Proposta de Trabalho 1 / Project Proposal 1:
Music Streaming Information System

Algoritmos e Estruturas Dados II / Algorithms and Data Structures II

José Manuel Torres

jtorres@ufp.edu.pt

Fevereiro de 2016 / February 2016

Universidade Fernando Pessoa

Faculdade de Ciência e Tecnologia / Faculty of science and technology

1 Descrição do Problema

Pretende-se, neste projeto, simular parcialmente as funcionalidades dum serviço online de informação de músicas em streaming[1]. A aplicação irá gerir tabelas de símbolos (symbol tables - ST) de: utilizadores, músicas, artistas, géneros musicais e playlists. Além disso, as músicas podem ser agrupadas por: género musical, playlist, artista, e por histórico de audição de um utilizador. Estes agrupamentos serão modelizados por tabelas de símbolos de elementos em que cada valor poderá ser outra tabela de símbolos.

Deverá utilizar STs para implementar as principais coleções envolvidas. Para cada ST deverá considerar, pelo menos, os seguintes atributos:

- Utilizadores: username (chave única), email, nome
- Músicas: International Standard Recording Code - ISRC (chave única), titulo, duração em segundos, artista, género.
- Artistas: username do artista (chave única), nome, género musical do trabalho do artista.
- Géneros musicais: chave género (chave única).
- Playlists: chave nome da playlist.

As STs seguintes ficam aninhadas dentro de uma das STs anteriores.

- Género - Músicas: chave será o ISRC. Cada instância desta ST contém as músicas dum determinado género.
- Artistas - Músicas: Cada instância desta ST contém as músicas dum artista.
- Utilizadores – Histórico de Músicas: como chave o tempo em que utilizador fez play da música. Cada instância desta ST contém as músicas reproduzidas por um utilizador.
- Playlists – Músicas: chave ordenada. Cada instância desta ST contém as músicas duma playlist.

A aplicação a ser desenvolvida deverá suportar/implementar a utilização de tabelas de símbolos usando árvores balanceadas e tabelas de *hash* usando a biblioteca de funções usadas na unidade curricular.

[1] Alguns serviços deste tipo: Spotify, MEO Music, Apple Music

A informação deverá poder ser carregada para a memória do sistema usando ficheiros de texto de entrada (*restore* / *load* da informação). O sistema poderá, igualmente, gravar em ficheiros de texto saída os dados residentes em memória (*dump* / *backup* da informação). Deverá ainda considerar a geração de dados sintéticos de teste na própria aplicação (eg: gerar utilizadores fictícios) e extrair estatísticas de desempenho.

2 Requisitos Funcionais

Pretende-se que desenvolva uma API de funções (biblioteca / conjunto de funções) que satisfaçam os requisitos listados a seguir e ainda casos de teste dessa API (funções de teste). Note que não é requisito do projeto o desenvolvimento uma interface de utilizador completamente funcional. Descrevem-se a seguir os requisitos a cumprir neste projeto de programação:

R1. Criar modelos de dados para os valores (classe genérica java Value das STs) dos elementos das várias tabelas de símbolos consideradas no projeto.

R2. Usar funções e tabelas de hash em, pelo menos, duas STs cuja chave não é ordenável (eg: utilizadores e artistas).

R3. Usar BSTs balanceadas (*redblack*) em, pelo menos, as STs cuja chave é ordenável como tempos ou ordem.

R4. Criar funções para inserir, remover, editar e listar tudo, para as várias STs consideradas: utilizadores, músicas, artistas, géneros musicais e playlists presentes na base de dados.

R5. Validar que todas as músicas que são consideradas noutras STs existem na ST músicas.

R6. Implementar várias pesquisa à base de informação como, por exemplo: quais as playlists em que aparece uma dada música; qual é a música mais popular nas diferentes playlists; qual o artista e género de música foram mais ouvidos num dado dia ou intervalo de tempo, entre outras pesquisas.

R7. Remover músicas da base de dados com validação da sua remoção total do sistema.

R8. Popular as diversas STs da aplicação com o conteúdo de ficheiros de texto de entrada (*restore* / *load* da informação em ficheiro). Gerar dados sintéticos de teste na própria aplicação (eg: gerar utilizadores fictícios) para popular as várias STs.

R9. Fazer o output (*dump*) de toda a informação para ficheiros de textos, isto é, de utilizadores, artistas, músicas, playlists e de todas as outras STs.

R10. Correr casos de teste (funções de teste) para todas as funcionalidades implementadas nos requisitos anteriores, considerando, para o efeito, uma ou várias classes de teste específicas. Deverá implementar mecanismos / instrumentos de medição temporal da execução das principais funções de inserção e pesquisa nesses testes. Esses cenários deverão incluir nos testes a leitura e escrita de ficheiros com dados de input (batch input data) e/ou resultados de output. Os resultados dos testes deverão ser escritos para ficheiro de texto. Para cada teste realizado deverá ser explicitada (no ficheiro de texto) a função testada e o resultado do teste.

3 Estilo, clareza de programação e organização do código fonte de AED2

Deve adotar um estilo de programação que seja claro, eficaz do ponto de vista de implementação e de fácil compreensão por humanos, considerando aspetos como: convenções de nomes das variáveis, organização das funções e atributos, clareza na implementação dos algoritmos entre outros[2].

4 Documentação de AED2

4.1 Anotações e comentários no código fonte

As estruturas de dados e os algoritmos definidos devem ser implementados em Java com os comentários apropriados, que facilitem a compreensão dos mesmos, escritos no código fonte desenvolvido. Todas as funções devem estar anotadas em formato javadoc[3] incluindo uma explicação dos algoritmos implementados e uma menção ao desempenho dos mesmos assim como dos testes efetuados/implementados. As principais estruturas de dados e variáveis devem estar anotadas neste formato. Deverão usar o software javadoc para gerar a documentação com base nos comentários.

[2] Exemplo: http://google.github.io/styleguide/javaguide.html
[3] (http://docs.oracle.com/javase/8/docs/technotes/guides/javadoc/index.html) Suportada pelo IDE NetBeans. Ver: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html , http://javaworkshop.sourceforge.net/chapter4.html

4.2 Relatório

Os alunos deverão ainda criar um texto / relatório que funciona como folha de rosto da documentação produzida automaticamente com o javadoc (source file javadoc do tipo: overview documentation comment file), no formato dum ficheiro html localizado na pasta dos ficheiros fonte do projeto, que descreva as funcionalidades/requisitos implementados, parcialmente implementados e não implementados, mencionando sempre o número do requisito de acordo com este documento de especificação:

1. Funcionalidades implementadas: devem descrever de forma resumida todas as funções desenvolvidas para assegurar os requisitos funcionais solicitados.
2. Funcionalidades não implementadas: nesta secção devem identificar de forma resumida as funcionalidades não implementadas; devem ainda apontar-se as justificações/dificuldades que impediram o seu desenvolvimento.

5 Submissão

A aplicação final (código fonte) deve estar depurada de todos os erros de sintaxe e de acordo com os requisitos funcionais pedidos. Só serão considerados os programas que não contenham erros de sintaxe e implementem as funcionalidades pedidas. A documentação, em html, juntamente com todo o código-fonte desenvolvido, deve ser submetida num ficheiro zip/rar (project1.zip) na plataforma elearning.ufp.pt (cacifo digital / dropbox).