Proposta de Trabalho Final

Disciplina de Programação Orientada a Objetos (POO)

Sistema de Gestão de Biblioteca Universitária (SGB)

Outubro de 2025

Tema do Projeto

O trabalho final consiste no desenvolvimento de um **Sistema de Gestão de Biblioteca Universitária (SGB)**, que simule as operações de cadastro, empréstimo e devolução de itens de acervo por diferentes tipos de usuários.

Objetivo Geral

Demonstrar a proficiência na modelagem e implementação de um sistema que utiliza de forma correta e consistente os quatro pilares da POO: Abstração, Encapsulamento, Herança e Polimorfismo.

1. Modelagem de Classes e Atributos

Os alunos deverão implementar as seguintes classes, garantindo que **todos os atributos sejam privados** e acessíveis via *getters* e *setters* (**Encapsulamento**).

1.1. Hierarquia de Usuários (Requisito: Herança)

Classe	Atributos (Exigidos)	Conceito POO
Usuario (Classe Base)	• `id` (int/String) • `nome` (String) • `endereco` (String) • `status` (String, ex: "Ativo", "Bloqueado") • `itensEmprestados` (`List<Emprestimo>`)	**Abstração e Herança**
Aluno (Filha de Usuario)	• `matricula` (String) • `curso` (String) • `limiteEmprestimo` (int, valor: 3)	**Herança**
Professor (Filha de Usuario)	• `siape` (String) • `departamento` (String) • `limiteEmprestimo` (int, valor: 5)	**Herança**

1.2. Hierarquia de Acervo (Requisito: Herança e Polimorfismo)

Classe	Atributos (Exigidos)	Conceito POO
ItemDeAcervo (Classe Abstrata/Base)	• `codigo` (String) • `titulo` (String) • `anoPublicacao` (int) • `isEmprestado` (boolean)	**Abstração e Polimorfismo**
Livro (Filha de ItemDeAcervo)	• `autor` (String) • `isbn` (String) • `edicao` (int)	**Herança**
Revista (Filha de ItemDeAcervo)	• `editora` (String) • `volume` (int) • `issn` (String)	**Herança**

1.3. Classes de Relacionamento e Gerenciamento

Classe	Atributos (Exigidos)	Conceito POO
Emprestimo	• `idEmprestimo` (int/String) • `usuario` (`Usuario`) • `item` (`ItemDeAcervo`) • `dataEmprestimo` (Date) • `dataDevolucaoPrevista` (Date) • `dataDevolucaoReal` (Date) • `multaCobrada` (double)	**Associação/Composição**
SistemaBiblioteca	• `listaUsuarios` (`List<Usuario>`) • `acervo` (`List<ItemDeAcervo>`) • `historicoEmprestimos` (`List<Emprestimo>`)	**Agregação/Composição**

1.4. Métodos Chave (Comportamento)

Os seguintes métodos devem ser implementados para garantir o fluxo de trabalho e o Polimorfismo:

- **Na Classe Usuario (ou subclasses):**
  - `isAptoParaEmprestimo(): boolean` (Verifica limite e se está bloqueado/com multa).
  - `calculaPrazoDevolucao(): Date` (**Polimorfismo**: Sobrescrito em `Aluno` e `Professor`).
  - `adicionarEmprestimo(Emprestimo): void`.
- **Na Classe ItemDeAcervo (ou subclasses):**
  - `emprestar(): void` (Muda `isEmprestado` para true).
  - `devolver(): void` (Muda `isEmprestado` para false).

- **Na Classe Emprestimo:**
  - `calcularMulta(Date dataDevolucaoReal): double`.
  - `finalizarEmprestimo(Date dataDevolucaoReal): void` (Chama o método de cálculo de multa).
- **Na Classe SistemaBiblioteca:**
  - `realizarEmprestimo(String idUsuario, String codItem): Emprestimo` (Contém a lógica central).
  - `realizarDevolucao(String idEmprestimo): void`.
  - `salvarDados(): void - O sistema deverá salvar dados em txt ou csv.`
  - `carregarDados(): void - O sistema deverá carregar os dados de txt ou csv.`

2. Requisitos Mínimos do Sistema (Funcionais e Regras de Negócio)

O sistema deve executar as seguintes funcionalidades, demonstrando a interação entre os objetos modelados na Seção .

2.1. Funcionalidades de Cadastro e Consulta

1. **Cadastro:** Permitir a inclusão de novos usuários (`Aluno`/`Professor`) e itens (`Livro`/`Revista`).
2. **Busca por Acervo:** Implementar a busca de itens pelo Título e/ou ISBN/ISSN.

2.2. Funcionalidades de Empréstimo e Regras de Negócio

1. **Empréstimo de Item (RF1):** Registrar o empréstimo de um item a um usuário, verificando as seguintes regras:
   - **RN1 (Disponibilidade):** O sistema deve **impedir** o empréstimo se o item estiver marcado como indisponível (`isEmprestado` = true).
   - **RN2 (Limite):** O sistema deve **impedir** o empréstimo se o usuário exceder seu limite máximo (`Aluno`=3, `Professor`=5).
2. **Cálculo da Data de Devolução (RF2):** No momento do empréstimo, a data de devolução deve ser calculada de forma polimórfica (**Polimorfismo/Sobrescrita** de método na classe `Usuario`):
   - `Aluno`: Prazo de empréstimo de 7 dias corridos.
   - `Professor`: Prazo de empréstimo de 15 dias corridos.
3. **Devolução de Item (RF3):** Registrar a devolução, atualizando o estado do item para disponível (`isEmprestado` = false).
4. **Cálculo e Controle de Multa (RF4):**
   - **RN3 (Multa):** Se a `dataDevolucaoReal` for posterior à `dataDevolucaoPrevista`, calcular multa de R$ 1,00 por dia de atraso.
   - **RN4 (Bloqueio):** Usuários com multa pendente (multa > 0) ou item em posse com prazo vencido **não podem realizar novos empréstimos**.

2.3. Requisitos de Estrutura de Código

1. **Persistência:** Os dados devem ser salvos e carregados entre as execuções (arquivos de texto simples ou serialização).
2. **Tratamento de Exceções:** Utilizar `try-catch` para lidar com erros de negócio (RN1, RN2, RN4) e erros de entrada de dados.

3. Entrega e Critérios de Avaliação

3.1. Documentação (Entrega Obrigatória)

Os alunos deverão incluir no relatório:

- **Diagrama de Classes UML:** Ilustrando todas as classes, atributos, métodos e os relacionamentos de Herança, Associação e Agregação.
- **Relatório Técnico:** Descrevendo como cada pilar da POO foi aplicado e implementado no código.

3.2. Critérios de Avaliação

Conceito Avaliado	Descrição Esperada	Peso na Nota
Modelagem e Abstração	Estrutura de classes coesa e correta representação das entidades.	20%
Encapsulamento	Uso correto de atributos `private` e métodos `public` de acesso.	20%
Herança	Implementação correta da hierarquia `Usuario` e `ItemDeAcervo`.	20%
Polimorfismo	Aplicação de Sobrescrita (*Override*) para cálculo de prazo de empréstimo (RN2).	20%
Associação/Exceções	Implementação correta dos relacionamentos e tratamento dos erros de negócio (RN1, RN2, RN4).	10%
Funcionalidade	O sistema executa as funcionalidades mínimas propostas sem erros.	10%