitando os princípios do Código de Ética do Departamento de Informática. Não são aceites grupos individuais sem autorização explícita de um docente.

Trabalho com 2 fases de entrega. Ver detalhes sobre submissão, datas de entrega e versões possíveis na Secção 6.1.

2 Sistema de Suporte de uma Rede Ferroviária

Nesta secção encontra informação sobre o problema, os conceitos associados, a sintaxe utilizada e a especificação genérica do sistema. Na secção 5.3, cada uma das funcionalidades será descrita na forma de comandos a disponibilizar.

3 Objectivo

O objectivo deste trabalho é o desenvolvimento de um sistema de suporte à gestão de uma rede ferroviária. Uma rede ferroviária é composta por um conjunto de linhas, cada uma contendo uma sequência de estações associadas.

Em traços gerais, pretende-se suportar a criação e remoção de linhas; criação e remoção de horários de comboio; consultar estações e horários de uma dada linha; e pesquisar as linhas de uma dada estação.

4 Conceitos

Cada linha na nossa rede ferroviária contém uma sequência de estações. Por exemplo, a linha de *Cascais* contém a seguinte sequência de estações: Cais do Sodré, Santos, Alcântara, Belém,

Algés, Cruz Quebrada, Caxias, Paço d'Arcos, Santo Amaro, Oeiras, Carcavelos, Parede, São Pedro, São João, Estoril, Monte Estoril e Cascais. As estações Cais do Sodré e Cascais são estações terminais.

A cada linha está associado um conjunto de horários diários. Assume-se que os horários são iguais para todos os dias da semana. Cada horário define a hora em que um comboio chega e parte de cada uma das estações da linha, num dado percurso. Assim sendo, cada horário é definido por um número de comboio, e um percurso que é uma sequência de pares (estação, hora). O sentido do percurso é definido pela estação de partida, que se trata de uma estação terminal. Por exemplo, um horário da linha de Cascais, realizado pelo comboio 19423 com partida na estação do Cais do Sodré (sentido Cais do Sodré - Cascais), é o seguinte:

Nº Comboio: 19423
Serviços a bordo:
Observações: Ⓐ

Estações	
Cais do Sodré	9:45
Santos	-
Alcântara	9:49
Belém	-
Algés	9:54
Cruz Quebrada	-
Caxias	-
Paço de Arcos	-
Santo Amaro	-
Oeiras	10:02
Carcavelos	-
Parede	-
S. Pedro	10:08
S. João	10:11
Estoril	10:13
Monte Estoril	10:15
Cascais	10:17

Notas:

- Num dado percurso, um comboio não pára necessariamente em todas as estações da linha;
- Não existem ultrapassagens entre comboios, ou seja, a ordem de passagem dos comboios, nas estações, é a mesma ordem com que partiram;
- Uma estação pode pertencer a várias linhas simultaneamente.

5 Especificação do Sistema

O funcionamento da aplicação deverá ser o seguinte: depois de arrancar, a aplicação começará a ler comandos da entrada padrão (System.in), processando-os um a um e enviando as respostas para a saída padrão (System.out); a terminação ocorrerá quando for atingido o comando adequado na entrada padrão.

A persistência dos dados deve ser assegurada entre execuções consecutivas. Isto significa que, antes de terminar a execução, o sistema deve guardar o estado do sistema em disco, utilizando as funcionalidades de serialização do Java. Na execução seguinte do programa, os dados armazenados deverão ser carregados do disco e o estado do sistema reconstituído.

5.1 Sintaxe

Pretende-se que a interface da aplicação seja muito simples, de modo a poder ser utilizada em ambientes diversos e, no caso da saída, para permitir automatizar o processo de teste. Por estes motivos, a entrada e a saída deverão respeitar o formato rígido que se indica na secção seguinte. Poderá admitir que a entrada está sintacticamente correcta, mas não deverá assumir que ela está semanticamente correcta.

5.2 Tipo dos dados

Os nomes das linhas e das estações são sequências de caracteres. O número do comboio é um número inteiro. As horas são representadas no formato hh:mm, onde hh é uma sequência de dois dígitos denotando um inteiro entre 0 e 23 (inclusive) e mm é uma sequência de dois dígitos denotando um inteiro entre 0 e 59 (inclusive).

O número de linhas, estações e horários não é limitado superiormente. Poderemos, no entanto, prever um número de 500 linhas, 1000 estações e 10000 horários. É esperado que haja um número proporcional de inserções e remoções no sistema, mas que o número de consultas seja superior ao de inserções e remoções.

Nota: Relativamente à primeira fase, assuma que existem apenas algumas dezenas de linhas, estações e horários.

O sistema não deve fazer distinção entre maiúsculas e minúsculas.

5.3 Sintaxe das Operações a Implementar

De seguida descreve-se em pormenor a sintaxe de cada uma das operações que a aplicação deverá implementar. O símbolo ↲ representa a mudança de linha. Condições de erro são apresentadas pela ordem com que devem aparecer. Por exemplo, para o comando de inserção de horário (Secção 5.3.5), a mensagem "Linha inexistente" será apresentada (e só essa mensagem) caso a linha em consideração não exista, independentemente do horário ser válido ou não.

5.3.1 Inserção de linha

Entrada	`IL nome-linha`↲ `nome-estacao-1`↲ `...` `nome-estacao-n`↲ ↲
Saída (com sucesso)	Inserção de linha com sucesso.↲
Saída (sem sucesso)	Linha existente.↲

Inserção de uma linha, dado um nome e uma lista não vazia de nomes de estações. A situação de erro ocorre caso uma linha com o mesmo nome já exista no sistema.

5.3.2 Remoção de linha

Entrada	`RL nome-linha`↲
Saída (com sucesso)	Remoção de linha com sucesso.↲
Saída (sem sucesso)	Linha inexistente.↲

Remoção de uma linha com o nome dado. Todas as estações da linha que não pertençam a outra linha serão eliminadas. A situação de erro aplica-se quando a linha não existe no sistema.

5.3.3 Consulta das estações de uma linha

Entrada	`CL nome-linha`↲
Saída (com sucesso)	`nome-estacao-1`↲ `...` `nome-estacao-n`↲
Saída (sem sucesso)	Linha inexistente.↲

Lista as estações de uma dada linha, caso a linha exista. As estações devem ser listadas pela ordem dada no momento de inserção da linha em questão.

5.3.4 Consulta das linhas de uma estação

Entrada	`CE nome-estação`↲
Saída (com sucesso)	`nome-linha-1`↲ `...` `nome-linha-n`↲
Saída (sem sucesso)	Estação inexistente.↲

Lista as linhas de uma dada estação, caso esta exista. A listagem deve ser feita por ordem lexicográfica. **Nota: Esta funcionalidade só será avaliada na Fase 2.**

5.3.5 Inserção de horário

Entrada	`IH nome-linha`↲ `número-comboio`↲ `nome-estacao-1 hora-1`↲ `...` `nome-estacao-n hora-n`↲ ↲
Saída (com sucesso)	Criação de horário com sucesso.↲
Saída (sem sucesso)	Linha inexistente.↲ Horário inválido.↲

Quando se insere um horário, a primeira estação indicada (nome-estação-1) tem que ser uma das duas estações terminais da linha em questão, ficando assim definido o sentido do percurso. Assim, é possível definir horários para qualquer uma das duas direcções de uma linha. Um horário tem sempre pelo menos duas estações. É importante notar que um dado horário, que tem de envolver apenas estações da linha dada, não tem de parar em todas as estações da linha. A única restrição neste sentido (para além da noção de validade descrita em seguida) é que a sua primeira estação tem de ser uma das duas estações terminais da linha.

Um horário é **inválido** quando:

Fase 1: as horas do mesmo não são estritamente crescente;

Fase 2: as horas do mesmo não são estritamente crescentes ou **quando se introduz uma situação de ultrapassagem (ou simultaneidade) relativamente a outro horário existente para esse sentido da linha**.

Considerando que um horário é realizado por um comboio concreto, não deve ser possível dois comboios partirem ao mesmo tempo da mesma estação, nem ultrapassarem-se em estações subsequentes. Ou seja, considerando o exemplo inicial do comboio 19423 a partir do Cais do Sodré às 9:45 em direcção a Cascais, podemos ver que ele chega a Alcântara pelas 9:49 e a Oeiras pelas 10:02. Um outro comboio que parta do Cais do Sodré às 9:46, para chegar a Alcântara pelas 9:48 teria de ultrapassar o comboio 19423 e portanto tem um horário **inválido**. De forma semelhante, um comboio que parta do Cais do Sodré pelas 9:40 e chegue a Algés pelas 9:55 teria de ser ultrapassado pelo comboio 19423 (que chega antes a Algés, partindo mais tarde do Cais do Sodré), tornando assim este horário **inválido**. Ou seja, um horário H que parta de uma estação terminal no momento t tem de chegar a todas as suas paragens **antes** de qualquer outro horário que parta da mesma estação terminal num momento t' tal que $t' > t$. Simetricamente, um horário que parta de uma estação terminal no momento t tem de chegar a todas as suas paragens **depois** de qualquer outro horário que parta da mesma estação terminal num momento t'' tal que $t'' < t$.

5.3.6 Remoção de horário

Entrada	`RH nome-linha`↲ `nome-estacao-partida hora-partida`↲
Saída (com sucesso)	Remoção de horário com sucesso.↲
Saída (sem sucesso)	Linha inexistente.↲ Horário inexistente.↲

Remoção de um horário, caso a linha e o horário existam.

5.3.7 Consulta dos horários de uma linha

Entrada	`CH nome-linha`↲ `nome-estacao-partida`↲
Saída (com sucesso)	`número-comboio-1`↲ `nome-estacao-11 hora-11`↲ `...` `nome-estacao-1j hora-1j`↲ `...` `número-comboio-n`↲ `nome-estacao-n1 hora-n1`↲ `...` `nome-estacao-ni hora-chegada-ni`↲
Saída (sem sucesso)	Linha inexistente.↲ Estação de partida inexistente.↲

Lista todos os horários de uma dada linha, caso a linha exista e a estação de partida seja uma estação terminal (determinando o sentido do trajecto). A listagem deverá estar ordenada por hora de partida.

5.3.8 Comboios por estação

Entrada	`LC nome-estação`↲
Saída (com sucesso)	`Comboio número-comboio-1 hora-1`↲ ... `Comboio número-comboio-n hora-n`↲
Saída (sem sucesso)	Estação inexistente.↲

Lista todos os comboios que passam pela estação dada por ordem crescente de horário de partida. **Nota: esta operação só será avaliada na Fase 2.**

5.3.9 Melhor Horário

Entrada	`MH nome-linha`↲ `nome-estacao-partida`↲ `nome-estacao-destino`↲ `hora-chegada-esperada`↲
Saída (com sucesso)	`nome-linha`↲ `nome-estacao-11 hora-11`↲ ... `nome-estacao-1j hora-1j`↲
Saída (sem sucesso)	Linha inexistente.↲ Estação de partida inexistente.↲ Percurso impossível.↲

Determina, caso seja possível, o "melhor" percurso entre duas estações de uma dada linha (de partida e destino). Neste contexto, o melhor percurso é aquele que, **não trocando de comboio**, chega à estação de destino o mais próximo da hora de chegada esperada (mas nunca depois).

5.3.10 Terminar aplicação

Entrada	`TA`↲
Saída	Aplicação terminada.↲

Termina a aplicação.

6 Desenvolvimento

Os estudantes que realizam o trabalho prático têm de estar registados em algum turno prático. O trabalho é realizado em grupos de dois alunos, preferencialmente do mesmo turno prático. Quaisquer desvios a este principio têm de ser autorizados explicitamente por um docente.

O trabalho é implementado em Java 21, havendo duas versões base do trabalho:

- Na versão I completa, avaliada entre 0 e 20 valores, não é permitido o uso do pacote *java.util*, à excepção da classe *Scanner*;
- Na versão J, avaliada entre 0 e 8 valores, podem ser usadas todas as interfaces e classes da biblioteca padrão do Java.

Os estudantes têm a opção de submeter uma versão com algumas interfaces e classes implementadas de acordo com a versão I e outras de acordo com a versão J. Neste caso, os docentes irão atribuir uma nota dependendo do número de recursos à *java.util*, penalizando fortemente estes usos.

6.1 Entregas e faseamento

O trabalho será desenvolvido incrementalmente, em duas fases com entregas já marcadas.

Na primeira fase, os estudantes deverão desenhar a sua solução utilizando os tipos abstractos de dados e respectivas estruturas de dados apresentados até ao final da aula teórica 5 (ver numeração das aulas do Moodle), podendo também usar o TAD Dicionário se acharem adequado (mas não a sua implementação como Tabela de Dispersão). É importante notar que no desenvolvimento do trabalho como um todo, poderá ser adequado implementar variantes das estruturas de dados discutidas nas aulas (mas sem mudar os tipos abstratos de dados), devendo os alunos discutir estas variantes com os docentes da disciplina.

A definição de horário válido mais completa (Secção 5.3.5), o comando de consulta das linhas de uma estação (Secção 5.3.4) e o comando de listagem de comboios por estação (Secção 5.3.8) só serão avaliados na Fase 2.

Todas as operações deverão ser implementadas na 2ª fase. A escolha dos tipos abstratos de dados deve ser revista na 2ª fase para melhorar a performance do trabalho da 1ª para a 2ª fase, tendo em conta os requisitos da secção 5.3. É expectável que alguns dos tipos abstratos de dados usados na solução da 1ª fase sejam alterados na 2ª fase.

O programa deve ser entregue nos concursos Mooshak para o efeito (AED2425_Fase1 e AED2425_Fase2, respectivamente). Na última fase deverá ainda ser entregue através do Moodle um relatório final do trabalho. Cada grupo deve registar-se nos concursos do Mooshak. O nome do utilizador (login no Mooshak) deve ser a concatenação dos números dos alunos que compõem o grupo, separados por _ (o primeiro número deverá ser o menor). Por exemplo, os alunos nº3435 e nº3434 devem ter como nome utilizador no Mooshak 3434_3435. Só serão considerados entregues os trabalhos cujo utilizador no concurso do Mooshak siga estas regras.

A entrega no Mooshak, é constituída por um zip contendo o código fonte devidamente comentado. O nome e número dos alunos que compõem o grupo deverá ser inserido no cabeçalho de todos os ficheiros entregues, da seguinte forma:

```
/**
 * @author NOMEALUNO1 (NUMEROALUNO1) emailALUNO1
 * @author NOMEALUNO2 (NUMEROALUNO2) emailALUNO2
 */
```

6.2 Relatório

Na última fase deverá ser entregue, no Moodle, um relatório final do trabalho, contendo o seguinte:

- Para cada um dos Tipos Abstratos de Dados (Interfaces) definidos no trabalho, uma justificação curta para as implementações realizadas para os mesmos, especificamente para as estruturas de dados escolhidas;
- Para cada uma das operações descritas na secção 5.3, uma descrição do comportamento do trabalho, em termos das operações efetuadas sobre as estruturas de dados;

- O estudo das complexidades temporais das operações descritas na secção 5.3, no melhor caso, pior caso e caso esperado;
- O estudo da complexidade espacial da solução proposta.

Não são definidos templates nem formatos para o relatório. Toda a informação necessária a incluir no relatório está aqui indicada, para além da referência à versão associada ao trabalho (I, J ou mista).

7 Requisitos do Mooshak

Para submeter o trabalho ao Mooshak, é necessário que o código fonte respeite as regras seguintes:

- O código fonte completo tem de ser guardado num arquivo ZIP.
- A classe principal tem de se chamar Main e estar localizada na raiz do arquivo (i.e., tem de pertencer ao pacote default).
- As pastas correspondentes aos pacotes do trabalho têm de estar localizadas na raiz do arquivo.
- O programa só pode criar ficheiros na directoria corrente.

8 Avaliação

O trabalho é obrigatório para todos os alunos que não têm frequência e dá frequência. A avaliação incidirá sobre todos os aspetos: concepção, qualidade e eficiência da solução, modularidade, estrutura e documentação do código, qualidade do relatório final, etc. A 1ª fase vale 15% da nota final. A 2ª fase vale 25% da nota final.

8.1 Testes de funcionalidade

O trabalho terá de satisfazer todos os requisitos especificados na secção 5.3. Essa verificação será efetuada automaticamente pelo sistema Mooshak. Em cada fase:

- Se o programa não passar nenhum dos testes de funcionalidade, os seus autores obterão a nota de 0 na fase em questão;
- Se o programa obtiver o número mínimo de pontos requerido para avaliação da fase, será objeto de avaliação preliminar que deverá ser confirmada no final do semestre, numa discussão ou teste prático.

8.2 Verificação de autoria

No final do semestre, todos os grupos em posição de obter frequência serão submetidos a uma discussão ou teste prático, para verificação de autoria do trabalho. Se um aluno faltar à discussão do trabalho, não obterá frequência.

8.3 Datas importantes

- **1ª Fase do trabalho:** 30 Outubro – 6 Novembro 23:59.
- **2ª Fase do trabalho:** 28 Novembro – 5 Dezembro 23:59.
- **Discussões:** TBD.