# Universidade Católica do Salvador – UCSAL

## Bacharelado em Engenharia de Software

## Disciplina de Programação Orientada a Objetos – BES008

## Atividade Pontuada – 02

**RELATÓRIO**

**SISTEMA DE BIBLIOTECA EM JAVA**

**USANDO CONCEITOS DE ORIENTAÇÃO A OBJETOS**

Aluno: Sérgio Lopes Júnior

Profa Cristiana Pereira Bispo

Novembro

2024

# Descrição do Sistema

* 1. Características Gerais:

Este é um trabalho acadêmico com o objetivo de utilizar os conceitos de programação orientada a objetos para a simulação de um sistema de gerenciamento de bibliotecas.

O aplicativo foi desenvolvido para gerenciamento simplificado de uma biblioteca sem a utilização de sistemas de gerenciamento de banco de dados e tampouco implementação de qualquer outro tipo de método para gravação de dados em disco. Tais funcionalidades podem ser implementadas em etapas posteriores, além de outras que se fizerem necessárias.

Para fins de simulação das funcionalidades foi implementada uma Classe que simula uma carga de dados chamada LoadDB, que será descrita posteriormente.

Em função da simplicidade do sistema foram criados apenas dois pacotes (packages) para organização dos códigos: *application*, que contém as classes de LoadDB, Main, Menu e TestMain, responsáveis pelas cargas iniciais, menus e inicializador do sistema; e o pacote *classes* que contém as classes necessárias a execução da aplicação com suas respectivas regras de negócio, que são Book, AcademicWork, CollectionItem (abstract), Lending, Library, LendingRules (interface)e User.

**classes.LendRules:** Interface criada para a implementação de regras para os empréstimos, sendo uma delas já implementada, que determina o tempo máximo de empréstimo e outra que determina multas por atraso (não implementada).

*int lendPeriod()*

*double lteReturnFee()*

**classes.CollectionItem:** Classe abstrata criada como superclasse para os itens a serem incorporados à biblioteca que implementa LendRules. Ela contém atributos que serão comuns a todos os itens da coleção como título, autor, número de páginas, ano de publicação e disponibilidade ou não no acervo. Nessa classe foi adicionado um atributo estático para servir como gerador de código único para os itens do acervo.

*public String avaiableStatus()*

*public int lendPeriod()*

*public abstract CollectionItem createCollectionItem(Scannner sc)*

**classes.Book:** Classe concreta que estende CollectionItem e adiciona os atributos característicos de um exemplar de um livro como edição, ano da edição e editora, os demais são herdados da superclasse.

*@Override public int lendPeriod()*

*@Override public CollectionItem createCollectionItem(Scannner sc)*

*@Override public double lateReturnFee() “ A ser implementada em versão futura”*

**classes.AcademicWork:** Classe concreta que estende CollectionItem e adiciona os atributos característicos de trabalhos acadêmicos como instituição de vínculo e tipo de trabalho, seja ele um artigo, trabalho de conclusão de curso, dissertação ou tese.

*@Override public int lendPeriod()*

*@Override public CollectionItem createCollectionItem(Scannner sc)*

*private Worktype chooseWorktype(Scanner sc)*

*@Override double lateReturnFee()*

**classes.User:** Classe concreta que cria usuários para a biblioteca. Em função da simplicidade não utiliza conceitos de herança ou interfaces. Possui os atributos de idUser criado com base em um contador de objetos estáticos criado para a classe e outros atributos como nome, username, data de nascimento e CPF.

*public static User createUser(Scanner SC)*

**classe Lending:** Classe concreta que cria objetos do tipo User e CollectionItem para que sejam armazenados na listagem de empréstimos. Além desses atributos a classe também contém dois atributos do tipo LocalDate para armazenar data de empréstimo e data de retorno calculada. Conceitualmente essa classe seria uma candidata a armazenar o método que calcula a multa por atraso, sendo necessário nesse caso que ela implemente a interface LendRules juntamente com os itens do acervo, deste modo, quando implementada, a multa pode ser diferente dependendo do tipo de item tomado emprestado.

**classe.Library:** Classe concreta que controla as listas de objetos necessários para o funcionamento da aplicação. São criadas três listas para armazenar os objetos do tipo CollectionItem, User e Lending. Não foram implementados métodos getter e setter para a classe.

public void lendItem(CollectionItem collectionItem, User user)

public void loadLendItem(CollectionItem collectionItem, User user)

public void lendItem(int idItem, int idUser)

private CollectionItem getItemFromList(int idItem)

private User getUserFromList(int idUser)

public void returnItem(CollectionItem collectionItem, User user)

public void returnItem(int idItem,int idUser)

public void listCollection()

public void listLendedItems()

public void listUsers()

* 1. Funcionalidades:

As funcionalidades implementadas não compõem a totalidade das necessidades para um sistema deste tipo em função das características e objetivo da atividade. As funcionalidades implementadas se dividem em três grupos: Inserção de dados, Listagem de dados armazenados e Controle de empréstimos.

