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

Algoritmos e Estruturas de Dados
BikePickUp

Ano Letivo 2018/2019
(Versão: 24 de setembro de 2018)

1 Objetivo

O objetivo do trabalho é a implementação de um sistema de partilha gratuita de bicicletas para a Cidade da Costa de Caparica. O sistema chama-se BikePickUp e cada utilizador registado pode levar consigo (fazer *pickUp*) uma bicicleta que esteja estacionada num dos parques do sistema, desde que tenha feito um depósito de, no mínimo, 5 euros, no sistema. Após o pickup, o utilizador tem direito a movimentar-se gratuitamente na cidade, devendo fazer a entrega da bicicleta no prazo máximo de 60 minutos. Se esse prazo for ultrapassado, o sistema irá retirar-lhe 1 euro do saldo, por cada período menor ou igual a 30 minutos de atraso. Por exemplo, um atraso de 17 minutos será pago com 1 Euro e um atraso de 61 minutos valerá 3 euros. Os utilizadores são também classificados (ordenados) com base no número de atrasos que tiveram na entrega das bicicletas ao sistema. Assim, por cada atraso na entrega, os utilizadores ganham um ponto.

O sistema deverá permitir a gestão das contas de utilizador, bicicletas e parques do sistema. Deverá ainda ser possível a identificação dos utilizadores que mais se atrasam na entrega das bicicletas assim como dos parques de bicicletas mais populares (mais movimentados).

2 Conceitos e Definições

Os conceitos associados ao sistema a desenvolver descrevem-se a seguir, de acordo com os requisitos do trabalho.

2.1 Utilizadores, bicicletas e parques

O sistema permite o registo de *Utilizadores* do sistema, habitantes ou visitantes da cidade que pretendem movimentar-se na mesma com as bicicletas do sistema. O registo de um utilizador corresponde aos seus dados pessoais: NIF, nome, morada, email e número de telefone. Cada utilizador é identificado de forma única por uma cadeia de caracteres (`idUser`). Será ainda necessário gerir o saldo do utilizador, assim como a pontuação relativa aos atrasos nas entregas das bicicletas. É sempre possível listar todos os pickups efetuados pelo utilizador.

As *Bicicletas* são também identificadas de forma única através de uma cadeia de caracteres (`idBike`). Além disso, o registo da bicicleta deverá incluir a matrícula. Num determinado momento, cada bicicleta estará estacionada num dos parques do sistema, ou em movimento. É sempre possível listar todos os pickups de que a bicicleta foi alvo.

Os parques do sistema são também identificados de forma única através de uma cadeia de caracteres (`idPark`). Além do identificador, os dados relativos ao parque de bicicletas irão incluir: nome e morada. Em qualquer momento, deve ainda ser possível saber quais as bicicletas estacionadas num determinado parque.

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 "Joaquim Fontes" é considerado o mesmo que "JOAQUIM FONTES". Acrescenta-se que, se o nome foi inserido utilizando a primeira forma ("Joaquim Fontes"), 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`), de uma bicicleta (`idBike`) e de um parque (`idPark`) 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 assim como a matrícula da bicicleta (`matricula`) serão guardados como tokens.

Dados de `nome` e `morada` 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 saldo do utilizador, assim como os pontos atribuídos pelos atrasos serão números inteiros não negativos.

3.3 Operações a implementar e diferenças por fase

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.

Para simplificar descrevem-se aqui, de forma sumária, as restrições aplicadas ao sistema em cada fase:

- Fase 1: No sistema, existem apenas 1 utilizador, 1 bicicleta e um parque. Não existem restrições ao número de pickups que um utilizador pode efetuar. Não existem restrições ao número de pickups de que uma bicicleta pode ser alvo;
- Fase 2: Existem vários utilizadores e várias bicicletas, mas existe apenas um parque. Não há atrasos na entrega das bicicletas.
- Fase 3: Não há restrições nos atrasos ou no número de parques.

3.3.1 Inserir utilizador

- SINTAXE DE ENTRADA

  AddUser *idUser nif email telefone nome*↵

  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 ***nif***, ***email***, ***telefone*** e ***nome*** do utilizador. A ***morada*** será inserida numa linha separada. Se já existir um utilizador no sistema com este ***idUser***, a operação não será realizada. Não há necessidade de verificação do ***nif***. O utilizador é criado com 5 euros de saldo atribuídos pelo sistema de forma gratuita.

  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 Remover utilizador

- SINTAXE DE ENTRADA

  `RemoveUser` *`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 tenha ainda realizado pickups.

- 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 ja utilizou o sistema.`
    Quando o utilizador já efetuou pickups.

3.3.3 Consultar dados de utilizador

- SINTAXE DE ENTRADA

  `GetUserInfo` *`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 *`nif`*, *`nome`*, a *`morada`*, o *`email`*, o número de *`telefone`* do utilizador assim como o seu *`saldo`* e o número de *`pontos`* obtidos devido a atrasos na entrega de bicicletas.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  nome: *nif, morada, email, telefone, saldo, pontos*↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Utilizador inexistente.`↵

  ↵

3.3.4 Inserir parque

- SINTAXE DE ENTRADA

  AddPark *idPark nome*↵

  morada↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Adicionar um parque de bicicletas do sistema, com respetivo ***nome*** e ***morada***. A ***morada*** será inserida numa linha separada. A operação só terá sucesso se ***idPark*** ainda não existir no sistema.

  Fases 1 e 2: Nesta fase, o sistema contém, no máximo, um parque de bicicleta. Não serão realizadas tentativas de inserção de mais do que um parque.

  Fase 3: Nesta fase o sistema poderá gerir vários parques de bicicletas.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Parque adicionado com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Parque existente.`↵

  ↵

3.3.5 Inserir bicicleta

- SINTAXE DE ENTRADA

  AddBike *idBike idPark matricula*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Adicionar uma bicicleta para partilha no sistema. A bicicleta é adicionada ao sistema já localizada num parque de bicicletas do sistema. A operação só terá sucesso se *idBike* ainda não existir no sistema e se *idPark* já existir no sistema. A matrícula não será verificada.

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

  Fase 2: Nesta fase o sistema poderá conter várias bicicletas, mas apenas um parque.

  Fase 3: Nesta fase todas as bicicletas do sistema poderão ser movimentadas pelos vários parques.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Bicicleta adicionada com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-insercao-bicicleta↵

  ↵

- A *mensagem-de-erro-de-insercao-bicicleta* é uma das seguintes:
  - `Bicicleta existente.`
    Quando o identificador da bicicleta já existir no sistema.
  - `Parque inexistente.`
    Quando o identificador do parque não existir no sistema.

3.3.6 Remover bicicleta

- SINTAXE DE ENTRADA

  RemoveBike *idBike*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Esta operação determina a remoção do registo de uma bicicleta do sistema. Para a operação ter sucesso, *idBike* tem de identificar uma bicicleta que não tenha sido utilizada (ou seja que não tenha sido alvo de pickup).

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Bicicleta removida com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `mensagem-de-erro-de-remocao-bicicleta`↵

  ↵

- A *`mensagem-de-erro-de-remocao-bicicleta`* é uma das seguintes:
  - `Bicicleta inexistente.`
    Quando o identificador da bicicleta não existir no sistema.
  - `Bicicleta ja foi utilizada.`
    Quando a bicicleta já foi alvo de pickups.

3.3.7 Consultar dados de Parque de bicicletas

- SINTAXE DE ENTRADA

  `GetParkInfo` *`idPark`*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Consulta dos dados de parque de bicicletas identificado por *`idPark`*. A operação só terá sucesso se *`idPark`* existir no sistema. O resultado da operação irá incluir o *`nome`*, a *`morada`* e o número de *`bicicletas`* estacionadas no parque no momento da consulta.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `nome`: *`morada, bicicleta`*↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Parque inexistente.`↵

  ↵

3.3.8 Iniciar Pickup

- SINTAXE DE ENTRADA

  `PickUp` *`idBike idUser`*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Este comando inicia a partilha de bicicleta ***IdBike*** pelo utilizador ***idUser***. A operação só terá sucesso se ***idBike*** e ***idUser*** já existirem no sistema, se a bicicleta estiver estacionada em algum parque de bicicletas do sistema e se o saldo do utilizador for igual ou superior a 5 euros.

  Fase 1: Nesta fase, o sistema contém, no máximo, uma bicicleta, um utilizador e um parque.

  Fase 2: Nesta fase, o sistema poderá conter vários utilizadores e várias bicicletas, mas apenas um parque.

  Fase 3: Nesta fase não existem limitações ao número de utilizadores, parques e bicicletas.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `PickUp com sucesso.`↵

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `mensagem-de-erro-de-pickup`↵

  ↵

- A *`mensagem-de-erro-de-pickup`* é uma das seguintes:
  - `Bicicleta inexistente.`
    Quando o identificador da bicicleta não existe no sistema.
  - `Bicicleta em movimento.`
    Quando a bicicleta não está estacionada.
  - `Utilizador inexistente.`
    Quando o identificador do utilizador não existe no sistema.
  - `Saldo insuficiente.`
    Quando o saldo atual do utilizador não atinge os 5 euros.

3.3.9 Entrega de bicicleta em parque

- SINTAXE DE ENTRADA

  `PickDown` *`idBike idPark minutos`*↵

  ↵

  DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Entrega da bicicleta ***idBike*** no parque ***Idpark***, depois de ***minutos*** minutos de passeio. A operação só terá sucesso se ***idBike*** e ***idPark*** já existirem no sistema, se a bicicleta não estiver estacionada no momento do pickdown, e se a duração do

pickup for válida (positiva). Todo o processo é gratuito se a bicicleta for entregue sem atrasos. Só no caso de existirem atrasos haverá lugar a pagamento e alteração do saldo e dos pontos associados ao utilizador.

A mensagem de saída da operação efetuada com sucesso deverá incluir o saldo atualizado do utilizador (*saldo*), e os pontos (*pontos*) acumulados relativos às entregas com atraso (ver abaixo).

Fase 1: Nesta fase, o sistema contém, no máximo, uma bicicleta e um parque, assim como um utilizador.

Fase 2: Nesta fase, o sistema poderá conter vários utilizadores e várias bicicletas, mas apenas um parque. Não serão efetuadas tentativas de entregas em atraso.

Fase 3: Nesta fase, não existem limitações ao número de bicicletas, utilizadores ou parques e poderão acontecer entregas com atraso.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Pickdown com sucesso: saldo euros, pontos pontos↵`

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-pickdown ↵

  ↵

- A *mensagem-de-erro-de-pickdown* é uma das seguintes:
  - `Bicicleta inexistente.`
    Quando o identificador da bicicleta não existe no sistema.
  - `Bicicleta parada.`
    Quando a bicicleta está estacionada.
  - `Parque inexistente.`
    Quando o identificador do parque não existe no sistema.
  - `Dados invalidos.`
    Duração do pickup é menor ou igual a zero.

3.3.10 Carregamento de saldo de utilizador

- SINTAXE DE ENTRADA

  ChargeUser *idUser valor*↵

  ↵

  DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Carregamento do saldo do utilizador *idUser* com o valor *valor*. A operação só terá sucesso se *idUser* já existir no sistema e se *valor* for positivo.

A mensagem de saída da operação efetuada com sucesso deverá incluir o saldo (*saldo*) atualizado do utilizador (ver abaixo).

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  `Saldo: saldo euros.↵`

  ↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-de-carregamento ↵

  ↵

- A *mensagem-de-erro-de-carregamento* é uma das seguintes:
  - `Utilizador inexistente.`
    Quando o identificador do utilizador não existe no sistema.
  - `Dados invalidos.`
    Valor de carregamento é menor ou igual a zero.

3.3.11 Listar pickups terminados de bicicleta

- SINTAXE DE ENTRADA

  BikePickUps *idBike*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem dos pickups terminados da bicicleta identificada por *idBike*. Esta operação só terá sucesso se *idBike* existir no sistema e se a bicicleta já tiver sido utilizada e o seu primeiro pickup já tiver terminado. A listagem é ordenada por ordem cronológica do pickup e deve incluir o identificador do utilizador que efetuou o pickup (*idUser*), o identificador do parque onde foi feito o pickup (*initialIdPark*) e a entrega (*finalIdPark*), a duração do pickup (*minutos*), o número de minutos de atraso (*atraso*) e o *valor* (em euros) atribuído ao pickup (ver sintaxe de saída abaixo).

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-pickups

  mensagem-de-listagem-pickups

  …

  mensagem-de-listagem-pickups

  ↵

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

idUser initialIdPark finalIdPark minutos atraso valor↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  mensagem-de-erro-listagem-pickups↵

  ↵

- A *mensagem-de-erro-listagem-pickups* é uma das seguintes:
  - `Bicicleta inexistente.`
    Quando o identificador da bicicleta não existe no sistema.
  - `Bicicleta nao foi utilizada.`
    Quando a bicicleta não foi ainda alvo de pickups.
  - `Bicicleta em movimento em primeiro pickup.`
    O primeiro pickup da bicicleta ainda não terminou.

3.3.12 Listar pickups terminados de utilizador

- SINTAXE DE ENTRADA

  UserPickUps *idUser*↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem dos pickups terminados do utilizador identificado por *idUser*. Esta operação só terá sucesso se *idUser* existir no sistema e se o utilizador já tiver efetuado algum pickup e se o seu primeiro pickup já tiver terminado. A listagem é ordenada por ordem cronológica do pickup e deve incluir o identificador da bicicleta (*idBike*), o identificador do parque onde foi feito o pickup (*initialIdPark*) e a entrega (*finalIdPark*), a duração do pickup (*minutos*), o número de minutos de atraso (*atraso*) e o *valor* (em euros) atribuído ao pickup (ver sintaxe de saída abaixo).

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-pickups2

  mensagem-de-listagem-pickups2

  …

  mensagem-de-listagem-pickups2

  ↵

Cada *mensagem-de-listagem-pickups2* terá a seguinte forma:

idBike initialIdPark finalIdPark minutos atraso valor↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

mensagem-de-erro-listagem-pickups2↵

↵

- A *mensagem-de-erro-listagem-pickups2* é uma das seguintes:
  - `Utilizador inexistente.`
    Quando o identificador do utilizador não existe no sistema.
  - `Utilizador nao utilizou o sistema.`
    Quando o utilizador ainda não efetuou nenhum pickup.
  - `Utilizador em primeiro PickUp.`
    O utilizador ainda não terminou o primeiro pickup.

3.3.13 Verificar se bicicleta está estacionada em parque

- SINTAXE DE ENTRADA

ParkedBike *idBike idPark*↵

↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

Este comando permite verificar se a bicicleta com o identificador *idBike* está estacionada no parque identificado por *idPark*. Esta operação só terá sucesso se *idBike* e *idPark* já existirem no sistema e se a bicicleta em questão estiver estacionada no parque mencionado.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

Bicicleta estacionada no parque.↵

↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

mensagem-de-erro-verificacao-estacionamento↵

↵

- A *mensagem-de-erro-verificacao-estacionamento* é uma das seguintes:
  - `Bicicleta inexistente.`
    Quando o identificador da bicicleta não existe no sistema.
  - `Parque inexistente.`
    Quando o identificador do parque não existe no sistema.
  - `Bicicleta nao esta em parque.`
    Quando a bicicleta em questão não se encontra no parque.

3.3.14 Listagem de utilizadores (com pontos de atraso)

- SINTAXE DE ENTRADA

  `ListDelayed`↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem dos utilizadores que tenham entregue alguma bicicleta com algum atraso. Esta listagem estará ordenada decrescentemente pelo número de pontos e, em caso de empate, por ordem cronológica de obtenção dos pontos. Os empates deverão ser pouco expressivos. Cada linha da listagem irá incluir o *nif*, *nome*, a *morada*, o *email*, o número de *telefone* do utilizador assim como o seu *saldo* e o número de *pontos* obtidos devido a atrasos na entrega de bicicletas.

  Fases 1 e 2: Nestas fases não serão registados atrasos. As bicicletas serão entregues sempre dentro do prazo gratuito.

  Fase 3: Nesta fase podem registar-se atrasos.

- SINTAXE DE SAÍDA – OPERAÇÃO EFECTUADA COM SUCESSO

  mensagem-de-listagem-utilizadores

  mensagem-de-listagem-utilizadores

  …

  mensagem-de-listagem-utilizadores

  ↵

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

  nome: *nif, morada, email, telefone, saldo, pontos*↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  `Nao se registaram atrasos.`↵

  ↵

3.3.15 Listar parques mais movimentados

- SINTAXE DE ENTRADA

  FavoriteParks↵

  ↵

- DESCRIÇÃO DA OPERAÇÃO (SE EFECTUADA COM SUCESSO)

  Listagem de todos os parques mais movimentados do sistema (com o número máximo de pickups que já se realizaram no sistema) ordenados crescentemente pelo nome do parque.

- 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:

  nome: *morada, pickups*↵

- SINTAXE DE SAÍDA – OPERAÇÃO NÃO EFECTUADA

  Não foram efetuados pickups.↵

  ↵

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 utilizadores, de parques e bicicletas, assim como de possíveis pickups. Cada utilizador poderá realizar milhares de pickups distribuídos pelos parques da cidade. 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 parques e bicicletas.

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
bikePickUp
```

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

```
zip aed.zip Main.java dataStructures/*.java bikePickUp/*.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 8 e 12 de outubro de 2018. No dia 12 de outubro a entrega eletrónica tem de ser realizada até às **17h**.
- Submissão da fase 2 do trabalho, no Mooshak: entre 5 e 9 de novembro de 2018. No dia 9 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 26 e 30 de novembro de 2018. No dia 30 de novembro, a entrega eletrónica tem de ser realizada até às **17h**.
- Marcação das discussões finais: 3 de dezembro de 2018, no Moodle.
- Realização das discussões: entre 4 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.