Relatório

João Vitor Venceslau Coelho – 2016040403

Gleydvan Macedo – 20160147018

Introdução

O projeto consiste na implementação de um tocador de música (media player), com funcionalidades para gerenciar conjuntos de músicas. Foi utilizado o paradigma da orientação a objetos, aprendido durante a disciplina de LP2, conhecimentos adquiridos em EDB2 e alguns conhecimentos adicionais adquiridos por meio da internet.

Objetivos

O objetivo principal desse trabalho é aplicar o conteúdo aprendido na disciplina de Linguagem de Programação II implementando um media player. Segue abaixo a lista dos conteúdos:

1. Organização de pacotes;
2. Interface (GUI) Swing;
3. Utilização de bibliotecas externas;
4. Documentação JavaDoc;
5. Herança;
6. Polimorfismo;
7. Classe Abstrata;
8. Interfaces;
9. Classes para tratamento de Exceções;
10. Utilizar pelo menos 2 padrões de projetos;
11. Testes unitários.

Informações sobre o Projeto

O projeto consiste em um tocador de MP3 (*media player)*, capaz de produzir arquivos de áudio individualmente, ou como parte de uma *playlist.* Especificamente, o sistema possui, assim como determinado, as seguintes funcionalidades:

1 – Reproduzir uma música (já cadastrada), ou uma *playlist* de músicas;

2 – Adicionar/remover um arquivo de música, ou todos os arquivos de um diretório;

3 – Criar uma *playlist* com músicas previamente adicionadas ao media player;

4 – Buscar músicas cadastradas;

5 – Autenticar um usuário e diferenciar ele em diferentes perfis de acesso;

6 – Cadastro de novos usuários;

Além de funcionalidades extras, comentadas ao fim do documento.

Controle de Acesso:

Ao realizar o logon na aplicação, os usuários são distinguidos entre usuários comuns e usuários VIP, onde, assim como especificado:

- Usuários comuns podem:
    - adicionar arquivos de música individualmente, ou um diretório com arquivos de música ao banco (conjunto) de arquivos do *media player*;
        - após a adição de um (ou vários) arquivos de música, as mesmas estarão visíveis a todos os usuários;
        - músicas adicionadas estarão ainda disponíveis no sistema em próximas execuções do mesmo (mesmo que o usuário feche o sistema e efetue um novo login);
    - tocar uma música previamente cadastrada no media player;
        - músicas podem ser selecionadas a partir de uma lista, ou filtradas pelo nome;
- Adicionalmente, usuários “VIP” podem:
    - criar playlists, que posteriormente serão visualizadas apenas por ele;
    - cadastrar usuários para acesso ao player;
- O player possui um usuário “VIP” default (admin) para iniciar aplicação. (Login e Senha são “admin”).

Salvando os dados da aplicação:

Para armazenar os dados da aplicação, utilizamos arquivos de texto **.txt.** Manipulando-os

de modo a salvar os dados de maneira correta, permitindo um acesso futuro. O arquivo *musicas.txt* guarda as informações sobre as músicas cadastradas, o arquivo *usuarios.txt* armazena a lista de usuários e os arquivos presentes na pasta playlists são arquivos para as playlists dos usuários VIPs. Sendo estes arquivos lidos quando o programa é aberto e alterados caso o usuário solicite, realize logoff ou feche a aplicação de maneira normal.

Especificamente:

- Musicas.txt: Salva o nome de cada música e seu respectivo caminho. Uma música por linha.
- playlist_xxx.txt: Exite um arquivo deste para cada playlist cadastrada. Como as playlists são privadas por usuário (cada usuário VIP pode criar e editar suas playlists) cada playlist também armazena um identificador para o seu dono.
- usuarios.txt: Armazena os usuários cadastrados, permitindo a autenticação durante o login.

Objetivos Alcançados

1 – Organização de pacotes

A organização foi feita com os seguintes pacotes:

lp.projeto.musicplayer	- Main da aplicação
lp.projeto.musicplayer.model	- Classes que modelam a aplicação
lp.projeto.musicplayer.utility	- Classes de suporte para a aplicação
lp.projeto.musicplayer.view	- Controladores e FXML da GUI
lp.projeto.musicplayer.tests	- Classes de testes

2 – Interface (GUI) Swing ou JavaFX

Inicialmente seria utilizado o Swing para o desenvolvimento da interface gráfica porém após algumas complicações durante a organização do código gerado automaticamente pelas ferramentas de construção de interface, optou-se por refazer tudo com o JavaFX, por utilizar um FXML para a parte gráfica, o código se tornou mais simples de organizar e as funcionalidades mais facilmente atribuídas a objetos como botões, listas, campos de preenchimento, etc.

3 – Utilização de bibliotecas externas

Não foi utilizada propriamente uma biblioteca externa, inicialmente seria utilizada a JLayer do JavaZoon para reproduzir as músicas, porém durante o decorrer do projeto optou-se por utilizar a classe MediaPlayer presente no JavaFX pois mesmo a Jlayer sendo mais simples de se utilizar, a classe MediaPlayer servia melhor para o propósito da aplicação.

4 – Documentação JavaDoc

Todas as classes e métodos possuem uma documentação JavaDoc, informando sucintamente seu propósito. Nos métodos também é explicado oque se espera como entrada e quais são os possíveis retornos, assim como exceções que podem ser lançadas.

5 – Herança

Foi utilizada herança nas seguintes classes: User e UserVip, que representam respectivamente um usuário comum e um com privilégios a mais.

6 – Polimorfismo

Foi utilizado o polimorfismo principalmente no manuseio dos usuários e nos construtores de classes como Music e PlayList.

7 – Classe Abstrata

Não foi utilizada nenhuma classe abstrata.

8 – Interfaces

Utilizou-se o conceito de interfaces na classe Named que é uma interface e a classe User implementa tanto a interface Comparable como a Named, requisitos para poder ser utilizada pela AvlTree, uma árvore de busca binaria otimizada com a ideia da Árvore AVL e com uma busca por nome de objeto.

9 – Classes para tratamento de Exceções

As seguintes classes são para tratamento de exceções: IncorrectPasswordException, NoMoreMusicException, NoMusicSelectedException, UserExistException, UserNotRegisteredException e UserPasswordInvalidException.

10 – Utilizar pelo menos 2 padrões de projetos

Foram utilizados os padrões MVC e o Singleton, sendo o MVC utilizado na propria arquitetura da aplicação separando interface, controladores e as classes funcionais da aplicação. Enquanto a Classe DataManager funciona como um Singleton, garantindo que só exite um local onde os dados da aplicação estão localizados impedindo a duplicação e o acesso a dados inválidos por outras partes da aplicação.

11 – Testes unitários

Os testes unitários ficaram a cargo das classes testes implementadas no pacote lp.projeto.musicplayer.tests

12 – Extras Implementados:

- Barra de execução da música. Permitindo ao usuário avançar e voltar na execução da música.

- O uso de *multithreads* para permitir a execução de outras atividades enquanto uma música é reproduzida. (Cadastro de nova música, modificação de playlists, etc).

Detalhe para o item 4 dos requisitos funcionais: *4 – Buscar músicas cadastradas*

Foi utilizada uma Trie de prefixos em vez de uma Árvore patrícia de sufixos, pois o funcionamento de uma Trie é mais simples, e mesmo ocupando mais memória, cumpre seu objetivo muito bem.

Durante a pesquisa deve ser digitado o nome exato da música a ser buscado a Trie implementada diferencia maiúsculas de minúsculas e realiza a pesquisa letra a letra desde o começo do nome cadastrado. Portanto não é possível pesquisar pelo segundo nome de uma música e obter nos resultados a música esperada.

Enquanto o nome da música é digitado são mostradas as músicas que podem estar sendo buscadas, como se fosse um autocomplete.

Ao ativar o botão da pesquisa é criada uma playlist temporária para mostrar as músicas que condizem com a pesquisa. Sendo essas músicas mostradas no campo de playlists.

Referências

http://code.makery.ch/library/javafx-8-tutorial/

https://docs.oracle.com/javase/10/docs/api/overview-summary.html#JavaFX

https://medium.com/basecs/trying-to-understand-tries-3ec6bede0014

https://www.jetbrains.com/help/idea/developing-a-javafx-application-examples.html

http://www.java2s.com/Code/Java/JavaFX/Slidervaluepropertychangelistener.htm

https://docs.oracle.com/javase/8/javafx/api/javafx/stage/DirectoryChooser.html

http://www.java2s.com/Tutorial/Java/0460__Design-Pattern/Catalog0460__Design-Pattern.htm

https://github.com/kamranahmedse/design-patterns-for-humans