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

Algoritmos e Estruturas de Dados

FitnessTracker

Ano Letivo 2016/2017

(Versão: 17 de novembro de 2016)

Esclarecimentos adicionados ao enunciado para a 3ª fase:

Secção 3.3.7 – Esclarecimento sobre o desempate na listagem dos treinos de um atleta, quando ordenado por número de calorias

Secção 3.3.13 – Esclarecimento sobre a listagem por ordem alfabética do nome dos atletas pertencentes a um grupo

Secção 3.3.14 e 3.3.15 – Esclarecimento sobre o desempate nas listagens LW e LG.

1 Objetivo

O objetivo do trabalho é a implementação de um sistema gestão de informação relacionada com a atividade física de utilizadores de dispositivos de monitorização de atividade física, ou "*fitness trackers*". Pretende-se que se implementem as funcionalidades típicas de um servidor de um sistema deste tipo, tal como gestão de atletas/trackers e atividades assim com algumas componentes sociais e de "gamification". Desta forma, o sistema permitirá a criação de grupos de atletas que irão contribuir com o seu esforço para evidenciar os resultados do grupo a que pertencem no sistema, num concurso para eleger o grupo com os melhores atletas.

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, trackers, atividades e treinos

O sistema permite o registo de utilizadores (*Atletas*) e, para esses utilizadores, a inserção de *Treinos* dedicados a atividades físicas específicas. O registo de um atleta corresponde aos seus dados pessoais e aos extraídos de um dispositivo de monitorização (tracker) que permite a captura em tempo real dos passos que o utilizador executa ao longo do tempo. Cada atleta é identificado por uma cadeia de caracteres (idTracker) que identifica de forma única o atleta no sistema, assim como o dispositivo que utiliza. O atleta regista ainda no sistema a seguinte informação: nome, peso (em quilogramas), altura (em centímetros) e idade (em anos). Além disso, o registo do atleta irá ainda guardar informações sobre os treinos efetuados pelo mesmo.

Cada treino de um atleta é efetuado no âmbito de uma *atividade* física definida no sistema. Após o treino, o sistema deverá calcular as calorias despendidas pelo atleta durante o treino, graças à equação de Harris-Benedict que se apresenta abaixo:

$$Calorias\ Gastas = (BMR\ /\ 24) \times MET \times T$$

Equação 1 – equação de Harris- Benedict para calcular as calorias gastas por uma determinada atividade

onde

Para os homens: $BMR = (13.75 \times WKG) + (5 \times HC) - (6.76 \times age) + 66$
Para as mulheres: $BMR = (9.56 \times WKG) + (1.85 \times HC) - (4.68 \times age) + 655$

e
BMR = Taxa Metabólica Basal (ao longo de 24 horas)
MET = Equivalente Metabólico (para a atividade selecionada)
T = Duração da atividade (em horas)
HC = Altura (em centímetros)
WKG = Peso (em quilogramas)
age = idade (em anos)

Para cada atividade reconhecida no sistema, precisaremos assim de conhecer uma forma única de a identificar, assim como o seu nome e o seu Equivalente Metabólico (MET) que permitirá o cálculo das calorias despendidas num treino. Os dados a inserir no sistema por cada treino individual irão incluir informação sobre a atividade efetuada, o dispositivo utilizado e o tempo de treino (em horas). Valores exemplo para o MET são: **Andar** (para 4km/h) – 3; **Correr** (no tapete) - 8.

2.2 Dimensão social e concurso

Como motivação para a atividade física, o sistema deverá permitir a criação de grupos de atletas e a organização de um concurso de grupos. Um atleta poderá decidir, em qualquer momento, aderir a um *Grupo* de atletas, mas apenas a um. Esta decisão irá incorporar o total de calorias gastas pelo atleta ao grupo a que aderiu, assim como o número de passos que efetuou. A decisão de sair do grupo irá também subtrair os valores de calorias despendidas e passos efetuados, do grupo. Os grupos existentes no sistema competem para dois troféus anuais:

- Os Caminhantes do Ano: grupo que realizou, no total, mais passos;
- Os Guerreiros do Ano: o grupo que despendeu, no total, mais calorias.

O sistema guarda apenas a informação do ano em decurso.

A criação de um grupo implica o armazenamento do seu identificador único e do seu nome.

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 atleta com o nome "Rodrigo Guedes de Carvalho" é considerado o mesmo que "RODRIGO GUEDES DE CARVALHO". Acrescenta-se que, se o nome foi inserido utilizando a primeira forma ("Rodrigues Guedes de Carvalho"), 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 atleta (`idTracker`), de uma atividade (`idAtividade`), e de um grupo (`idGrupo`) serão armazenados como sequências de caracteres que não conterão espaços (são tokens de Java). O `nome` de um atleta, atividade e grupo serão também sequências de caracteres apenas terminadas com o carácter de fim de linha. O `peso`, `altura` e `idade` de um atleta assim como o `tempo` de uma atividade, o número de `passos` percorridos, o número de `calorias` despendidas e os valores utilizados e calculados (`BMR` e `MET`) serão todos números inteiros. Embora as fórmulas apresentadas na secção 2.1 contenham valores constantes decimais **(sendo mencionados em java com o sufixo f)**, os seus resultados deverão ser inteiros sem arredondamento, para simplificar.

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 atleta

- SINTAXE DE ENTRADA

  IU *idTracker peso altura idade sexo nome*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Inserção de atleta no sistema. A operação começa por receber o ***idTracker***, que deverá identificar o atleta e o seu dispositivo de forma única no sistema. Seguidamente são inseridos o ***peso, a altura,*** a ***idade***, o ***sexo*** e o ***nome*** do utilizador. O sexo do atleta será inserido com o caráter "F" se o atleta for do sexo feminino ou "M" se o atleta for do sexo masculino. Se já existir um atleta no sistema com este ***idTracker***, a operação não será realizada.

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

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

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Insercao de atleta com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-criacao-atleta↵

  ↵

- A *mensagem-de-erro-de-criacao-atleta* é uma das seguintes:
  - `Valores invalidos.`
    Se o peso, altura ou idade forem não positivos ou se o caráter que representa o sexo do atleta não for válido.
  - `Atleta existente.`
    Quando o identificador do atleta já existe no sistema.

3.3.2 Alterar informação de atleta

- SINTAXE DE ENTRADA

  UU *idTracker peso altura idade*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

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

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Atleta atualizado com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-atualizacao-atleta↵

  ↵

- A *mensagem-de-erro-de-atualizacao-atleta* é uma das seguintes:
  - `Valores invalidos.`
    Se o peso, altura ou idade forem não positivos.
  - `Atleta inexistente.`
    Quando o identificador do atleta não existe no sistema.

3.3.3 Remover atleta

- SINTAXE DE ENTRADA

  RU *idTracker*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Esta operação determina a remoção de toda a evidência relativa ao atleta identificado por *idTracker* do sistema, inclusive no que respeita aos dados dos grupos e à ordenação do concurso. A remoção de um atleta, implica a atualização dos valores relativos ao grupo a que o atleta pertence, se este pertencer a algum grupo à data de remoção. Para a operação ter sucesso, *idTracker* tem de identificar um atleta existente no sistema.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Atleta removido com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Atleta inexistente.`↵

  ↵

3.3.4 Consultar dados de atleta

- SINTAXE DE ENTRADA

  `CU` *`idTracker`*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Consulta dos dados de atleta identificado por *`idTracker`*. A operação só terá sucesso se *`idTracker`* existir no sistema. O resultado da operação irá incluir o *`nome`*, o *`sexo`*, o *`peso`*, a *`idade`* e o total de *`calorias`* e *`passos`* realizados pelo atleta até ao momento.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `mensagem-de-consulta-de-atleta`↵

  ↵

- A *`mensagem-de-consulta-de-atleta`* é uma das seguintes:
  - *`nome`*`: `*`sexo`*`, `*`peso`*` kg, `*`idade`*` anos, `*`calorias`*` cal, `*`passos`*` ps (`*`grupo`*`)`
    Quando o atleta pertencer a um grupo. Neste caso *`grupo`* será o nome do grupo a que o atleta pertence. O sexo deve ser apresentado com as palavras "`Feminino`" ou "`Masculino`".
  - *`nome`*`: `*`sexo`*`, `*`peso`*` kg, `*`idade`*` anos, `*`calorias`*` cal, `*`passos`*` ps`
    Quando o atleta não pertencer a um grupo. O sexo deve ser apresentado com as palavras "`Feminino`" ou "`Masculino`".
- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Atleta inexistente.`↵

  ↵

3.3.5 Criar Atividade

- SINTAXE DE ENTRADA

  `IA` *`idAtividade MET nome`*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Criação de tipo de atividade identificada por ***idAtividade***. A operação só terá sucesso se ***idAtividade*** ainda não existir no sistema e se ***MET*** for positivo. A inserção da atividade com sucesso permitirá que os atletas possam, posteriormente, executar treinos associados à mesma. O ***MET*** da atividade irá contribuir para o cálculo das calorias do treino, de acordo com a Equação 1.

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

  Fases 2 e 3: Nestas fases o sistema poderá conter vários tipos de atividade.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Atividade criada com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `mensagem-de-erro-de-criacao-atividade`↵

  ↵

- A *`mensagem-de-erro-de-criacao-atividade`* é uma das seguintes:
  - `MET invalido.`
    Quando o valor do MET for não positivo.
  - `Atividade existente.`
    Quando o identificador da atividade já existe no sistema.

3.3.6 Adicionar treino

- SINTAXE DE ENTRADA

  `AW` *`idTracker idAtividade duracao`*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Adicionar treino da atividade ***idAtividade*** ao atleta identificado por ***idTracker***, com ***duracao*** em número de horas dedicada à mesma. A operação só terá sucesso se ***idTracker*** e ***idAtividade*** já existirem no sistema e se ***duracao*** for positivo. Um atleta pode realizar vários treinos, em qualquer fase do trabalho.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Treino adicionado com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-treino↵

  ↵

- A *mensagem-de-erro-de-treino* é uma das seguintes:
  - `Tempo invalido.`
    Quando o tempo de realização da atividade for não positivo.
  - `Atleta inexistente.`
    Quando o identificador do atleta não existe no sistema.
  - `Atividade inexistente.`
    Quando o identificador da atividade não existe no sistema.

3.3.7 Consultar treinos de atleta

- SINTAXE DE ENTRADA

  CW *idTracker tipo*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem dos treinos do atleta identificado por *idTracker*. Esta operação só terá sucesso se *idTracker* existir no sistema. A listagem dos treinos pode ser ordenada de duas formas:

  - Se o carácter em tipo for `'T'` a listagem será ordenada por ordem cronológica da realização do treino, iniciando no treino mais recente;
  - Se o carácter em tipo for `'C'`, a listagem será ordenada decrescentemente pelo total de calorias despendidas no treino. A circunstância possível de que um atleta tenha executado mais do que um treino com o mesmo número de calorias deve ser considerada. Os treinos com o mesmo número de calorias deverão ser listados por ordem cronológica da realização do treino, iniciando no treino mais recente.

  Fases 1 e 2: Nestas fases, apenas serão executadas listagens por ordem cronológica.

  Fase 3: Nesta fase ambas as opções de listagem serão possíveis.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-treino

  mensagem-de-listagem-treino

  …

  mensagem-de-listagem-treino

  ↵

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

  NomeAtividade calorias cal↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-listagem-treino↵

  ↵

- A *mensagem-de-erro-listagem-treino* é uma das seguintes:
  - `Opcao invalida.`
    Quando a opção de listagem não é válida.
  - `Atleta inexistente.`
    Quando o identificador do atleta não existe no sistema.
  - `Atleta sem treinos.`
    Quando o atleta não realizou nenhum treino.

3.3.8 Atualizar passos

- SINTAXE DE ENTRADA

  AS *idTracker passos*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Atualização dos passos realizados pelo atleta identificado por ***idTracker***. A operação só terá sucesso se ***idTracker*** já existe no sistema e se ***passos*** for positivo. Os valor em ***passos*** será adicionado ao valor corrente.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Atualizacao de passos com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-atualizacao-passos↵

  ↵

- A *mensagem-de-atualizacao-passos* é uma das seguintes:
  - `Numero de passos invalido.`
    Quando o número de passos é não positivo.
  - `Atleta inexistente.`
    Quando o identificador do atleta não existe no sistema.

3.3.9 Criar grupo de atletas

- SINTAXE DE ENTRADA

  IG *idGrupo nome*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Criação de grupo de atletas identificado por ***idGrupo*** e com nome ***nome***. A operação só terá sucesso se ***idGrupo*** ainda não existir no sistema. Após a criação do grupo, qualquer atleta que não pertença já a um grupo poderá aderir ao novo grupo.

  Fase 1 e 2: Nestas fases, o sistema contém, no máximo, um grupo. Não serão realizadas tentativas de inserção de mais do que um grupo.

  Fase 3: Nesta fase o sistema poderá conter vários grupos.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Grupo criado com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Grupo existente.`↵

  ↵

3.3.10 Adesão de atleta a grupo

- SINTAXE DE ENTRADA

  AG *idTracker idGrupo*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Adesão do atleta identificado por ***idTracker*** ao grupo identificado por ***idGrupo***. Esta operação só terá sucesso se ***idTracker*** e ***idGrupo*** já existirem no sistema e se o atleta ainda não pertencer a nenhum grupo. A adesão com sucesso implicará que os totais de passos e calorias do atleta serão adicionados aos totais do grupo, o que terá também implicações na posição do grupo no concurso.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  ```
  Adesao realizada com sucesso.↵
  ```

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-adesao↵

  ↵

- A *mensagem-de-erro-de-adesao* é uma das seguintes:
  - `Atleta inexistente.`
    Quando o identificador do atleta não existe no sistema.
  - `Grupo inexistente.`
    Quando o identificador do grupo não existe no sistema.
  - `Atleta ja tem grupo.`
    Quando atleta já pertence a um grupo.

3.3.11 Desistência de grupo

- SINTAXE DE ENTRADA

  DG *idTracker idGrupo*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Desistência do atleta identificado por ***idTracker*** do grupo identificado por ***idGrupo***. Esta operação só terá sucesso se ***idTracker*** e ***idGrupo*** já existirem no sistema e se o atleta ainda pertencer ao grupo referido. A desistência com sucesso implicará que os totais de passos e calorias do atleta serão subtraídos aos totais do grupo, o que terá também implicações na posição do grupo no concurso.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  ```
  Desistencia realizada com sucesso.↵
  ```

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-desistencia↵

  ↵

- A *mensagem-de-erro-de-desistencia* é uma das seguintes:
  - `Atleta inexistente.`
    Quando o identificador do atleta não existe no sistema.
  - `Grupo inexistente.`
    Quando o identificador do grupo não existe no sistema.
  - `Atleta nao pertence ao grupo.`
    Quando atleta não pertence ao grupo referido (pertence a outro grupo ou não pertence a nenhum).

3.3.12 Consulta de grupo

- SINTAXE DE ENTRADA

  CG *idGrupo*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Consulta dos resultados do grupo identificado por *idGrupo* no âmbito do concurso. Esta operação só terá sucesso se *idGrupo* já existir no sistema e apresenta os totais atuais de passos e calorias do grupo.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Grupo` *nome*`:` *calorias* `cal,` *passos* `ps`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Grupo inexistente.`↵

  ↵

3.3.13 Listar Grupo

- SINTAXE DE ENTRADA

  LG *idGrupo*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem dos atletas que aderiram ao grupo identificado por *idGrupo*. Esta listagem será realizada por ordem alfabética do nome do atleta e só será realizada se *idGrupo* existir no sistema. Pode assumir que os nomes dos atletas são únicos.

  Fases 1 e 2: Nestas fases, esta listagem não será executada.

  Fase 3: Nesta fase poderão ser submetido pedidos para esta listagem.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-atletas

  mensagem-de-listagem-atletas

  …

  mensagem-de-listagem-atletas

  ↵

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

  nome: *sexo*, *peso* kg, *idade* anos, *calorias* cal, *passos* ps↵
  O sexo deve ser apresentado com as palavras "Feminino" ou "Masculino".

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-listagem-atletas↵

  ↵

- A *mensagem-de-erro-de-listagem-atletas* é uma das seguintes:
  - Grupo inexistente.
    Quando o grupo referido não existe no sistema.
  - Grupo nao tem adesoes.
    Quando o grupo não tem atletas.

3.3.14 Listar Caminhantes

- SINTAXE DE ENTRADA

  LC↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem da ordenação atual dos grupos para o concurso Caminhantes do ano. A ordenação do concurso Caminhantes do ano é feita por ordem decrescente do número de passos acumulados do grupo. A circunstância possível de que vários

grupos possam estar empatados na ordenação deve ser considerada. Os grupos empatados deverão ser listados por ordem cronológica da obtenção do número de passos, do mais antigo para o mais recente.

Fases 1 e 2: Nestas fases, existe apenas um grupo no sistema.

Fase 3: Nesta fase, não existem limitações ao número de grupos.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-caminhantes

  mensagem-de-listagem-caminhantes

  …

  mensagem-de-listagem-caminhantes

  ↵

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

  Grupo *nome*: *calorias* cal, *passos* ps↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  Nao existem grupos.↵

  ↵

3.3.15 Listar Guerreiros

- SINTAXE DE ENTRADA

  LW↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem da ordenação atual dos grupos para o concurso Guerreiros do ano. A ordenação do concurso Guerreiros do ano é feito por ordem decrescente do número de calorias despendidas acumuladas pelo grupo. A circunstância possível de que vários grupos possam estar empatados na ordenação deve ser considerada. Os grupos empatados deverão ser listados por ordem cronológica da obtenção do número de calorias, do mais antigo para o mais recente.

  Fases 1 e 2: Nestas fases, existe apenas um grupo no sistema.

  Fase 3: Nesta fase, não existem limitações ao número de grupos.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-guerreiros

  mensagem-de-listagem-guerreiros

  …

  mensagem-de-listagem-guerreiros

  ↵

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

  Grupo *nome*: *calorias* cal, *passos* ps↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  Nao existem grupos.↵

  ↵

3.3.16 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 atletas e de grupos, assim como de possíveis atividades. Cada atleta poderá realizar milhares de treinos ao longo do ano e vários no mesmo dia. Não existindo valores máximos definidos, podemos prever que não deverão existir mais do que seis mil atletas no sistema, nem mais do que três mil grupos.

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
fitnessTracker
```

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

```
zip aed.zip Main.java dataStructures/*.java fitnessTracker/*.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 10 e 14 de outubro de 2016. No dia 14 de outubro a entrega eletrónica tem de ser realizada até às **17h**.
- Submissão da fase 2 do trabalho, no Mooshak: entre 7 e 11 de novembro de 2016. No dia 11 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 28 de novembro e 2 de dezembro de 2016. No dia 2 de dezembro, a entrega eletrónica tem de ser realizada até às **17h**.
- Marcação das discussões finais: 5 de dezembro de 2016, até às 12h, no Moodle.
- Realização das discussões: entre 6 e 9 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.