FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

Algoritmos e Estruturas de Dados

HomeAway

Ano Letivo 2017/2018
(Versão: 15 de novembro de 2017)

Correções

Secção 3.3.11 – Mensagem de erro para utilizador (viajante) inexistente

Secção 3.3.12 – Descrição da operação continha informação não relevante.

Secção 3.3.8 – Esclarecimento relativo à importância desta operação nos resultados devolvidos pela operação 3.3.13.

Secção 3.3.12 – Clarificação da operação, relativa aos seus resultados e à ordenação dos mesmos.

Secção 3.3.13 - Clarificação da operação, relativa aos seus resultados e à ordenação dos mesmos.

1 Objetivo

O objetivo do trabalho é a implementação de um sistema de aluguer de casas e apartamentos (propriedades) online, permitindo aos utilizadores do mesmo constituírem-se como viajantes e/ou anfitriões relativamente a locais que façam parte do sistema. Um anfitrião é um utilizador que possui uma propriedade para alugar. Um viajante é também um utilizador do sistema que procura viajar para um determinado lugar e que pretende alugar uma propriedade nessa região. O sistema deverá permitir a gestão das propriedades para aluguer associadas aos anfitriões que as possuem. Deve ainda permitir a avaliação das propriedades pelos viajantes, de forma a permitir a ordenação das mesmas, de acordo com a qualidade da estadia, após o término da viagem.

2 Conceitos e Definições

Existe um conjunto de conceitos associados ao sistema a desenvolver e que se descreve a seguir, de acordo com os requisitos do trabalho.

2.1 Utilizadores, viajantes, anfitriões e propriedades

O sistema permite o registo de *Utilizadores* e, para esses utilizadores, a inserção de *Propriedades* que possuam para aluguer. A partir do momento em que regista uma propriedade para aluguer, o utilizador será também um *Anfitrião*. Um utilizador que tenha alugado uma (ou mais) propriedades do sistema constitui ainda um *Viajante*.

O registo de um utilizador corresponde aos seus dados pessoais: nome, nacionalidade, morada, email e número de telefone. Cada utilizador é identificado de forma única por uma cadeia de caracteres (`idUser`).

As propriedades são também identificadas de forma única através de uma cadeia de caracteres (`idHome`). Os dados adicionais associados à propriedade são: descrição, morada, local, preço diário e número de pessoas que podem pernoitar na propriedade ao mesmo tempo. Uma propriedade inserida no sistema tem sempre um proprietário. As propriedades visitadas são passíveis de avaliação pelos viajantes que lá pernoitam.

2.2 Pesquisa e avaliação de propriedades

Após uma estadia numa propriedade, o viajante deve registá-la no sistema, associando-lhe uma avaliação (uma pontuação de, no máximo, 20 pontos). Esta pontuação será acumulada para fins de ordenação (ranking) das propriedades. Um viajante pode pernoitar numa propriedade que lhe pertença, mas, neste caso, não pode avaliar a mesma.

O sistema irá permitir efetuar pesquisas sobre a informação guardada, de forma a poder obter o ranking das melhores propriedades (isto é, com melhor avaliação) num determinado local. Deverá ainda ser possível saber quais as propriedades existentes num local que possam albergar um certo número de pessoas.

3 Especificação do Sistema

O sistema deverá ler comandos da entrada padrão (`System.in`), processando-os um a um e enviando os resultados para a saída padrão (`System.out`). A terminação ocorrerá quando for atingido o fim de ficheiro na entrada padrão, ou quando for executado o comando de finalização de execução do programa. A execução do programa pode ser assegurada com a introdução de dados e respetiva apresentação de resultados em ficheiro, através do redireccionamento do input e do output. Este processo é explicado na página "Recomendações para os testes do Trabalho" que será disponibilizada atempadamente na página da disciplina no Moodle.

O sistema não faz qualquer distinção entre maiúsculas e minúsculas. Também não serão incluídas palavras com acentos ou cedilha. No entanto, o output da informação introduzida deverá ser gerado tal como foi inserido. Isto significa que um utilizador com o nome "Madonna Louise Ciccone" é considerado o mesmo que "MADONNA LOUISE CICCONE". Acrescenta-se que, se o nome foi inserido utilizando a primeira forma ("Madonna Louise Ciccone"), o sistema deverá listá-lo nesta forma, quando gerar o seu output.

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

3.1 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 3.3. Convém referir que o símbolo ↵ representa uma mudança de linha e que **cada comando termina com duas mudanças de linha**.

Poderá admitir que a entrada está sempre sintaticamente correta e que os dados satisfazem as restrições enunciadas na secção 3.2.

3.2 Tipos dos Dados e dos resultados

Esta secção serve para apoiar a compreensão da secção 3.3, onde se descrevem as operações.

O identificador de um utilizador (*idUser*) e de uma propriedade (*idHome*) serão armazenados como sequências de caracteres que não conterão espaços (são palavras - ou tokens de Java). Também o *email* e o número de telefone (*telefone*) do utilizador serão guardados como tokens. O *nome*, *morada* e *nacionalidade* de um utilizador, assim como a descrição (*descricao*), *morada* e *local* de uma propriedade serão também sequências de caracteres apenas terminadas com o carácter de fim de linha. Isto significa que poderão conter mais do que uma palavra. O preço diário (*preco*) e o número de pessoas aceites (*pessoas*) numa propriedade serão todos números inteiros. Podemos considerar, no contexto deste trabalho que o valor de *pessoas* nunca será superior a 20. As avaliações das estadias (*pontos*) inseridas pelos viajantes também serão valores inteiros.

3.3 Operações a implementar

Estas operações serão desenvolvidas incrementalmente. Assim, a descrição de cada uma das operações poderá ser diferente para cada uma das fases do trabalho a implementar, ou não. Em cada uma das subsecções abaixo, cada operação é especificada, de acordo com as diferenças por fase.

3.3.1 Inserir utilizador

- SINTAXE DE ENTRADA

  IU *idUser email telefone nome*↵

  nacionalidade↵

  morada↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Inserção de utilizador no sistema. A operação começa por receber o *idUser*, que deverá identificar o utilizador de forma única no sistema. Seguidamente são inseridos o *email*, *telefone* e *nome* do utilizador. Tanto a *nacionalidade* como a *morada* serão inseridas em linhas separadas. Se já existir um utilizador no sistema com este *idUser*, a operação não será realizada.

  Fase 1: Nesta fase, o sistema contém, no máximo, um utilizador. Não serão realizadas tentativas de inserção de mais do que um utilizador.

  Fases 2 e 3: Nestas fases o sistema poderá conter vários utilizadores.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Insercao de utilizador com sucesso.`↵

↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Utilizador existente.`↵

  ↵

3.3.2 Alterar informação de utilizador

- SINTAXE DE ENTRADA

  UU *idUser email telefone*↵

  morada↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Atualiza a informação associada a um utilizador. Toda a informação contida no comando será atualizada, caso o ***idUser*** já exista no sistema. O nome do utilizador, assim como a nacionalidade, não pode ser alterado.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Utilizador atualizado com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Utilizador inexistente.`↵

  ↵

3.3.3 Remover utilizador

- SINTAXE DE ENTRADA

  RU *idUser*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Esta operação determina a remoção do registo de um utilizador do sistema. Para a operação ter sucesso, ***idUser*** tem de identificar um utilizador que não possua propriedades, existente no sistema.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

`Utilizador removido com sucesso.`↵

↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `mensagem-de-erro-de-remocao-utilizador`↵

  ↵

- A *`mensagem-de-erro-de-remocao-utilizador`* é uma das seguintes:
  - `Utilizador inexistente.`
    Quando o identificador do utilizador não existir no sistema.
  - `Utilizador e proprietario.`
    Quando o utilizador possui propriedades para alugar.

3.3.4 Consultar dados de utilizador

- SINTAXE DE ENTRADA

  `GU` *`idUser`*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Consulta dos dados de utilizador identificado por *`idUser`*. A operação só terá sucesso se *`idUser`* existir no sistema. O resultado da operação irá incluir o *`nome`*, a *`nacionalidade`*, a *`morada`*, o *`email`* e o número de *`telefone`* do utilizador.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `nome`: *`morada, nacionalidade, email, telefone`*↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Utilizador inexistente.`↵

  ↵

3.3.5 Adicionar propriedade

- SINTAXE DE ENTRADA

  `AH` *`idHome idUser preco pessoas local`*↵

  `descricao`↵

  `morada`↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Adicionar uma propriedade, para aluguer, ao sistema. A operação só terá sucesso se *idUser* já existir no sistema (o anfitrião, dono da propriedade) e se *idHome* ainda não existir no sistema. Tanto o *preco* como o número de *pessoas* que podem pernoitar na propriedade são números inteiros positivos.

  Fase 1: Nesta fase, o sistema contém, no máximo, uma propriedade. Não serão realizadas tentativas de inserção de mais do que uma propriedade.

  Fase 2: Nesta fase o sistema poderá conter várias propriedades. No entanto, cada anfitrião terá, no máximo, uma propriedade. Não serão feitas tentativas de inserção de várias propriedades pertencentes ao mesmo anfitrião. Além disso, não serão inseridas várias propriedades no mesmo local.

  Fase 3: Nesta fase não serão impostas limitações às propriedades de um anfitrião ou relativamente aos locais onde se situam as propriedades.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Propriedade adicionada com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-adicao-propriedade↵

  ↵

- A *mensagem-de-erro-de-adicao-propriedade* é uma das seguintes:
  - `Dados invalidos.`
    Quando os valores *preco* ou *pessoas* não forem válidos.
  - `Utilizador inexistente.`
    Quando o identificador do utilizador não existe no sistema.
  - `Propriedade existente.`
    Quando o identificador da propriedade já existe no sistema.

3.3.6 Remover propriedade

- SINTAXE DE ENTRADA

  RH *idHome*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Esta operação determina a remoção do registo de uma propriedade do sistema. Para a operação ter sucesso, *idHome* tem de identificar uma propriedade que não tenha sido visitada.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Propriedade removida com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `mensagem-de-erro-de-remocao-propriedade`↵

  ↵

- A *`mensagem-de-erro-de-remocao-propriedade`* é uma das seguintes:
  - `Propriedade inexistente.`
    Quando o identificador da propriedade não existir no sistema.
  - `Propriedade ja foi visitada.`
    Quando a propriedade já foi visitada.

3.3.7 Consultar dados de propriedade

- SINTAXE DE ENTRADA

  `GH` *`idHome`*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Consulta dos dados de propriedade identificada por *`idHome`*. A operação só terá sucesso se *`idHome`* existir no sistema. O resultado da operação irá incluir a *`descricao`*, a *`morada`*, o *`local`*, o *`preco`*, o número de *`pessoas`* que lá podem pernoitar, o valor acumulado das avaliações realizadas por viajantes (*`pontos`*) e o nome do proprietário (*`nomeProp`*).

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `descricao`: *`morada, local, preco, pessoas, pontos, nomeProp`*↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Propriedade inexistente.`↵

  ↵

3.3.8 Adicionar estadia com avaliação

- SINTAXE DE ENTRADA

  AT *idUser idHome pontos*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Adicionar estadia do viajante ***idUser*** na propriedade ***IdHome*** com a avaliação ***points***. A operação só terá sucesso se ***idUser*** e ***idHome*** já existirem no sistema, se o utilizador ***idUser*** não for o proprietário de ***idHome*** e se ***pontos*** for positivo.

  Fase 1: Nesta fase, o sistema contém, no máximo, uma propriedade e um utilizador. Assim, todas as estadias serão realizadas na propriedade do utilizador. Desta forma, a estadia com avaliação não é possível nesta fase, sendo uma situação de exceção.

  Fases 2 e 3: Nestas fases, o sistema poderá conter vários utilizadores e várias propriedades, não havendo limitações à realização de estadias. É de notar que a inserção de uma estadia poderá alterar a posição da propriedade no ranking das melhores propriedades (ver secção 3.3.13).

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Estadia adicionada com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-estadia-pontos↵

  ↵

- A *mensagem-de-erro-de-estadia-pontos* é uma das seguintes:
  - `Dados invalidos.`
    Quando a avaliação for não positiva.
  - `Utilizador inexistente.`
    Quando o identificador do viajante não existe no sistema.
  - `Propriedade inexistente.`
    Quando o identificador da propriedade não existe no sistema.
  - `Viajante e o proprietario.`
    Quando o viajante é também o proprietário.

3.3.9 Adicionar estadia de proprietário

- SINTAXE DE ENTRADA

  AT *idUser idHome*↵

  ↵

DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

Adicionar estadia do viajante *idUser* na propriedade *IdHome* quando este é o proprietário. Este tipo de estadia é possível mas o utilizador não pode avaliar a propriedade. A operação só terá sucesso se *idUser* e *idHome* já existirem no sistema e se o utilizador *idUser* for o proprietário de *idHome*.

Fase 1: Nesta fase, o sistema contém, no máximo, uma propriedade e um utilizador. Assim, nesta operação, todas as estadias serão realizadas na propriedade do utilizador.

Fase 2: Nesta fase, o sistema poderá conter vários utilizadores e várias propriedades, mas cada anfitrião possui apenas uma propriedade.

Fase 3: Nestas fases, o sistema poderá conter vários utilizadores e várias propriedades, não havendo limitações à realização de estadias.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Estadia adicionada com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-estadia ↵

  ↵

- A *mensagem-de-erro-de-estadia* é uma das seguintes:
  - `Utilizador inexistente.`
    Quando o identificador do viajante não existe no sistema.
  - `Propriedade inexistente.`
    Quando o identificador da propriedade não existe no sistema.
  - `Viajante nao e o proprietario.`
    Quando o viajante não é o proprietário.

3.3.10 Listar propriedades de Anfitrião

- SINTAXE DE ENTRADA

  LH *idUser*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem das propriedades do anfitrião identificado por *idUser*. Esta operação só terá sucesso se *idUser* existir no sistema. A listagem é ordenada por identificador das propriedades.

Fase 1 e 2: Nestas fases, cada anfitrião possui, no máximo, uma propriedade.

Fase 3: Nesta fase não há limite ao número de propriedades que um anfitrião pode possuir.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-propriedades

  mensagem-de-listagem-propriedades

  …

  mensagem-de-listagem-propriedades

  ↵

  Cada *mensagem-de-listagem-propriedades* terá a seguinte forma:

  idHome descricao morada local preco pessoas pontos↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-listagem-propriedades↵

  ↵

- A *mensagem-de-erro-listagem-propriedades* é uma das seguintes:
  - `Utilizador inexistente.`
    Quando o identificador do utilizador não existe no sistema.
  - `Utilizador nao e proprietario.`
    Quando o utilizador não possui propriedades.

3.3.11 Listagem de estadias de um viajante

- SINTAXE DE ENTRADA

  LT *idUser*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem das propriedades visitadas pelo viajante identificado por *idUser*. Esta operação só terá sucesso se *idUser* já existir no sistema e se o utilizador desta forma identificado for, de facto, um viajante, ou seja, se tiver efetuado estadias em propriedades que façam parte do sistema. A listagem é efetuada por ordem da realização das estadias (viagens), da mais recente para a mais antiga. Em qualquer fase do trabalho será possível realizar várias estadias.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-propriedades

  mensagem-de-listagem-propriedades

  …

  mensagem-de-listagem-propriedades

  ↵

  Cada *mensagem-de-listagem-propriedades* terá a seguinte forma:

  idHome descricao morada local preco pessoas pontos↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-listagem-viagens↵

  ↵

- A *mensagem-de-erro-listagem-viagens* é uma das seguintes:
  - `Utilizador inexistente.`
    Quando o identificador do utilizador não existe no sistema.
  - `Utilizador nao viajou.`
    Quando o utilizador não efetuou estadias (viagens).

3.3.12 Pesquisa de propriedades

- SINTAXE DE ENTRADA

  PH *pessoas local*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Pesquisa das propriedades que, num determinado *local*, podem receber **(no mínimo)** um determinado número de *pessoas*. Por exemplo, a pesquisa de propriedades em Sesimbra que possam albergar 6 pessoas, deve devolver informação sobre todas as propriedades que possam receber **pelo menos** 6 pessoas (todas as propriedades que possam albergar entre 6 e 20 pessoas). A listagem resultante desta pesquisa será ordenada pelo número de pessoas (*pessoas*) e dentro desse conjunto, pelo identificador das propriedades(*idHome*). A operação só será realizada se os parâmetros da pesquisa forem valores válidos.

  Fase 1: Nesta fase apenas existirá uma propriedade no sistema.

  Fase 2: Nesta fase, apenas existirá uma propriedade em cada local.

  Fase 3: Nesta fase não existirão limites às propriedades guardadas no sistema.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-pesquisa-propriedades

  mensagem-de-pesquisa-propriedades

  …

  mensagem-de-pesquisa-propriedades

  ↵

  Cada *mensagem-de-pesquisa-propriedades* terá a seguinte forma:

  idHome descricao morada local preco pessoas pontos↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-pesquisa-propriedades↵

  ↵

- A *mensagem-de-erro-de-pesquisa-propriedades* é uma das seguintes:
  - `Dados invalidos.`
    Quando valor *pessoas* não for válido.
  - `Pesquisa nao devolveu resultados.`
    Quando não existirem propriedades para alugar no local que possam albergar o número de pessoas pedido.

3.3.13 Listar melhores propriedades

- SINTAXE DE ENTRADA

  LB *local*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem de todas as propriedades para aluguer existentes num local, ordenadas pelo valor acumulado das avaliações, da propriedade mais valorizada para a menos valorizada. Deverá ser tida em conta a possibilidade de empates nas classificações. Nestes casos, que poderão ser frequentes, dado o número elevado de propriedades e de estadias, os empates deverão ser ordenados pelo identificador da propriedade (*idHome*).

  Fase 1: Nesta fase apenas existirá uma propriedade no sistema.

Fase 2: Nesta fase, apenas existirá uma propriedade em cada local.

Fase 3: Nesta fase não existirão limites às propriedades guardadas no sistema.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-melhores

  mensagem-de-listagem-melhores

  …

  mensagem-de-listagem-melhores

  ↵

  Cada *mensagem-de-listagem-melhores* terá a seguinte forma:

  idHome descricao morada local preco pessoas pontos↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Pesquisa nao devolveu resultados.`↵

  ↵

3.3.14 Terminar execução

- SINTAXE DE ENTRADA

  XS↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Termina a execução do programa, guardando o estado corrente para a próxima execução.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  Gravando e terminando…↵

  ↵

3.4 Números esperados

Relativamente à fase final do trabalho, o sistema deverá ter capacidade para lidar com um grande número de utilizadores e de propriedades, assim como de possíveis viagens (estadias). Cada viajante poderá realizar milhares de viagens ao longo do ano e um anfitrião poderá também possuir milhares de propriedades. Não existindo valores máximos definidos, podemos prever que não deverão existir mais do que dez mil utilizadores no sistema, nem mais do que cinco mil propriedades.

3.5 Requisitos do Mooshak

3.5.1 Regras a Satisfazer pelo Programa

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

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

3.5.2 Como preparar o arquivo ZIP

Para evitar problemas causados pela dimensão do ficheiro submetido, somente o código fonte (ficheiros *.java) deve ser enviado para o servidor.

Assuma, para exemplificar, que o código fonte do trabalho está dentro de uma pasta chamada `/home/me/aulas/AED/trabalhoFinal/src` e que, dentro dessa pasta, há um ficheiro e duas sub-pastas.

```
Main.java
dataStructures
homeAway
```

Nesse caso, o arquivo poderia ser construído (por exemplo, em Linux) com o seguinte comando:

```
zip aed.zip Main.java dataStructures/*.java homeAway/*.java
```

executado na pasta `/home/me/aulas/AED/trabalhoFinal/src`.

4 Desenvolvimento

O trabalho é realizado em grupos de dois alunos e é implementado em Java, nomeadamente em JDK 8, instalado nos laboratórios das aulas práticas, em Windows e Linux.

Há duas versões base do trabalho, tal como descrito abaixo:

- Na **Versão I completa**, avaliada entre 10 e 20 valores, não é permitido fazer uso do *package java.util*, à exceção da classe *Scanner*;
- Na **Versão J mínima**, avaliada entre 10 e 15 valores, podem ser usadas todas as interfaces e classes do Java.

Os estudantes têm também a opção de submeter uma versão com alguns dos interfaces e classes implementados em Versão I, o que lhes permitirá entregar um trabalho que será cotado para um valor máximo entre 15 e 20. Neste caso, durante a avaliação do trabalho,

os docentes irão verificar quais os Tipos Abstratos de Dados que foram implementados sem recurso ao *package java.util*, atribuindo uma nota de acordo com as referências encontradas.

4.1 Entregas e Faseamento

Como já descrito acima, o trabalho será desenvolvido incrementalmente, em três fases diferentes, com entregas marcadas. Em cada fase, todas as operações deverão ser implementadas, mas poderão crescer de complexidade de fase para fase, tal como descrito na secção 3.3.

A entrega no Mooshak, é constituída por um zip contendo o código fonte **devidamente comentado utilizando as regras para geração de javadoc**, tal como descrito na secção 3.5.2. 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
 */
```

4.2 Entrega final

Adicionalmente, na última fase deverá ainda 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 para as implementações realizadas para os mesmos, especificamente para as estruturas de dados escolhidas;
- Para cada uma das operações descritas em 3.3, uma descrição do comportamento do trabalho, em termos das operações efetuadas sobre as estruturas de dados;
- O estudo das complexidades temporais das operações descritas em 3.3, no melhor caso, no pior caso e no caso esperado;
- O estudo da complexidade espacial da solução proposta.

5 Avaliação

O trabalho é obrigatório para todos os alunos que não têm frequência e dá frequência. A avaliação incidirá sobre todos os aspetos: conceção, qualidade e eficiência da solução, modularidade, estrutura e documentação do código, qualidade do relatório final, etc. As fases 1 e 2 valem ambas 5% da nota final da disciplina e a fase 3 vale 15% da mesma nota. Se uma das fases não for entregue dentro do prazo definido, a nota associada à mesma fase será 0 (zero).

5.1 Testes e discussão

O trabalho terá de satisfazer todos os requisitos especificados na secção 3. Essa verificação será efetuada automaticamente pelo sistema Mooshak, com os Testes de Avaliação.

Em cada fase:

- Se o programa não passar todos os Testes de Avaliação, os seus autores obterão a nota de 0 (zero) na fase em questão;
- Se o programa passar todos os Testes de Avaliação da fase, o docente do turno prático dos alunos que constituem o grupo fará uma avaliação preliminar da fase, que deverá ser confirmada no final do semestre, numa discussão final. Os alunos poderão requerer um relatório oral do docente que avaliar o seu trabalho, sobre os aspetos a melhorar na fase seguinte.

5.1.1 Verificação de autoria

O código submetido pelos alunos como fazendo parte do seu trabalho deve ser desenvolvido **de raiz pelos mesmos,** expressamente para o trabalho em questão. As exceções a esta regra serão os interfaces e classes disponibilizados na página Moodle da disciplina. Se por uma razão de força maior, **o trabalho contemplar interfaces ou classes que não foram desenvolvidas de raiz pelos alunos**, **estas deverão ser devidamente referidas no código e no relatório final e poderão condicionar a nota do trabalho.**

Dica: Para iniciar o desenvolvimento do trabalho deve criar-se um projeto vazio no Eclipse, inserindo, conforme as necessidades, os TADs e classes disponíveis na página da disciplina.

No final do semestre, todos os grupos em posição de obter frequência serão submetidos a uma discussão do trabalho, para verificação de autoria. Nesta discussão, os membros do grupo poderão ser questionados sobre **qualquer das classes e interfaces submetidas ao Mooshak em qualquer das fases de submissão**.

Se se detetar:

- Que um trabalho não foi realizado apenas pelos alunos que o assinaram;
- Que um aluno assinou um trabalho que não realizou; ou
- Que a distribuição das tarefas pelos membros do grupo não foi equilibrada,

esse trabalho será anulado e **nenhum** dos elementos do(s) grupo(s) envolvido(s) obterá frequência.

Se um aluno faltar à discussão do trabalho que assinou, não obterá frequência, reprovando à disciplina.

5.2 Resumo das datas importantes

- Submissão da fase 1 do trabalho, no Mooshak: entre 9 e 13 de outubro de 2017. No dia 13 de outubro a entrega eletrónica tem de ser realizada até às **17h**.
- Submissão da fase 2 do trabalho, no Mooshak: entre 6 e 10 de novembro de 2017. No dia 10 de novembro a entrega eletrónica tem de ser realizada até às **17h**.
- Submissão da fase 3 do trabalho e do relatório final, no Mooshak e Moodle: entre 27 de novembro e 3 de dezembro de 2017. No dia 3 de dezembro, a entrega eletrónica tem de ser realizada até às **17h**.
- Marcação das discussões finais: 4 de dezembro de 2017, no Moodle.

- Realização das discussões: entre 5 e 7 de dezembro, **em princípio,** dentro do horário da disciplina (das aulas teóricas ou práticas). Os alunos devem verificar todos os horários possíveis para confirmar a data e hora da discussão.