nUNIVERSIDADE DO VALE DO RIO DOS SINOS

EXATAS E TECNOLÓGICAS – 6

ANÁLISE DE SISTEMAS E CIÊNCIA DA COMPUTAÇÃO

Disciplina: Laboratório II Semestre: 2012/2

Prof.: Luciano Cavalheiro da Silva

Trabalho do Grau A (peso 30% do GA)

JukeBox

O tema do TGA é o desenvolvimento de uma aplicação que emula, de forma aproximada, o funcionamento de uma JukeBox (ver http://en.wikipedia.org/wiki/Jukebox), e similares), conforme caracterizado pelos requisitos apresentados a seguir.

Requisitos Funcionais da Aplicação

1. A aplicação possui uma lista de execução (playlist) e uma biblioteca de músicas disponíveis (library).
2. Cada música é descrita por, pelo menos, três atributos: o título da música, o cantor ou conjunto/banda, e o tempo de execução (em segundos).
3. A qualquer momento, novas músicas podem ser carregadas de arquivos texto e incorporadas na lista global de músicas. Dentro do arquivo texto, as músicas estarão representadas uma por linha, sendo seus atributos (titulo, cantor e duração) separados por ';'. Por simplicidade, não é necessário tratar o caso de duplicatas.
4. A qualquer momento, novas músicas podem também ser inseridas diretamente a partir de informações cadastradas pelo usuário.
5. Ao ser inserida na biblioteca de músicas, a música recebe um ID sequencial único, que é definido por sua posição da lista de músicas da biblioteca.
6. A playlist é montada a partir das músicas disponíveis na biblioteca de músicas.
7. As músicas nunca são removidas da biblioteca global, mas podem ser removidas ou reordenadas dentro da playlist.
8. No encerramento da aplicação, a lista global atual é salva (serializada) para um arquivo binário denominado jb-library.db, o qual é novamente carregado, caso exista no momento da inicialização da aplicação.
9. No encerramento da aplicação, a playlist atual é salva para um arquivo texto denominado jb-playlist.txt, o qual também é novamente carregado, caso exista no momento da inicialização da aplicação. Neste arquivo são registrados apenas os IDs das músicas que estão na playlist.

A interface de usuário da aplicação deve disponibilizar ao menos as seguintes opções:

1. **Adicionar uma Musica** -- deve solicitar ao usuário a informação dos dados da música.
2. **Adicionar músicas de Arquivo** -- deve solicitar ao usuário a informação do nome de arquivo a ser carregado
3. **Listar músicas disponíveis na biblioteca de músicas** -- ao lado de cada música, deve ser apresentado seu ID
4. **Adicionar Música à Lista de Execução** -- deve solicitar ao usuário o ID da música a ser adicionada
5. **Remover Música da Lista de Execução** -- deve solicitar ao usuário o ID da música a ser adicionada
6. **Tocar Lista de execução** -- (simulado) deve sucessivamente imprimir os das músicas, removendo cada música executada da playlist.
7. **Listar músicas atualmente na Lista de Execução** -- ao lado de cada música, deve ser apresentado seu ID
8. **Mover música para posição anterior na Lista de Execução** -- deve solicitar ao usuário o ID da musica a ser movida
9. **Mover música para posição posterior na Lista de Execução** -- deve solicitar ao usuário o ID da musica a ser movida
10. **Sair** -- deve salvar as informações da biblioteca e lista de execução para os arquivos apropriados e finalizar a aplicação.

Restrições de Modelagem

RM1) A aplicação deve ser composta pelos seguintes elementos principais:

- **Music** – representa uma música individual, com seus quatro atributos.
- **JukeBoxUI** – trata das questões de interação com o usuário (apresentação do menu da aplicação, entrada e valição de dados e apresentação de mensagens de erro) e aciona os métodos de alguma implementação de JukeBox para cada operação selecionada.
- **JukeBox –** agrupa a biblioteca de músicas e lista de execução e implementa as restrições de inserção na lista de execução; Conhece objetos Music, porém nada sabe sobre interação com o usuário. Coordena a execução das operações de geração em arquivo.
- **PersistentObject** – "marker interface", indica que a classe sabe salvar (persistir seu estado) em um arquivo.
- **MusicList** – define e implementa as operações comuns a qualquer lista de músicas.
- **MusicLibrary** – representa a lista global de músicas disponíveis para montagem da lista de execução.
- **PlayList** – define a lista particular de execução que representa a seleção realizada pelo usuário.

RM2) As funcionalidades comuns a todas listas de música (insert, remove, get, set, size, isEmpty) devem ser fatoradas em uma classe **MusicList** e herdadas pelas classes apropriadas para evitar duplicação de código. A implementação do armazenamento deve utilizar arrays e oferecer acesso aos elementos via métodos get(int pos). Em particular, as classes ArrayList e Vector não devem ser utilizadas.

RM3) Deve ser definida uma interface denominada **PersistentObject**, a qual deve incluir dois métodos **loadFromFile(File f)** e **saveToFile(File f)** e deve ser implementada pelas classes cujo estado deve ser salvo e carregado de arquivos, PlayList e MusicLibrary.

RM4) Não há limite no número de músicas cadastradas na biblioteca ou na lista de execução, ou seja, o programa deve redimensionar dinamicamente suas estruturas de armazenamento para acomodar novos itens se necessário.

RM5) A classe **JukeBoxUI** deve tratar de todas as questões de interação com o usuário e validação das informações fornecidas por este. Após a validação ela aciona métodos da classe JukeBox para realização efetiva da operação.

RM6) Para reduzir a dependência/acoplamento entre classes, tanto a PlayList quanto a MusicLibrary, se necessário, podem conhecer a **JukeBox** a qual estão vinculadas, porém uma não conhece nem tem acesso a outra diretamente.

RM7) Não deve haver qualquer tipo de duplicação de código dentro de uma classe ou entre classes. Utilize herança e métodos auxiliares para atingir esse efeito. Duplicações detectadas serão penalizadas com descontos parciais ou integrais na avaliação das classes onde ocorrem.

RM8) Todos os métodos de todas as classes deverão estar documentados através de um bloco javadoc fora do método, o qual deve explicar o propósito geral da operação e seus parâmetros. Além disso, dentro de cada método, todos os statements (comandos) de seleção ou repetição deverão estar justificados no programa, ou seja, deve incluir antes de cada linha um comentário // explicando o propósito da próxima instrução dentro do objetivo daquele método.

Dica: sugere-se fortemente que a implementação de qualquer método inicie sempre pela documentação do mesmo (javadoc + documentação interna/algoritmo) e somente na sequencia se passe para a implementação.

Informações Sobre a Entrega

- A atividade poderá ser desenvolvida individualmente ou em duplas
- Um arquivo .zip contendo o código fonte das classes desenvolvidas, devidamente formatado e documentado, com comentários complementares ao código, deverá ser entregue através do moodle dentro do prazo especificado. **Observar as datas das entregas parcial e final definidas no moodle**.
- Para ser útil, o comentário incluído deve esclarecer o propósito do comando, no contexto da lógica (algoritmo do script) e não ser uma simples tradução literal das palavras.
- O programa deverá ser apresentado ao professor, na data fixada no moodle, com presença e participação de todos os membros do grupo, os quais deverão estar aptos a responder questões sobre quaisquer partes da aplicação.

AVISO IMPORTANTE:

É **permitido** e **benéfico** discutir e **trocar ideias** com os colegas ou mesmo utilizar os fóruns para resolver dúvidas. Porém, **em nenhuma circunstância cópias serão admitidas**, mesmo que parciais, sendo penalizadas com a nota zero a todos os envolvidos. Se emprestar diretamente a sua solução a um colega, tenha isso em mente.

Bom Trabalho!