Universidade Lusófona de Humanidades e Tecnologias
LEI / LIG / LEIRT
Algoritmia e Estruturas de Dados
2020/2021
2º Semestre - 1ª época
v1.0.0

Enunciado do projeto prático

 DEISI Rockstar 2021 

Primeira Parte

Objetivos

Este projeto tem como objetivo o desenvolvimento de um programa/aplicação (o "**DEISI Rockstar 2021**") em linguagem Java que seja capaz de receber e responder a pedidos de informações/estatísticas sobre uma base de dados de músicas e artistas (bandas, músicos, etc.).

O programa terá de ler dados dessa base de dados, que será fornecida a partir de ficheiros num formato especificado mais à frente.

Tendo em conta que um dos temas tratados em **AED** é a questão da **eficiência** das estruturas de dados e dos algoritmos, pretende-se que o programa desenvolvido seja não só funcionalmente correto, mas também que seja o mais rápido possível.

Para alcançar estes objetivos, os alunos devem ter em conta tudo o que foram aprendendo em AED. Recomenda-se que façam abordagens exploratórias, fazendo medições dos tempos de execução de cada sub-passo do programa, no sentido de encontrar eventuais *bottlenecks* e que se dediquem a tentar melhorar os tempos de execução dos mesmos.

"*Make it work, Make it right, Make it fast.*"

Bruno Cipriano, Rodrigo Correia, Bruno Saraiva, Pedro Alves

Organização do trabalho

Este trabalho está dividido em duas partes, sendo ambas de entrega obrigatória.

Este é o enunciado da Primeira Parte.

O enunciado da Segunda Parte será publicado em *moodle* em data posterior.

A não entrega de qualquer uma das partes do projeto implica reprovação na componente prática da disciplina.

Objetivos da Primeira Parte

Nesta primeira parte do projeto pretende-se que os alunos realizem as seguintes tarefas:

- Definição e implementação de classes Java para representação das entidades da realidade/domínio do problema;
- Implementação da Leitura dos Ficheiros que são o *input* do programa;
- Criação (instanciação) de objectos em memória, considerando os ficheiros lidos.

Realidade / Domínio do Problema

As entidades principais neste problema são:

- Artista (banda, músico, etc.)
- Tema Musical

A entidade **Artista** é caracterizada por:

- ID - uma String
- Nome - uma String

A entidade **Tema Musical** é caracterizada por:

- ID - uma String
- Título - uma String
- Artista(s) envolvidos - uma Lista com um ou mais Artistas
- Ano de lançamento - um número inteiro (positivo e menor que 2021)
- Duração do tema - um número inteiro representando milissegundos
- Letra Explícita - um booleano que indica se a letra contém termos obscenos ou grosseiros
- Popularidade - um número inteiro positivo, quanto maior mais popular
- Grau de “dançabilidade” - quanto maior mais dançável é a música
- Grau de “vivacidade” - quanto maior mais animada é a música
- Volume médio - representado em decibéis

Dados de entrada (*Input*) do programa

O *input* do programa consiste em **três** ficheiros de texto:

Nome do ficheiro	**Conteúdo**
songs.txt	Este ficheiro descreve os temas musicais.
song_artists.txt	Este ficheiro descreve quais os artists que participam em cada tema musical. (Nota: Algumas músicas poderão não ter artistas associados)
song_details.txt	Este ficheiro descreve diversos detalhes sobre os temas musicais como a sua duração, popularidade, etc. (Nota: Poderão faltar alguns detalhes para algumas músicas)

Não são conhecidas, à priori, as dimensões (número de linhas) destes ficheiros.

De seguida são apresentadas as sintaxes de cada um dos ficheiros.

Ficheiro songs.txt

Cada linha deste ficheiro descreve um **Tema Musical** e segue a seguinte sintaxe:

```
<ID Tema Musical> @ <Nome> @ <Ano lançamento>
```

Onde:

- `<ID Tema Musical>` é uma `String`;
- `<Nome>` é uma `String`;
- `<Ano lançamento>` é um `int`

Tratamento de erros

Como se pode ver, este ficheiro deverá ter 3 componentes em cada linha.

No entanto, pode acontecer que existam ficheiros com dados a mais, ou dados a menos. **As linhas que tenham dados a mais ou a menos devem ser ignoradas.**

Ficheiro song_artists.txt

Cada linha deste ficheiro segue uma das seguintes sintaxes consoante a música está associada a apenas um artista ou vários artistas:

```
<ID Tema Musical> @ ['<Nome Artista>']
<ID Tema Musical> @ "['<Nome Artista>','<Nome Artista>',...]"
```

Onde:

- `<ID Tema Musical>` é uma `String;`
- `<Nome Artista>` é uma `String;`

Como se pode ver, os nomes dos artistas estão sempre entre plicas. Caso sejam vários estão separados por `",` (*vírgula*). Todo o conjunto de artistas (seja um ou vários) está sempre entre parênteses rectos mas caso sejam vários, ainda estão entre aspas.

Notas:

O mesmo artista pode estar associado a vários Temas Musicais.

Algumas músicas poderão não estar associadas a nenhuns artistas. Nota: Na parte 1 do projeto, esta situação não será testada.

Tratamento de erros

Como se pode ver, este ficheiro deverá ter 2 componentes em cada linha. No entanto, pode acontecer que existam ficheiros com dados a mais, ou dados a menos. **As linhas que tenham dados a mais ou a menos devem ser ignoradas.**

Ficheiro song_details.txt

Cada linha do ficheiro segue a seguinte sintaxe (tudo na mesma linha):

```
     <ID Tema Musical> @ <Duração> @ <Letra explícita> @
<Popularidade> @ <Dançabilidade> @ <Vivacidade> @ <Volume médio>
```

Onde:

- `<ID Tema Musical>` é uma `String;`
- `<Duração>` é um número inteiro representando milissegundos
- `<Letra explícita>` - um inteiro com o valor 1 quando é uma letra explícita e 0 nos outros casos
- `<Popularidade>` - um número inteiro positivo
- `<Dançabilidade>` - um número real
- `<Vivacidade>` - um número real
- `<Volume médio>` - um número real representando decibéis

Tratamento de erros

Como se pode ver, este ficheiro deverá ter 7 componentes em cada linha. No entanto, pode acontecer que existam ficheiros com dados a mais, ou dados a menos. **As linhas que tenham dados a mais ou a menos devem ser ignoradas.**

Dados de saída (Output) do programa

Nesta primeira parte do trabalho não há necessidade de fazer qualquer *output*.

Restrições Técnicas e Comportamento a implementar

Existem algumas **restrições técnicas** que terão de ser obrigatoriamente cumpridas pelo código dos alunos. Trabalhos que não cumpram estas restrições serão avaliados com a nota **zero**.

As restrições são as seguintes:

- Existem três classes obrigatórias:
  - `Main`
  - `Song`
  - `ParseInfo`

- A classe **`Song`** tem de ser capaz de representar a entidade **Tema Musical** previamente descrita.

- A classe **`Song`** deve conter um método `toString()` que devolva uma String representando alguns dos dados do tema musical, conforme se descreve de seguida:

```
ID | Título | Ano
```

Exemplo:

```
1oYYd2gnWZYrt89EBXdFiO | Message In A Bottle | 1979
```

Nota: Para garantir que o projeto passa os testes automáticos dos Professores, deverá haver um espaço (e apenas um espaço) à volta de cada separador. Por exemplo:

`"1oYYd2gnWZYrt89EBXdFiO|Message In A Bottle|1979"`

não será considerado correto, pois não tem espaçamento certo.

- A classe **`ParseInfo`** será utilizada durante a leitura dos ficheiros para guardar as seguintes informações:
  - Número de linhas lidas que estão OK
  - Número de linhas lidas que foram ignoradas
- A classe **`ParseInfo`** deve conter um método `toString()` que devolva uma String representando estas estatísticas, conforme se descreve de seguida:

```
OK: <NUM_LINHAS_OK>, Ignored: <NUM_LINHAS_IGNORED>
```

Exemplo:

```
OK: 341, Ignored: 5
```

- A classe principal do projeto deve ter o nome Main.

- A classe Main deve incluir obrigatoriamente as seguintes funções:

```
public static void loadFiles() throws IOException
public static ArrayList<Song> getSongs()
public static ParseInfo getParseInfo(String fileName)
```

- A função **loadFiles()**, ao ser invocada, deverá:
  - ler os três ficheiros de input e criar objetos da classe **`Song`** (e outras classes que achem necessárias) com os dados existentes nos ficheiros;
  - guardar esses objetos em estruturas de dados apropriadas;
  - a parte "throws IOException" indica que a função poderá lançar uma exception relacionada com a leitura de ficheiros;
- Nota Importante: Na parte 1 do projeto não vão ser testadas as associações entre estes objetos (ex: quais os artistas associados a uma certa música). Isso será desenvolvido na parte 2.

- A função **`getSongs()`** deverá retornar um `ArrayList` com todos os temas musicais (válidos) que existem na base de dados carregada a partir dos ficheiros. A ordem pela qual as músicas aparecem no ficheiro `songs.txt` deve ser preservada.

- A função **`getParseInfo(...)`** deverá retornar as estatísticas associadas à leitura do ficheiro cujo nome foi passado como argumento à função, por terem dados a menos ou

a mais do que o previsto. Caso o nome do ficheiro passado como argumento seja inválido, a função deve retornar null. Nota: Os testes executarão sempre a função loadFiles() antes de executarem esta função.

- A existência ou não de outras classes para além das 2 acima indicadas fica ao critério dos alunos.

- Nesta parte, não é permitida a utilização da classe Map ou qualquer uma das suas variantes (HashMap, LinkedHashMap, TreeMap, ConcurrentHashMap, etc.).

- Todas as classes do projeto devem pertencer ao seguinte *package*:

```
pt.ulusofona.aed.deisiRockstar2021
```

Dica:

Os testes automáticos dos professores podem chamar a função `loadFiles()` mais do que uma vez. Os alunos devem garantir que nunca ocorre duplicação de informação no `ArrayList<Song>` devolvido por essa função (p.e. fazendo um *reset* das estruturas de dados relevantes).

Entrega e Avaliação

O que é preciso entregar?

Nesta primeira parte do projeto, os alunos têm de entregar:

- Todo o código necessário para compilar e executar o projeto;
- Um ficheiro de texto (chamado AUTHORS.txt) contendo os nomes e números de aluno dos membros do grupo).

A entrega deve ser feita usando um ficheiro comprimido (formato .zip).

Formato de entrega

O ficheiro .zip deve seguir a estrutura que se indica de seguida:

```
AUTHORS.txt  (contém linhas NUMERO_ALUNO;NOME_ALUNO, uma por aluno do grupo)
… (outros ficheiros)
+ src
|---+ pt
|------+ ulusofona
|---------+ aed
|------------+ deisiRockstar2021
|--------------- Main.java
|--------------- ...   (outros ficheiros java do projecto)
|--------------- TestXXX.java
|--------------- ... (outras classes de testes)
```

Reparem que esta estrutura corresponde ao *package* obrigatório previamente indicado.

Nota importante: Os ficheiros de *input* devem ser lidos a partir da raíz do projeto.

Como entregar - Drop Project

Os alunos terão acesso a uma página no **Drop Project** a partir da qual poderão submeter o seu projeto e obter um relatório sobre os testes aplicados ao mesmo.

O DP da primeira parte do projeto estará disponível no URL seguinte:

https://deisi.ulusofona.pt/drop-project/upload/aed-2021-projeto-p1

Nota importante: O programa submetido pelos alunos tem de compilar e executar no contexto do Drop Project.

Filosofia do uso do DP

O objetivo do DP não é servir de guião àquilo que os alunos têm que implementar. Os alunos devem implementar o projeto de forma autónoma tendo apenas em conta o enunciado e usar o DP apenas para validar que estão no bom caminho. Nas empresas nas quais um dia irão trabalhar não vão ter o DP para vos ajudar.

Desta forma, as submissões no DP serão limitadas temporalmente: cada grupo apenas pode fazer uma submissão a cada 30 minutos (isto é, sempre que fazem uma submissão, têm que esperar 30 minutos até que possam fazer outra submissão).

Prazo de entrega

A data limite de entrega é o **dia 23 de Abril de 2021, pelas 22h00m** (hora de Lisboa, Portugal). Não serão aceites trabalhos após essa data e hora.

Avaliação

A avaliação do projeto será dividida pelas 2 partes:

- Nas duas partes existirão baterias de testes automáticos implementadas no sistema **Drop Project** ([DP]) que irão avaliar o projeto do ponto de vista funcional.
- Será também feita, em cada parte, uma avaliação por parte dos Professores.

- Existirá uma nota em cada parte do projeto.

- Na primeira parte do projeto existe a nota mínima de 8 (oito) valores.
    - Quem não alcançar essa nota mínima ficará excluído da avaliação de primeira época.

- Após a entrega final, será atribuída ao projeto uma nota final quantitativa, que será calculada considerando a seguinte fórmula:

    - `0.25 * Nota_Parte_1 + 0.75 * Nota_Parte_2`

- No final, existirá uma defesa presencial do projeto.
    - Mais informações na secção "**Outras informações relevantes"** deste enunciado.

Cotações da primeira parte

Item	**Descrição**	**Cotação**
Componente funcional	Testes automáticos do DP.	12
Avaliação do Professor	Os Professores irão realizar testes extra das funcionalidades a implementar.	4
Qualidade do código (checkstyle)	Alguns indicadores da qualidade de código como variáveis com nomes auto-explicativos ou funções que não sejam demasiado grandes	1.5
Testes automáticos	Os alunos devem implementar 2 casos de teste em JUnit para o método `toString()` da sua classe Song. Por caso de teste considera-se uma função/método anotada com `@Test`. A versão do JUnit a usar deve ser a 4.	1.5
Participação no piazza	Alunos que participem ativamente no piazza. Ver secção "Esclarecimento de Dúvidas"	1

Cópias

Trabalhos que sejam identificados como cópias serão anulados e os alunos que os submetam terão nota zero em ambas as partes do projeto (quer tenham copiado, quer tenham deixado copiar).

Para evitar situações deste género, recomendamos aos alunos que nunca partilhem ou mostrem o código do seu projeto a pessoas fora do grupo de trabalho.

A decisão sobre se um trabalho é uma cópia cabe exclusivamente aos docentes da unidade curricular.

Esclarecimento de dúvidas

Todas as dúvidas relativas ao projeto (seja de interpretação do enunciado, seja da forma como se implementa) devem ser colocadas na ferramenta **piazza**.

Os alunos inscritos irão receber um convite por email para aderirem à ferramenta. O email será aquele ao qual estão associados no Moodle. Deverão registar-se com o nome no seguinte formato: <Primeiro Nome> <Último Nome> (<Número>). Exemplo: António Silva (21801234).

As regras de utilização estão explicadas no próprio piazza.

Os professores darão prioridade máxima às perguntas colocadas no piazza, em detrimento de perguntas colocadas por outras vias. Alunos que coloquem dúvidas por outras vias (ex: email) poderão só ter resposta passados vários dias e a resposta poderá ser “Essa dúvida deverá ser colocada no piazza”.

Outras informações relevantes

– Os projetos devem ser realizados em grupos de **2 alunos,** de preferência que pertençam à mesma turma prática. Excepcionalmente poderão ser aceites trabalhos individuais, desde que seja pedida autorização prévia ao Professor das aulas práticas e desde que exista uma justificação razoável.

- Na segunda parte do projeto devem-se manter os grupos definidos para a primeira parte. Não será permitida a entrada de novos membros nos grupos.

– Existirá uma defesa presencial e individual do projeto. Durante esta defesa individual, será pedido ao aluno que faça alterações ao código para dar resposta a alterações aos requisitos. Da discussão presencial de cada aluno, resultará uma nota de 0 a 100%, que será aplicada à nota do projeto.

- O ficheiro .zip entregue deve seguir escrupulosamente as regras de estrutura indicadas neste enunciado. Ficheiros que não respeitem essa estrutura terão nota zero.

– Trabalhos cujo código não compile e/ou não corra não serão avaliados e terão automaticamente nota zero.

- Trabalhos que não apresentem resultados devido à existência de *runtime* errors (p.e. `NullPointerException`, `ArrayIndexOutOfBounds`, etc.) não serão avaliados e terão automaticamente nota zero.

- Grupos que não entreguem a primeira parte ou tenham nota zero na mesma (seja por cópia, não compilação ou outra das situações indicadas no enunciado), não podem entregar a segunda parte do projeto, e reprovam automaticamente na componente prática da disciplina (de primeira época).

– É possível que sejam feitas alterações a este enunciado, durante o tempo de desenvolvimento do projeto. Por esta razão, os alunos devem estar atentos ao Moodle de AED.

- Eventuais situações omissas e/ou imprevistas serão analisadas caso a caso pelos docentes da cadeira.

Anexo I - Exemplos dos ficheiros de input

Neste anexo é apresentado um exemplo de cada um dos três ficheiros de *input*.

Este exemplo deve ser usado pelos alunos para testarem o **comportamento** do seu programa, antes de o submeterem para avaliação.

Como se trata de um exemplo pequeno, não será uma boa forma de testar a **eficiência** do programa dos alunos. Desta forma, sugere-se que os alunos criem ficheiros de *input* (com muitos dados) destinado aos testes de eficiência.

Ficheiros

songs.txt

```
17ZnveSDBpG9QtL7zLJNPy @ Only For You @ 2012
0tStWvUMHODuLN4TIaSGab @ Loops & Tings - Radio Edit @ 2013
37zoJIMxB98H2fbidrwxUO @ Downshift - Original Mix
1sNctqVr9zdS7i1RZNjIgY @ Crazy Kids @ 2012
2pGlftIo2UcDfZznGTCahM @ Move - Tune Brothers Remix @ 2013
1ODA4mZMaoBzT1TbPBW8B1 @ Shiver @ 2012
60dEggLtB2CRgq14EtoCip @ 24Seven - Dan Lemur Progressive Remix @ 2013
0nxvFG50rGXkiGQqOO2MHr @ Be Alright @ 2012
```

Nota: a terceira linha tem apenas 2 componentes, em vez dos 3 esperados. Por esta razão, essa linha tem de ser ignorada.

song_artists.txt

```
7dD5DEzjhofoItSG7QwVoY @ ['Frank Sinatra']
5TnAnWRXDM6awfvAveK91N @ ['Mountain John']
5tNhreO6L8kfvQSqPjuGUg @ ['Rammstein']
4Cna7QxOOTNnylVHLtShCi @ ['Madonna']
2Sy1r4fGfq4Lslfc0gHkbk @ "['La Sonora Matancera', 'Carlos Argentino']"
3uXp4hZmCnEmVjffupKCiT @ ['Supertramp']
```

song_details.txt

```
0xJ6Xr620TwVr6kqDSk5tn @ 347951 @ 0 @ 51 @ 0.397 @ 0.127 @ -6.678
2GKyRxh2DAC0sSVddqUE4l @ 178077 @ 0 @ 4 @ 0.292 @ 0.0993 @ -8.497
1lxVGQgbt0g7mMkmevl9Bm @ 205613 @ 0 @ 34 @ 0.608 @ 0.115 @ -10.265999999999998
0GmLrYUBXDC5vti77zBZfJ @ 167680 @ 0 @ 36 @ 0.5770000000000001 @ 0.226 @ -9.825
```