**MATA62 – ENGENHARIA DE SOFTWARE I**

Trabalho Prático

## 1. Objetivo

Neste trabalho, o aluno projetará e implementar um sistema de porte muito pequeno. O objetivo é permitir que os alunos usem seus conhecimentos em projeto orientado a objetos e programação orientada a objetos. Todos esses conhecimentos já foram estudados pelos alunos nessa disciplina ou em disciplinas anteriores.

As Seções 2 e 3 descrevem os requisitos do sistema. A Seção 4 lista algumas exigências de projeto. A Seção 5 explica os critérios de avaliação. A Seção 6 descreve como será a entrega do trabalho. Finamente, 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 dispor na biblioteca de mais de um exemplar. Assim, é possível encontrar na biblioteca dois ou mais exemplares de um mesmo livro.

Cada livro deve possuir um código que o identifique e um título. Além do código e do título, os livros devem manter as seguintes informações adicionais: editora, autores, edição e ano da publicação.

Cada usuário deve ter um código de identificação e nome. Cada um dos três tipos de usuários possui regras específicas para poder pegar livro emprestado. Essas regras são detalhadas na descrição da funcionalidade de empréstimo, na Seção 3 deste documento. Além disso, a cada tipo de usuário é permitido um determinado intervalo de tempo, em dias, durante o qual ele pode ficar com o livro emprestado, conforme a Tabela 1. Sempre que o empréstimo de um livro é solicitado na biblioteca, é feito o registro daquela operação no sistema e é fixada uma data de devolução baseada no tempo de empréstimo do tipo de usuário.

| **Tipo de Usuário** | **Tempo de Empréstimo** |
|---|---|
| Aluno Graduação | 3 dia |
| Aluno Pós-Graduação | 4 dias |
| Professor | 7 dias |

Tabela 1: Tempo de empréstimo de cada tipo de usuário

Usuários têm também o direito de realizar reservas de livros. A reserva de um livro garante a prioridade no seu empréstimo apenas entre os alunos, como ficará mais claro nas regras de empréstimo, detalhadas na Seção 3. A reserva também tem que ser registrada no sistema.

## 3. Funcionalidades

1. O sistema deve permitir o empréstimo de livros. Durante o empréstimo, o usuário informará o comando “emp” seguido do código do usuário e do código do livro, separados por espaço em branco. Ex.: “emp 123 100”. Caso o usuário tenha uma reserva feita previamente por ele para o dado livro, a reserva deve ser excluída e o empréstimo efetivado. Ao final do procedimento o sistema deve emitir uma mensagem de sucesso ou insucesso, que mencione o nome do usuário e o título do livro. Se for uma mensagem de insucesso, ela deve também mencionar o motivo do insucesso.

   O empréstimo do livro só será concretizado para um **aluno de graduação** ou um **aluno de pós-graduação** se: (i) houver a disponibilidade de algum exemplar daquele livro na biblioteca; (ii) o usuário não estiver “devedor” de um livro em atraso; (iii) forem obedecidas as regras específicas daquele tipo de usuário no que se refere à quantidade máxima de empréstimos, de acordo com a Tabela 2; (iv) a quantidade de reservas existentes do livro for

menor do que a quantidade de exemplares disponíveis, caso o usuário não tenha reserva para ele; (v) a quantidade de reservas for maior ou igual a de exemplares, mas uma das reservas é do usuário; e (vi) o usuário não tiver nenhum empréstimo em curso de um exemplar daquele mesmo livro.

| Tipo de Usuário | Limite de Empréstimos em Aberto |
|---|---|
| Aluno Graduação | 3 livros |
| Aluno Pós-Graduação | 4 livros |

Tabela 2: Limites da quantidade de livros tomados como empréstimo

O empréstimo do livro só será concretizado para um **professor** se: (i) houver a disponibilidade de algum exemplar daquele livro na biblioteca; e (ii) o usuário não estiver "devedor" de um livro em atraso.

Note que os professores não tem empréstimo negado caso haja reservas para aquele livro e não tem limite da quantidade de livros que pode pegar emprestado.

É sabido que nesse tipo de domínio essas regras estão sujeitas a frequentes mudanças. Além disso, podem surgir novos tipos de usuário para os quais as regras de empréstimo sejam diferentes das já existentes para professor e alunos de graduação e de pós.

2. O sistema deve permitir a devolução de um dado livro. 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. Ao final, o sistema deve emitir uma mensagem de sucesso ou insucesso da devolução, que mencione o nome do usuário e o título do livro. A mensagem de insucesso deve dizer o motivo. Nesse caso, o insucesso só ocorre se não houver empréstimo em aberto daquele livro para aquele usuário.
3. 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. Será permitida a reserva de apenas 3 livros por usuário. Ao final, o sistema deve emitir uma mensagem de sucesso ou insucesso da reserva, que mencione o nome do usuário e o título do livro. A mensagem de insucesso deve dizer o motivo.
4. O sistema deve permitir que professores registrem que querem observar toda vez que determinado livro tiver mais de duas reservas simultâneas. O professor se registra como "observador" do livro que desejar. Toda vez que o livro tiver mais de duas reservas simultâneas, o livro deve "avisar" aos "observadores". O observador deve simplesmente registrar internamente quantas vezes ele foi notificado. 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 no futuro. 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. Não há necessidade de checar se o código do usuário se refere realmente a um professor.
5. O sistema deve fornecer as seguintes consultas:
    a. 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, deve ser apresentado seu código, seu status (disponível ou emprestado), e em caso do exemplar estar emprestado deverá ser exibido o nome do usuário que realizou o empréstimo, a data de empréstimo e a data prevista para devolução. Para solicitar tal consulta, o usuário deverá digitar o comando "liv", seguido do código do livro.
    b. 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 daquele empréstimo (em curso ou finalizado) e a data da devolução já 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.

c. Dado um professor, o sistema deve retornar a quantidade de vezes que ele foi notificado sobre mais de duas reservas simultâneas em livros observados por ele. Para solicitar tal consulta, o usuário deverá digitar o comando "ntf", seguido do código do usuário. Não há necessidade de checar se o código se refere realmente a um professor.

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

**4. Exigências de Projeto**

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.

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

3. O projeto deve ter uma classe responsável por ler os comandos e mostrar as respostas no console. Essa classe deve se comunicar com a fachada do sistema por meio de um esquema de comandos, projetados de acordo com o padrão de projeto "Command".

4. A comunicação entre os comandos e as classes de negócio deverão ser feitos por meio de uma classe fachada que também deverá ser um Singleton.

5. Evite o uso de "if "ou "switch" para selecionar entre implementações dos diferentes tipos de usuário. Use polimorfismo para todas as diferenças entre os tipos de usuário. **Em particular, use algum padrão de projeto para implementar as diferentes regras para realização de empréstimo.**

**5. Quanto à Entrega do Trabalho**

O trabalho deve ser desenvolvido em **Java** e **feito em duplas**.

Além do código fonte, a equipe deverá entregar o diagrama de classes. O diagrama deve estar legível. Se for necessário, pode quebrar o diagrama em mais de uma página. Sugestão: uma página mostrando a classe que faz a interação com usuário via console e as classes dos comandos, e outra página mostrando as classes de negócio.

**Devem ser entregues no AVA:**

- PDF do diagrama de classes
- Arquivo Zip do código-fonte;

**Arguição sobre o projeto:**

As equipes irão se reunir individualmente com o professor para mostrar e discutir o diagrama, mostrar o código e executar casos de teste fornecidos no momento da demonstração. A apresentação se dará em um computador do laboratório.

**6. Critérios de Avaliação**

Durante a arguição, o professor fará perguntas individuais a cada membro de cada dupla. A nota do trabalho de um membro de uma dupla pode ser diferente da nota do outro membro da mesma dupla a depender das respostas às perguntas.

O projeto será avaliado de acordo com os seguintes critérios:

1. Modelagem
   (a) Diagrama de classes
      i. Aplicação correta dos conceitos de OO
         A. Herança, Polimorfismo
         B. Associações
      ii. Conformidade com a descrição do trabalho

iii. Uso correto de UML:
   A. Multiplicidade
   B. Navegabilidade

(b) Implementação
   i. Conformidade com o diagrama de classes.
   ii. Todas as classes no código fonte (exceto classes de bibliotecas, como a API Java) estão representadas no diagrama?

(c) Uso de padrões de projeto.

(d) Execução correta de acordo com casos de testes executados pelo professor.

Serão utilizados alguns casos de teste para verificar se o sistema desenvolvido atende aos mesmos. Portanto, é muito importante que os dados de teste (Seção 7) sejam utilizados. Caso os dados de teste não sejam utilizados, os casos de teste falharão e a sua nota será prejudicada.

## 7. Dados de Teste

**O sistema NÃO deve se preocupar com a persistência de dados, ou seja, NÃO deve usar banco de dados. Entretanto, informações suficientes para efetuar os testes devem estar disponíveis (na memória) desde o início da execução da aplicação. A seguir são fornecidos os dados que devem ser carregados no sistema.**

Usuário

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

Livros

| **Código** | **Título** | **Editora** | **Autores** | **Edição** | **Ano Publicação** |
|---|---|---|---|---|---|
| 100 | Engenharia de Software | AddisonWesley | 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: Elements 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 |