

UDF

Trabalho de desenvolvimento de software de gerenciamento de biblioteca.

Aluno: Ricardo Juvencio Oliveira

RGM: 43728651

Conteúdo;

Conteúdo;	2
Introdução	3
Introdução ao desenvolvimento	4
<i>Possibilidades do Sistema:</i>	4
Componentes do Sistema	4
Modelagem UML	6
<i>Operações pensadas inicialmente:</i>	6
Interface do Sistema	7
Estrutura do Projeto e Código-Fonte	8
Aplicação dos Conceitos de POO	10
<i>Encapsulamento:</i>	10
<i>Herança:</i>	10
<i>Polimorfismo:</i>	11
<i>Outros tópicos importantes do projeto:</i>	11

Introdução.

Esse texto aborda informações de desenvolvimento de um programa que gerencia um sistema de biblioteca, desenvolvido em Java podendo trabalhar com usuários, livros e empréstimos como entidades principais, através da orientação a objetos e manipulação de dados, podendo salvar dados de transações de livros, opções de prazo, cadastro de usuários(alunos, professores e funcionários), o programa funciona essencialmente com esses sistemas de gerenciamento através de vários arquivos que serão comentados posteriormente no texto além de uma GUI(interface gráfica de usuário), que roda de forma multi-operacional através da JVM(apesar de ter sido desenvolvimento em um ambiente de desenvolvimento Windows) .

Introdução ao desenvolvimento.

O Objetivo, era desenvolver um programa capaz de gerenciar dados de uma biblioteca, sua função era da melhor forma possível, usando uma manipulação de dados persistentes e orientação a objetos, desenvolver um sistema capaz de armazenar vários tipos de dados como livros e usuários, e em sua ação, gerenciar esses sistemas de empréstimo de livros.

Possibilidades do Sistema:

Através de uma interface gráfica, que tem por trás a lógica em java, será possível, gerenciar livros, usuários e empréstimos, podendo:

1. Registrar novos livros.
2. Registrar novos usuários (sendo eles 3 tipos, professores, funcionários e alunos.)
3. Realizar empréstimos.

Os dados que são guardados em memória em um arquivo no repositório, e que será mais bem abordado no restante desse texto, podendo ser vistos diretamente ao fazer novos registros gerais, e fora as já citadas funções do programa, ele também conta com alguns outros sistemas como.

1. A lógica de devolução de livros.
2. SGB, um sistema geral de busca

Componentes do Sistema.

Esse projeto, é uma aplicação Java completa que implementa um software para gerenciamento de acervo bibliotecário, usuários e empréstimos, o sistema utiliza interface gráfica Swing para interação com o usuário e persistência de dados através de serialização de objetos Java.

O início do desenvolvimento seria a criação de estruturas de formulários, que além de guardar os dados, os manipula, retornando através de métodos esses dados posteriormente, o

arquivo “Livro.java” iniciou esses processos que de forma semelhante seria replicado em outros arquivos de coleta de dados parecidos.

Em “livro.java”, podemos ver a aplicação de partes essenciais do programa como.

1. ContadorID (static int) - que gera IDs sequenciais de forma automática, apesar de uma lógica simples de um contador.
2. titulo (String): Título do livro
3. autor (String): Autor(es) do livro
4. editora (String): Editora do livro
5. ano (int): Ano de publicação
6. isbn (String): Número ISBN (identificador único do livro)

Além dessas variáveis importantes para guardar e manipular os dados, que seria colocado no método construtor nas classes, também teríamos nessas classes, funções importantes para 1. manipular e 2. guardar esses dados como:

Getters e setters para todos os atributos:

`toString()`: Retorna uma representação simplificada do livro

`getDescricaoCompleta()`: Retorna uma descrição detalhada para exibição na interface

`equals()`: Implementação para comparar livros por ISBN

Outro fator relevante seria a utilização da biblioteca java, Em Java, a interface Serializable permite que objetos sejam convertidos em uma sequência de bytes para que possam ser armazenados em disco, quando um objeto é serializado, o Java transforma todo o seu conteúdo (atributos, valores e estrutura) em dados binários que podem ser escritos em um arquivo físico no computador, isso é feito com as classes com: ObjectOutputStream e FileOutputStream,

Além é claro da biblioteca Swing para um sistema de interface e experiência de usuário UI, Essa estrutura seria amplamente usada em outros arquivos do programa, como “professor.java” e “emprestimos.java”, com a criação inicial de formulários, e posteriormente, esses sistemas de serialização de dados.

Modelagem UML.

A modelagem UML (*Unified Modeling Language*) é uma etapa fundamental no desenvolvimento de sistemas orientados a objetos, pois permite representar graficamente a estrutura e o comportamento do sistema antes de sua implementação, neste contexto, foi realizada a modelagem de um sistema de biblioteca em Java, com foco no ator Bibliotecário, principal responsável pelas operações do sistema.

Ator: Bibliotecário

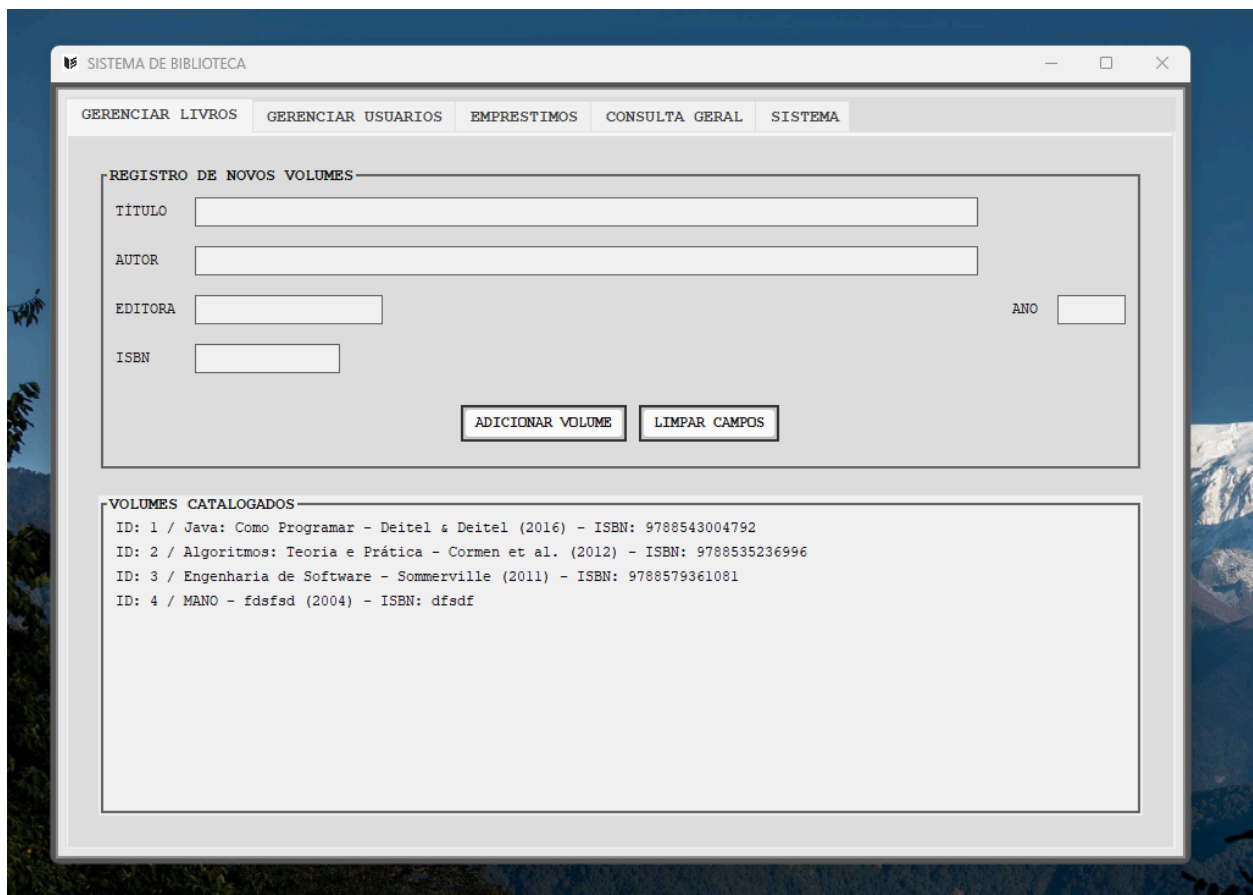
- └─> Cadastrar Livro
- └─> Remover Livro
- └─> Atualizar Informações do Livro
- └─> Registrar Empréstimo
- └─> Registrar Devolução
- └─> Cadastrar Usuário
- └─> Consultar Usuários
- └─> Consultar Acervo

Operações pensadas inicialmente:

Classes	Atributos	Métodos
Livro	- id: int- titulo: String- autor: String- ano: int- status: String (disponível, emprestado)	+ emprestar()+ devolver()
Usuario	- id: int- nome: String- email: String	(nenhum)
Emprestimo	- id: int- livro: Livro- usuario: Usuario- dataEmprestimo: Date- dataDevolucao: Date	+ registrarDevolucao()
Bibliotecario	- id: int- nome: String- senha: String	+ login()+ gerenciarSistema()

Interface do Sistema.

O sistema utiliza uma interface gráfica de usuário (GUI) desenvolvida com a biblioteca Java Swing, a interface é contida em uma janela principal (JFrame) e organizada através de um painel de abas (JTabbedPane), que segmenta as funcionalidades do sistema de forma clara e intuitiva para o usuário.



As principais seções da interface são:

Gerenciar Livros: Formulários para cadastrar, editar e remover livros do acervo.

Gerenciar Usuários: Interface para adicionar, modificar e excluir usuários (Estudantes, Professores, Funcionários).

Empréstimos: Painel para registrar novos empréstimos e efetuar a devolução de livros.

Consulta Geral: Uma área dedicada para realizar buscas em todo o acervo e nos registros do sistema.

Sistema: Funcionalidades para o gerenciamento de dados, como salvar o estado atual da biblioteca em arquivos.

Notavelmente, a aplicação possui um estilo visual personalizado,retro/industrial". Esse estilo é implementado através da classe InterfaceConfig e de configurações diretas na classe BibliotecaGUI, que definem um esquema de cores sóbrias (tons de cinza, preto e branco) e o uso de fontes monoespaçadas como "Courier New", isso confere à aplicação uma identidade visual única e consistente, afastando-se da aparência padrão das aplicações Swing.

Estrutura do Projeto e Código-Fonte.

Embora todas as classes residam no mesmo pacote SysBiblioteca, o projeto segue uma organização lógica que espelha o padrão de arquitetura Model-View-Controller (MVC). A estrutura pode ser descrita da seguinte forma:

Pacote main (lógico): Contém a classe SistemaBiblioteca, o ponto de entrada da aplicação. Sua única responsabilidade é instanciar e exibir a interface gráfica principal.

Pacote view ou ui (lógico): Composto pelas classes BibliotecaGUI e InterfaceConfig. A BibliotecaGUI é a classe central da interface, construindo todos os componentes visuais e tratando os eventos de interação do usuário (cliques em botões, seleção de itens, etc.).

Pacote model (lógico): Inclui as classes que representam as entidades de dados do sistema:

Livro:	Modela os atributos de um livro (título, autor, ISBN, etc.)
Usuario:	Classe base abstrata para os usuários da biblioteca.
Estudante, Professor, Funcionario:	Subclasses de Usuario que representam os diferentes tipos de usuários, cada um com suas próprias regras.

Empréstimo:	Modela o registro de um empréstimo, associando um Livro a um Usuario.
-------------	---

Pacote controller ou service (lógico): Representado pela classe GerenciadorBiblioteca. Esta classe centraliza a lógica de negócio da aplicação. Ela atua como um intermediário entre a interface gráfica (BibliotecaGUI) e as classes de modelo e persistência, contendo os métodos para adicionar livros, realizar empréstimos, etc.

Pacote persistence (lógico): A classe BibliotecaDB é responsável pela persistência dos dados. Ela lida com a gravação e a leitura das listas de livros, usuários e empréstimos em arquivos binários (.dat).

Trechos Exemplares do Código-Fonte: Estrutura da classe de serviço (GerenciadorBiblioteca).

```
public class GerenciadorBiblioteca {  
    private ArrayList<Livro> livros;  
    private ArrayList<Usuario> usuarios;  
    private ArrayList<Emprestimo> emprestimos;  
  
    public GerenciadorBiblioteca() {  
        if (BibliotecaDB.existemDadosSalvos()) {  
            livros = BibliotecaDB.carregarLivros();  
            // ... carrega outros dados  
        } else {  
            this.livros = new ArrayList<>();  
            // ... inicializa listas vazias  
        }  
    }  
  
    public void adicionarLivro(Livro livro){  
        livros.add(livro);  
    }  
    // ... outros métodos de negócio  
}
```

Aplicação dos Conceitos de POO.

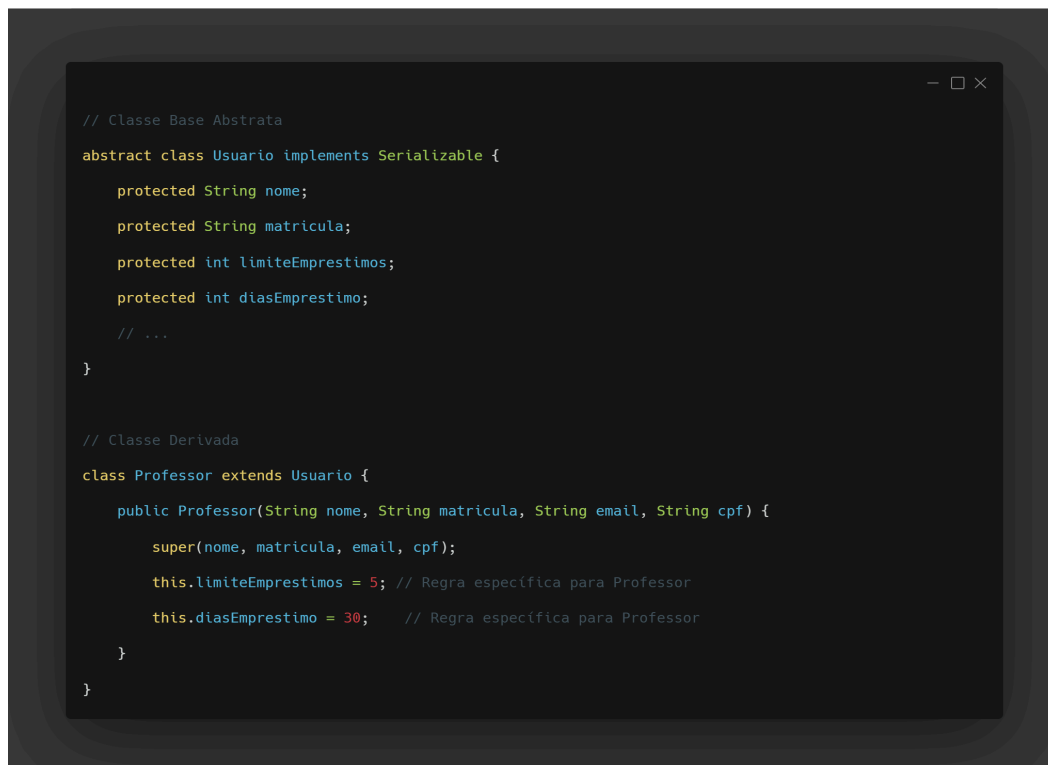
O projeto demonstra uma sólida aplicação dos principais conceitos de Programação:

Encapsulamento:

As classes de modelo (Livro, Usuário, etc.) possuem seus atributos declarados como `private` ou `protected`. O acesso a esses atributos é controlado por meio de métodos públicos `getters` e `setters`. Isso protege a integridade dos dados e oculta a complexidade interna dos objetos.

Herança:

O conceito de herança é claramente aplicado na criação dos tipos de usuário. A classe `Usuario` é `abstract`, definindo uma base comum para todos os usuários. As classes `Estudante`, `Professor` e `Funcionario` herdam de `Usuario`, reutilizando atributos e métodos comuns (como `nome`, `matricula`, `cpf`) e especializando a classe pai com regras de negócio específicas, como pode ser visto abaixo.



```
// Classe Base Abstrata
abstract class Usuario implements Serializable {
    protected String nome;
    protected String matricula;
    protected int limiteEmprestimos;
    protected int diasEmprestimo;
    // ...
}

// Classe Derivada
class Professor extends Usuario {
    public Professor(String nome, String matricula, String email, String cpf) {
        super(nome, matricula, email, cpf);
        this.limiteEmprestimos = 5; // Regra específica para Professor
        this.diasEmprestimo = 30;   // Regra específica para Professor
    }
}
```

Exemplo de Herança (Professor estendendo Usuario)

Polimorfismo:

O polimorfismo se manifesta na forma como as subclasses de `Usuario` definem seus próprios valores para os atributos `limiteEmprestimos` e `diasEmprestimo`. Embora o `GerenciadorBiblioteca` trate todos os usuários através da referência da classe base `Usuario`, o comportamento real de um empréstimo (quantos livros pode pegar e por quanto tempo) depende do tipo específico do objeto (se é `Estudante`, `Professor` ou `Funcionario`). Além disso, métodos como `toString()` e `getDescricaoCompleta()` são sobrescritos nas classes para fornecer representações textuais específicas de cada objeto.

Outros tópicos importantes do projeto:

Coleções: A API de Coleções do Java é fundamental para o projeto. A classe `GerenciadorBiblioteca` utiliza `ArrayList<Livro>`, `ArrayList<Usuario>` e `ArrayList<Emprestimo>` para armazenar e gerenciar em memória todos os registros do sistema.

Organização Modular: Como descrito na estrutura do projeto, o sistema exibe uma clara separação de responsabilidades (interface, lógica, modelo e persistência), promovendo um design modular e de fácil manutenção.

Persistência de Dados (Serialização): É importante notar que o sistema não utiliza JDBC para interação com um banco de dados relacional. Em vez disso, a persistência é implementada através da serialização de objetos Java. A classe `BibliotecaDB` utiliza `ObjectOutputStream` para gravar o estado das coleções (`ArrayLists`) em arquivos e `ObjectInputStream` para restaurá-los, permitindo que os dados da aplicação persistam entre execuções.