MAC0321 – Exercício Programa em Duplas – Biblioteca

Marcelo Finger

30 de junho de 2021

Instruções

Nome do projeto java: EP2-NUSPs. Sendo NUSPs a sequência de tamanho no máximo 2 dos Números USP dos membros da equipe.
Arquivo de envio: Um ÚNICO projeto java zipado, ou seja, enviar o arquivo EP2-NUSPs zipado.

Seguir as seguintes instruções, sob pena de desconto de nota por código confuso:

- Esse EP pode ser feito em duplas. Neste caso: (i) inserir ambos os NUSPS no nome do projeto; e (ii) cada membro da dupla deve submeter uma cópia idêntica do projeto no eDisciplinas.
- Não enviar arquivos soltos.
- Não enviar mais de um projeto java.
- Não enviar um projeto com muitas subpastas fazendo com que dê trabalho para encontrar o exercício.
- Não enviar rascunhos dos exercícios. Envie apenas os arquivos que vocês efetivamente querem que sejam corrigidos para que não corram o risco de corrigirmos os arquivos errados.
- Colocar cada exercício dentro de um pacote diferente (o pacote fica dentro do projeto java).
- Separar a classes funcionais das classes de testes deixando-as em arquivos diferentes.

Introdução

Finalmente temos um cliente cujo negócio fomos contratados para modelar e implementar. Finalmente temos um cliente cujo negócio fomos contratados para modelar. Vamos modelar e implementar um sistema que gerencia os empréstimos de uma biblioteca, armazenando informações em planilhas e lendo informações destas planilhas. Nesta fase do projeto vamos utilizar apenas planilhas no formato CSV (*comma separated values*, ou valores separados por vírgulas). Esse sistema se restringe apenas a uma parte do acervo físico da biblioteca, ou seja, aos livros em papel.

Esse sistema se restringe apenas a uma parte do acervo físico da biblioteca, ou seja, aos livros em papel. Os elementos principais do sistema resultantes da modelagem são os seguintes.

Clientes que são identificados pelo seu CPF e possuem um nome.

Livros que se referem aos títulos, suas edições, lista de autores, a língua em que foram escritos e demais detalhes. Os livros são identificados por um dente ficador exclusivo da biblioteca IdLivro. Para essa tarefa não usamos o ISBN do livro, pois algumas edições antigas não possuem tal informação.

Exemplares que se referem às unidades físicas do livro. um livro pode ter de 0 a um número qualquer de exemplares disponíveis na biblioteca. cada exemplar é identificado com o identificador único, IdExemplar.

Empréstimos que relacionam um exemplar com o cliente, registrando a data do empréstimo, a da máxima do retorno, e uma data em que o livro foi efetivamente retornado. Este último campo permanece vazio enquanto o livro não foi retornado.

1 Leitura dos dados

Os dados devem ser lidos de arquivo CSV Para isso você pode utilizar o método de BufferedReader, ou pode usar diretamente um pacote como OpenCSV, a qual se encontra disponível no sourceforge (recomendado), com documentação sobre utilização, download, classes e funcionalidades disponíveis também no sourceforge.

Imaginando que alguma hora o método de armazenagem pode migrar para um meio mais profissional como um banco de dados, vamos implementar a leitura dos dados através de uma interface que implementa o padrão DAO. considere para tanto a seguinte interface `LeitorBibliotecaDAO.java`:

```
1 package br.usp.ime.mac321.ep2;
2 import java.util.List;
3
4 public interface LeitorBibliotecaDAO {
5     public List<Livro>    getAllLivros(String    nomeArquivoLivros);
6     public List<Exemplar> getAllExemplares(String nomeArquivoExemplares);
7     public List<Usuario>  getAllUsuarios(String  nomeArquivoUsuarios);
8     public List<Emprestimo> getAllEmprestimos(String nomeArquivoEmprestimos);
9 }
```

Essa interface é fornecida no projeto em anexo, bem como as cascas das classes mencionadas nela. A interface não deve ser alterada, mas as demais classes devem ser alteradas. Implementar uma classe que implementa esta interface de leitura.

A leitura dos dados deve jogar exceções no seguintes casos:

- Arquivo não encontrado.
- Dois livros possuem o mesmo valor do campo de identificação.
- Um exemplar de livro foi lido contendo uma referência a um livro que não existe.
- Um empréstimo foi feito envolvendo um cliente que não existe.
- Um empréstimo foi feito envolvendo um exemplar que não existe.

Notem que estas regras implicam que os arquivos devem ser lidos em uma determinada ordem. Junto com o projeto fornecemos planilhas para testar a leitura correta e a leitura com problemas dos arquivos csv. Também fornecemos um arquivo de teste com alguns testes para serem usados com os arquivos que fornecemos, `TestaLeitorBiblioteca.java`. Nessa classe de teste JUNIT, há testes de detecção de erro de entrada que devem ser completados por vocês.

2 Escrita dos dados

Implementar uma interface no padrão DAO que contém métodos de escrita para cada um dos arquivos lidos no item anterior. Não implemente nenhuma classe para esta interface ainda.

Pesquisar na internet sobre o princípio de Segregação de Interfaces. Este princípio faz parte de uma série de boas práticas de desenvolvimento de software chamado de desenvolvimento SOLID. Pesquise na internet sobre este conjunto de boas práticas e tente aplicá-lo no desenvolvimento deste programa (e dos demais programas que você vier a escrever na sua vida). Note que a segregação de interfaces consiste no item I do acrônimo SOLID.

Baseado no princípio da Segregação de Interfaces, refatorar a interface única de escrita acima em diversas interfaces. Implemente as funcionalidades de armazenamento por meio de delegação de parte importante desta competência para cada uma das classes cuja casca foi fornecida. Gere uma nova classe de testes, em que arquivos CSV são lidos e escritos; seu teste deve verificar que os arquivos são idênticos ou ao menos correspondentes.

3 Estado dos Exemplares

Pesquise na internet sobre o padrão de desenvolvimento de software comportamental chamado State (Estado)[1]. Vamos agora usar este padrão para lidar com os exemplares.

Um exemplar pode estar em um dos seguintes estados:

Disponível quando o exemplar existe e não está emprestado.

Emprestado quando há algum empréstimo envolvendo um usuário ativo e o exemplar, e este empréstimo não registra ainda a devolução do exemplar. Um exemplar disponível torna-se emprestado quando a um empréstimo válido. Um exemplar emprestado retorna ao estado disponível quando o exemplar é retornado e tal informação é anotada no empréstimo.

Extraviado quando um livro está emprestado por mais de um mês além da data de devolução. o exemplar volta a ser disponível se for retornado, e volta a ser emprestado Se houver uma prorrogação na data de devolução.

Implemente a classe exemplar e empréstimo de forma que seja capaz de informar o estado de um exemplar a cada instante. Apresente uma classe de testes em que o mesmo livro passa por todos os estados descritos acima e por todas as transições possíveis.

4 Funcionalidades da Biblioteca

A biblioteca deve possuir as seguintes funcionalidades:

- Realizar um empréstimo de um exemplar a um cliente, com data de entrega determinada (padrão: um mês).
- Buscar os exemplares disponíveis de um livro pelo seu título, retornando uma lista de exemplares encontrados; note que isso inclui a lista vazia, quando todos os exemplares estiverem emprestados ou extraviados. Ou nulo caso o livro não conste do catálogo.
- Buscar os livros que possuem um determinado autor. Note que este autor pode ser o único autor ou um co-autor do livro.
- Listar os exemplares extraviados de um livro, a partir do IdLivro.
- Listar os empréstimos de todos os exemplares de um determinado livro, a partir do título do livro.
- Listar os empréstimos vigentes de um determinado cliente, a partir de seu CPF

Implementar a classe Biblioteca, que recebe estas solicitações e entrega a resposta.

Para comprovar se a sua implementação está cumprindo os requisitos acima, apresentar testes de unidade no formato JUnit que comprovem a adequação da implementação. Para estes testes, novas planilhas podem ser preparadas com os dados de livros e exemplares.

[1] O exercício do EP1 é uma implementação sofisticada de um simulador que implementa este padrão, mas como aqui não há necessidade de simular nada, a sua implementação pode ser bem mais simples