MATA62 – ENGENHARIA DE SOFTWARE I

Trabalho Prático

1. Objetivo

Neste trabalho, o aluno deve projetar e implementar um programa simples. O objetivo é permitir que os alunos apliquem seus conhecimentos em projeto e programação orientados a objetos, incluindo princípios de projeto e padrões de projeto.

As Seções 2 e 3 descrevem os requisitos do sistema. A Seção 4 lista algumas exigências do projeto. A Seção 5 explica os critérios de avaliação. A Seção 6 descreve a entrega do trabalho. E por fim, a Seção 7 lista os dados de teste que devem ser usados na execução do sistema.

2. Visão Geral do Sistema

O sistema de biblioteca consiste no gerenciamento e manutenção de livros disponíveis em uma biblioteca acadêmica. Ele permite que três tipos de usuários (alunos de graduação, alunos de pós-graduação e professores) realizem o empréstimo, devolução e reserva de livros disponíveis.

Um livro específico pode ter mais de um exemplar disponível na biblioteca. Assim, é possível encontrar dois ou mais exemplares de um mesmo livro.

Cada livro deve possuir um código de identificação e um título. Além do código e título, os livros devem incluir informações adicionais, como editora, autores, edição e ano de publicação.

Cada usuário, por sua vez, deve ter um código de identificação e um nome. Os três tipos de usuários (alunos de graduação, alunos de pós-graduação e professores) possuem regras específicas para o empréstimo de livros, que estão detalhadas na Seção 3 deste documento. Cada tipo de usuário também tem um período específico, em dias, durante o qual pode manter o livro emprestado, conforme indicado na Tabela 1. Sempre que o empréstimo de um livro é solicitado, essa operação é registrada no sistema e é registrada também a data de devolução com base no prazo de empréstimo correspondente ao tipo de usuário.

Tipo de Usuário	Tempo de Empréstimo
Aluno Graduação	4 dias
Aluno Pós-Graduação	5 dias
Professor	8 dias

Tabela 1. Tempo de empréstimo por tipo de usuário.

Os usuários também podem realizar reserva dos livros. A reserva garante a prioridade no empréstimo apenas entre os alunos. A reserva também deve ser registrada no sistema, incluindo a data em que a reserva foi feita.

3. Funcionalidades

3.1. Empréstimo

A funcionalidade principal do programa é permitir o empréstimo dos livros. Para realizar um empréstimo, o usuário deverá informar o comando "emp" seguido pelo código do usuário e pelo código do livro, separados por um espaço. Ex: "*emp 123 100*". Caso o usuário tenha feito uma reserva para o livro em questão, a mesma deve ser cancelada e o empréstimo será efetivado.

3.1.1. Regras de Empréstimo

Por enquanto, temos duas regras de empréstimos no sistema. **No entanto, no futuro, se novos tipos de usuários forem incluídos no sistema, novas regras podem ser criadas.** Portanto, o projeto deve estar preparado para que novas regras possam ser incluídas de forma mais facilitada, alterando o mínimo o código fonte já existente nas classes do programa.

Regra 1: Empréstimo para Alunos

O empréstimo de um livro será realizado para um aluno de graduação ou de pós-graduação apenas se:

1. Houver exemplares disponíveis na biblioteca;
2. O usuário não estiver "devedor" com livros em atraso;
3. O usuário seguir as regras específicas referentes à quantidade máxima de livros que podem ser emprestados (conforme Tabela 2);
4. A quantidade de reservas existentes do livro deve ser menor do que a quantidade de exemplares disponíveis, desde que o usuário não tenha uma reserva para esse livro;
5. Se a quantidade de reservas for igual ou superior à de exemplares disponíveis, o empréstimo será permitido apenas se uma das reservas for do usuário;
6. O usuário não pode ter nenhum empréstimo em andamento de um exemplar desse mesmo livro.

Tipo de Usuário	Limite de Empréstimos em Aberto
Aluno Graduação	2 livros
Aluno Pós- Graduação	3 livros

Tabela 2. Limites da quantidade de livros para empréstimo.

Regra 2: Empréstimo para Professor

O empréstimo do livro só será concretizado para um **professor** se:

1. Houver a disponibilidade de algum exemplar daquele livro na biblioteca; e
2. O usuário não estiver como "devedor" de um livro em atraso.

Observe que os professores não terão o empréstimo negado, mesmo que haja reservas para o livro em questão, e não há limite para a quantidade de livros que eles podem pegar emprestado.

3.1.2. Mensagens de Insucesso do Empréstimos

Se a regra de empréstimo não for atendida e o não for possível realizar um empréstimo, o programa deve mostrar no console uma mensagem informando o motivo. Por exemplo, "Não foi possível realizar o empréstimo, pois o usuário já tem um exemplar deste mesmo livro em empréstimo no momento."

3.2. Devolução

O sistema deve permitir a devolução de livros. Durante a devolução, o usuário deve digitar o comando "*dev*" seguido do código de identificação do usuário e do código de identificação do livro emprestado.

3.3. Reserva

O sistema deve permitir a reserva de um livro. Durante esse processo de reserva, o usuário deve digitar o comando "*res*", o código de identificação do usuário e o código de identificação do livro que o usuário deseja reservar. **O sistema deve registrar a reserva com a data em que ela foi realizada.**

3.4. Registro de Observação de Livros

O sistema deve permitir que professores se registrem para receber notificações sempre que um livro atingir mais de duas reservas simultâneas. Ao se registrar como "observador" de um livro específico, o professor será notificado toda vez que esse livro tiver mais de duas reservas ao mesmo tempo. Internamente, o sistema deve registrar quantas notificações foram recebidas pelo observador. **Basta um contador de notificações único por observador, independentemente se ele observa mais de um livro.** Para registrar um professor como observador de um livro, o usuário deve digitar o comando "*obs*" seguido do código do usuário e do código do livro. Para simplificar a implementação, o programa não precisa verificar se o código do usuário informado realmente pertence a um professor. No futuro, o sistema pode ser evoluído de forma que permita outros tipos de usuários, por exemplo, coordenadores, que também possam observar a reserva de livros. Implemente essa funcionalidade usando um padrão que permita facilmente essa evolução.

3.5. Consulta de Informações de Livro

Dado o código de um livro, o sistema deve apresentar suas informações da seguinte forma: (i) título, (ii) quantidade de reservas para aquele livro, e, se diferente de zero, devem ser também apresentados o nome dos usuários que realizaram cada reserva, (iii) para cada exemplar do livro, devem ser apresentados seu código, seu status (disponível ou emprestado). Caso o exemplar esteja emprestado, deverá ser exibido o nome do usuário que realizou o empréstimo, a data de empréstimo e a data prevista para devolução. Atenção: essa consulta mostra apenas os dados dos empréstimos correntes do livro. Para solicitar tal consulta, o usuário deverá digitar o comando "*liv*", seguido do código do livro.

3.6. Consulta de Informações de Usuário

Dado um usuário, o sistema deverá apresentar a lista de todos os seus empréstimos correntes **e passados**, assim como de suas reservas. A listagem de cada empréstimo deverá apresentar o título do livro, a data do empréstimo, o status atual do empréstimo (em curso ou finalizado) e a data da devolução realizada ou prevista. A listagem das reservas deverá apresentar o título do livro reservado e a data da solicitação da reserva. Para solicitar tal consulta, o usuário deverá digitar o comando "*usu*", seguido do código do usuário.

3.7. Consulta de Notificações Recebidas

Dado um professor, o sistema deve retornar o número total de vezes que ele foi notificado sobre a ocorrência de mais de duas reservas simultâneas nos livros que ele está observando. Para solicitar tal consulta, o usuário deverá digitar o comando "*ntf*", seguido do código do usuário. **Não há necessidade de validar se o código digitado se refere realmente ao de um professor.**

6. E por fim, o usuário deve ter a opção de sair do sistema. Para isso, basta digitar o comando "*sai*".

4. Modelo Conceitual

A Figura 1 mostra o diagrama de classes que representa o modelo conceitual do sistema. O modelo conceitual define o relacionamento entre as entidades de negócio do programa. Apenas a classe Repositorio não corresponde a uma entidade de negócio, mas foi colocada no diagrama para facilitar projeto. A implementação do programa deve partir desse modelo conceitual e complementá-lo com outras classes.

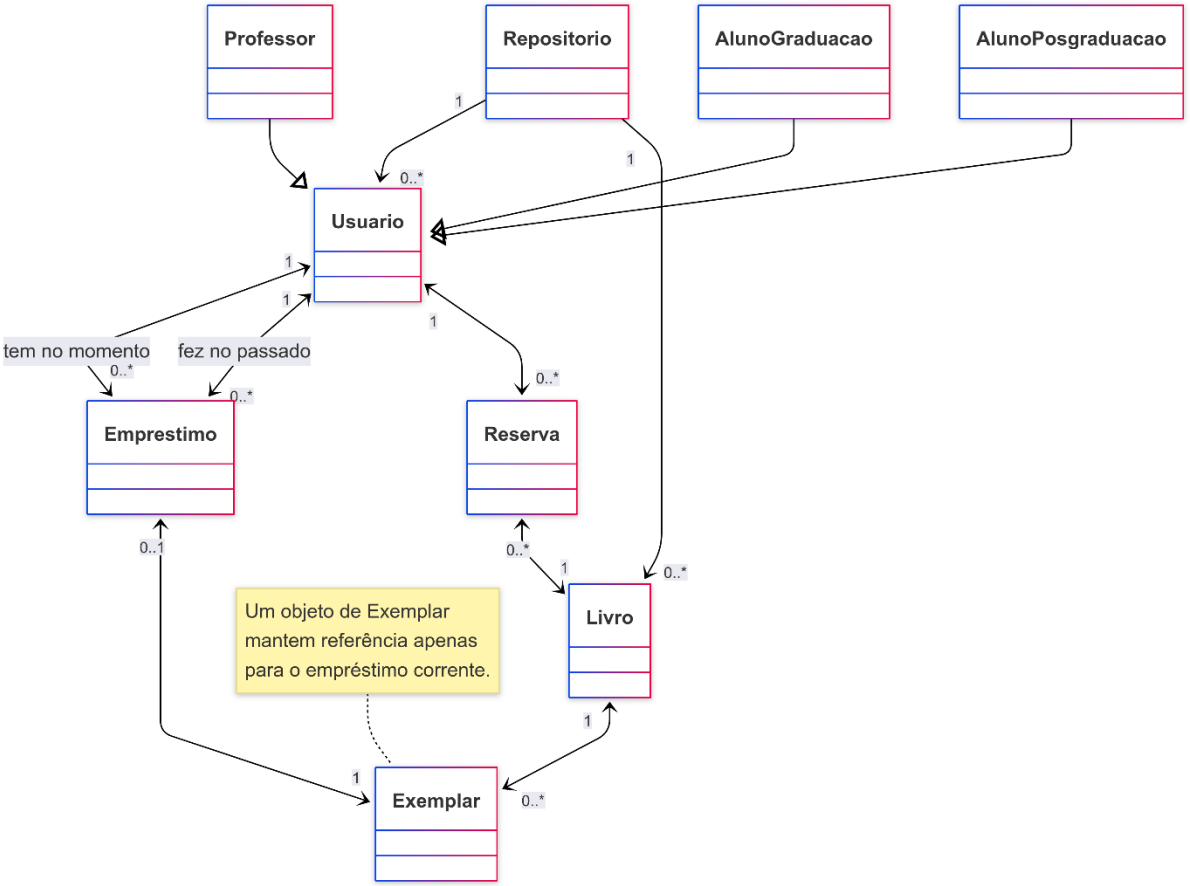

Figura 1. Modelo Conceitual do Programa.

5. Exigências do Projeto

5.1. O sistema NÃO deve se preocupar com a persistência de dados, ou seja, NÃO deve usar banco de dados. Os objetos relativos aos dados de teste (Seção 7) deverão ser instanciados na memória no momento da inicialização do sistema.

5.2. O projeto deve ter uma classe Repositorio que tem apenas a responsabilidade de manter as listas (ou estruturas similares) de usuários e livros existentes no sistema, assim como, ter métodos de busca de usuários e livros (por exemplo, o método buscarUsuarioPorCodigo(codigoUsuario). Essa classe deve ser projetada de acordo com o padrão de projeto Singleton.

5.3. O sistema NÃO deve disponibilizar funcionalidades de cadastros de livros, usuários e exemplares.

5.4. O sistema NÃO deve ter interface gráfica com o usuário. Todos os comandos devem ser fornecidos via linha de comando, e suas respostas, se houver, devem ser mostradas no console.

5.5. O projeto deve ter uma classe responsável por ler os comandos do console e mostrar as respostas no mesmo console.

5.6. A classe do item anterior deve se comunicar com as classes de negócio (inclusive a classe Repositorio) por meio de um esquema de comandos, projetados de acordo com o padrão de projeto "Command".

5.7. Evite o uso de "if "ou "switch" para saber o tipo de usuário que está lidando. **Em particular, implemente, sem uso de condicionais, um padrão de projeto que permita selecionar, a depender do tipo de usuário, qual regra de empréstimo (Regra 1 ou Regra 2) deve ser usada para verificar se o empréstimo pode ser realizado ou não.**

6. Entrega do Trabalho

O trabalho deve ser desenvolvido em uma **linguagem orientada a objetos** e **feito em duplas**.

Além do código fonte, a equipe deverá elaborar o diagrama de classes. O código deverá ser disponibilizado no AVA até o prazo de entrega. O diagrama de classes e o código devem ser apresentados no dia da arguição. O diagrama deve estar legível. Para isso, pode-se optar por quebrá-lo em mais de uma página. Sugestão: Crie uma página que exiba a classe responsável pela interação com o usuário via console, juntamente com as classes que representam os comandos. Em outra página, mostre as classes de negócio do sistema.

Arguição sobre o projeto

Cada dupla irá se reunir remotamente com o professor para mostrar e discutir o diagrama e o código fonte.

7. Critérios de Avaliação

Durante a arguição, o professor fará perguntas individuais a cada membro da dupla. A nota de cada membro pode ser diferente da nota do outro membro, pois será baseada nas respostas dadas durante a sua apresentação.

O projeto será avaliado de acordo com os seguintes critérios:

- Uso coerente dos conceitos de orientação a objetos (herança, polimorfismo, interfaces, associação, etc.);
- Uso de padrões de projeto;
- Conformidade com a descrição do trabalho.

8. Dados de Teste

O sistema NÃO deve se preocupar com a persistência dos dados, ou seja, não deve usar banco de dados. Para testar o sistema, vocês devem criar dados de teste na memória (instanciar) durante a inicialização do programa. Abaixo, são fornecidos alguns exemplos de dados de teste.

Usuários

Código	Tipo Usuário	Nome
123	Aluno Graduação	João da Silva
456	Aluno Pós-Graduação	Luiz Fernando Rodrigues
789	Aluno Graduação	Pedro Paulo
100	Professor	Carlos Lucena

Livros

Código	Título	Editora	Autores	Edição	Ano Publicação
100	Engenharia de Software	Addison Wesley	Ian Sommervile	6ª	2000
101	UML - Guia do Usuário	Campus	Grady Booch, James Rumbaugh, Ivar Jacobson	7ª	2000
200	Code Complete	Microsoft Press	Steve McConnell	2ª	2014
201	Agile Software Development, Principles, Patterns and Practices	Prentice Hall	Robert Martin	1ª	2002
300	Refactoring: Improving the Design of Existing Code	Addison Wesley Professional	Martin Fowler	1ª	1999
301	Software Metrics: A rigorous and Practical Approach	CRC Press	Norman Fenton, James Bieman	3ª	2014
400	Design Patterns: Element of Reusable Object-Oriented Software	Addison Wesley Professional	Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides	1ª	1994
401	UML Distilled: A Brief Guide to the Standard Object Modeling Language	Addison Wesley Professional	Martin Fowler	3ª	2003

Exemplares

Código do Livro	Código Exemplar	Status Exemplar
100	01	Disponível
100	02	Disponível
101	03	Disponível
200	04	Disponível
201	05	Disponível
300	06	Disponível
300	07	Disponível
400	08	Disponível
400	09	Disponível