Projeto

Rockstar Inc.

Introdução à Programação em Java – AoR
Departamento de Engenharia Informática

1 2 9 0
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Objetivos:

- Criação de aplicações em Java com interfaces gráficas e armazenamento de dados

Requisitos:

- Plataforma Java e um editor de código
- Resolução das fichas práticas 1-13

Material de Apoio:

- Slides das aulas teóricas

Datas importantes:

Entrega intermédia - 11 de Dezembro, 23h59
Apresentação Intermédia – 12 de Dezembro, 9h30-12h30
Entrega final – 8 de Janeiro, 23:59
Defesa Final – 10 de Janeiro

Descrição

A empresa Rockstar Inc. criou um negócio que consiste numa plataforma para músicos disponibilizarem as suas criações ao público. Assim, é necessário construir uma *aplicação standalone* que permita que clientes possam, por exemplo, pesquisar e adicionar músicas às suas playlists, e que permita que músicos individuais possam desempenhar tarefas como adicionar ou remover músicas do sistema. Segue-se uma breve descrição das funcionalidades a implementar.

Cliente:

– Registar cliente. Requer username e password.
– Login (o username do cliente deve ser sempre visível na aplicação após o login) / Logout.
– Listar todas as músicas e respetivos detalhes. Deve ser possível ordenar especificando um atributo (de um conjunto fixo de dois dos atributos da música) a usar para a ordem. Deve ser possível também aplicar os dois tipos de ordem aos resultados. Por exemplo: ordenar por nome de música ascendente; ordenar por data descendente.
– Pesquisa: deve ser possível pesquisar músicas por um de dois critérios à sua escolha.
– Criar playlist vazia.
– Adicionar uma música a uma playlist.
– Criar playlist especificando um género musical e o número de músicas a incluir na playlist. As músicas são selecionadas aleatoriamente, dentro do género especificado.
– Ver as suas playlists.
– Remover playlist.
– Definir a visibilidade de uma playlist como sendo *pública* (visível para todos os utilizadores) ou *privada* (visível apenas para o utilizador).

– Os clientes podem adicionar um rating numérico (e.g., de 0 a 10) a uma certa música.
– Algumas músicas não são grátis, estando apenas disponíveis para serem adicionadas a playlists, após a compra das mesmas ser efetuada. Quando o cliente efetua uma compra (que pode incluir várias músicas) devem ser registados os detalhes respetivos.
– Adicionar uma música ao carrinho de compras do cliente.
– Finalizar compras. O cliente compra todos os itens no seu carrinho, gastando o seu saldo. Deve existir algum mecanismo que impeça a compra, se o saldo não for suficiente.
– Alterar saldo.

Músico:

– Registar músico. Requer username, password e um código PIN numérico.
– Login (o username do músico deve ser sempre visível na aplicação após o login) / Logout.
– Adicionar uma nova música. A música pode fazer parte de um álbum.
– Corrigir o título de uma música.
– Alterar o preço de uma música. Devem ser registadas as alterações de preço ao longo do tempo e permitir que os clientes as possam ver quando veem os detalhes de uma música.
– Inativar uma música (não deve ser possível criar novas playlists com esta música, mas a mesma deve manter-se em playlists já existentes).
– Listar/Pesquisar as suas músicas (de forma semelhante ao que foi descrito para o cliente, porém considera apenas as músicas do próprio).
– Ver estatísticas: total de utilizadores, total de músicas, valor total das músicas no sistema, valor total das vendas, total de álbuns por género musical. Para além destes valores, apresente mais dois valores à sua escolha, que considere úteis num sistema deste tipo.

Para grupos de 3 alunos:

– Deve ser possível remover uma música, o que resulta também na remoção de todas as suas ocorrências em playlists. O saldo respetivo é devolvido ao utilizador.
– Existem **administradores** de sistema que podem criar outros administradores. Podem também inativar (e reativar) outros utilizadores, incluindo outros administradores (a aplicação tem sempre um administrador definido por omissão que não pode ser inativado). A um utilizador inativo não é permitida autenticação no sistema.
– Os administradores ativos podem criar campanhas de descontos. Cada campanha tem uma data de início e fim e um certo número total de cupões. Cada cupão pode ser associado a uma compra, desde que a campanha ainda esteja a decorrer, e confere um desconto à compra em questão. O valor do desconto a aplicar é uma percentagem, definida ao nível da campanha, que se aplica ao valor total da compra.
– O **cliente,** antes de finalizar uma compra, pode ver as campanhas que estão a decorrer no momento e que ainda têm cupões. O cliente seleciona a campanha do seu interesse e finaliza a compra o que faz com que um dos cupões seja utilizado (quando os cupões esgotarem, na prática a campanha termina). No máximo, cada cliente pode usufruir de um cupão por campanha.

Desenvolvimento do projeto:

O trabalho deve ser realizado em grupos de **2 elementos** (com a exceção de 1 grupo de 3 elementos em turmas com número ímpar de alunos). A aplicação a desenvolver deverá ter em conta os seguintes aspetos:

1. Ao entrar na aplicação, a mesma deve ler todos os dados resultantes da última utilização;
2. Cada classe deve gerir internamente os seus dados, pelo que deverá cuidar da proteção das suas variáveis e métodos;
3. Devem ser usados ficheiros de objetos para guardar os dados relativos às classes criadas;
4. Não deverão ser criados getters ou setters desnecessários.
5. Cada objeto deverá ser responsável por tarefas ou objetivos específicos, não lhe devendo ser atribuídas operações indevidas;
6. Utilize as *keywords static* e *break* apenas quando tal se justifique e não para contornar possíveis erros no código ou problemas na execução.

Elabore um diagrama de classes (em UML) antes de iniciar a implementação.

Tenha ainda em conta os seguintes pontos que serão importantes na **avaliação**:

1. Comente as classes, métodos e/ou variáveis segundo o formato Javadoc. Isto permitir-lhe-á gerar automaticamente uma estrutura de ficheiros HTML, descritivos do seu código, que deve incluir no seu relatório.
2. Comente o restante código sempre que a leitura dos algoritmos não seja óbvia;
3. Idente devidamente o seu código.
4. Tal como sugerido acima, evitar o uso de *static*, *break* e de variáveis e métodos *public*, estes últimos quando desnecessário.
5. Na escolha de nomes para variáveis, classes e métodos, seguir as convenções adotadas na linguagem Java.
6. Na organização das classes deverá evitar criar estruturas que mantenham dados redundantes.
7. Teste devidamente o seu programa.

Prazos de Entrega:

A entrega do trabalho compreende duas entregas distintas:

1. **Entrega Intermédia**: Entrega até **11 de dezembro (23:59)** – Os alunos devem entregar um relatório PDF através do InforEstudante que deve incluir os diagramas de classe UML e o *design* inicial da interface gráfica da aplicação. Terão um espaço breve de discussão, sem avaliação quantitativa, sobre o seu plano/diagrama na **manhã do dia 12 de Dezembro** (com inscrição uns dias antes no Inforestudante).
2. **Entrega Final**: Entrega até **8 de janeiro (23:59)** – Os alunos devem submeter um ficheiro .zip através do InforEstudante com a versão final da aplicação e que inclui:
    1. Código fonte de todas as classes Java;
    2. Ficheiros executáveis;
    3. Ficheiros para teste da aplicação com valores iniciais;

4. Ficheiros Javadoc;
5. Um relatório em PDF com os seguintes elementos:
    i. Descrição geral da estrutura da aplicação;
    ii. Diagrama de classes;
    iii. Descrição das principais estruturas de dados e dos ficheiros usados;
    iv. Breve explicação de como é que as principais partes da aplicação estão implementadas e interligadas;
    v. *Design* da interface gráfica da aplicação;

Avaliação do trabalho:

Para a avaliação do trabalho contam fatores de dois tipos:

1. Caixa preta (do ponto de vista externo, e.g., como é percecionado pelo utilizador):
    1. Conjunto de funcionalidades implementadas;
    2. Robustez do programa;
    3. Qualidade da interface.
2. Caixa branca (i.e., com conhecimento das estruturas internas do programa):
    1. Qualidade geral das soluções técnicas encontradas para os problemas em causa;
    2. Estruturação do código;
    3. Qualidade dos comentários.

Não se aceitam trabalhos submetidos com erros de compilação e que não estejam minimamente estruturados do ponto de vista da Programação Orientada a Objetos. Note ainda que os trabalhos serão comparados (entre si, com trabalhos de anos anteriores e também com código disponível na Internet), para deteção de eventuais fraudes por cópia. Nos casos de cópia de trabalho total ou parcial, os grupos envolvidos terão os projetos anulados, reprovando à disciplina.

A avaliação inclui ainda uma **defesa** presencial onde cada trabalho é discutido com cada aluno individualmente. Tanto esta defesa como a classificação final é feita de forma individual. Para isso, cada grupo deve inscrever-se num horário que esteja disponível para essa defesa no InforEstudante. As defesas finais irão ocorrer no dia **10 de janeiro**, sendo que os *slots* para inscrição nas defesas serão disponibilizados alguns dias antes desta data.