```java
package library;

class No {
    Book livro;
    No esquerda, direita;

    public No(Book livro) {
        this.livro = livro;
        esquerda = direita = null;
    }
}

public class Arvore {
   private No raiz;

   public Arvore() {
      this.raiz = null;
   }

   public void inserir(Book livro) {
      this.raiz = this.inserirRecursivo(this.raiz, livro);
   }

   private No inserirRecursivo(No raiz, Book livro) {
      if (raiz == null) {
         return new No(livro);
      } else {
         if (livro.getTitle().compareToIgnoreCase(raiz.livro.getTitle()) < 0) {
            raiz.esquerda = this.inserirRecursivo(raiz.esquerda, livro);
         } else {
            raiz.direita = this.inserirRecursivo(raiz.direita, livro);
         }

         return raiz;
      }
   }

   public void buscarPorTitulo(String titulo) {
      System.out.println("Buscando livros com título: " + titulo);
      this.buscarTituloRecursivo(this.raiz, titulo);
   }
```

```java
    private void buscarTituloRecursivo(No raiz, String titulo) {
        if (raiz != null) {
            this.buscarTituloRecursivo(raiz.esquerda, titulo);
            if (raiz.livro.getTitle().toLowerCase().contains(titulo.toLowerCase())) {
                System.out.println(raiz.livro);
            }
            this.buscarTituloRecursivo(raiz.direita, titulo);
        }
    }

    public void buscarPorAno(int ano) {
        System.out.println("Buscando livros do ano: " + ano);
        this.buscarAnoRecursivo(this.raiz, ano);
    }

    private void buscarAnoRecursivo(No raiz, int ano) {
        if (raiz != null) {
            this.buscarAnoRecursivo(raiz.esquerda, ano);
            if (raiz.livro.getReleaseYear() == ano) {
                System.out.println(raiz.livro);
            }
            this.buscarAnoRecursivo(raiz.direita, ano);
        }
    }

    public void buscarPorAutor(String autor) {
        System.out.println("Buscando livros do autor: " + autor);
        this.buscarAutorRecursivo(this.raiz, autor);
    }

    private void buscarAutorRecursivo(No raiz, String autor) {
        if (raiz != null) {
            this.buscarAutorRecursivo(raiz.esquerda, autor);
            if (raiz.livro.getAuthor().equalsIgnoreCase(autor)) {
                System.out.println(raiz.livro);
            }
            this.buscarAutorRecursivo(raiz.direita, autor);
        }
    }
}
```

Descrição Geral

A classe Arvore implementa uma árvore binária de busca (BST - Binary Search Tree) usada para armazenar objetos da classe Book. A árvore organiza os livros com base em atributos como título, autor e ano de lançamento, oferecendo métodos para inserção e busca eficiente de livros. Cada nó na árvore contém um livro e possui referências para os filhos à esquerda e à direita.

Atributos da Classe

raiz (No): Representa o nó raiz da árvore. Inicialmente, é nulo, indicando uma árvore vazia.

Construtor

Arvore(): Inicializa a árvore com a raiz definida como nula, criando uma árvore vazia pronta para receber inserções.

Métodos Públicos

inserir(Book livro)

Este método insere um novo livro na árvore. O livro é inserido com base no título, utilizando comparação lexicográfica. A inserção ocorre de forma recursiva, garantindo que os livros com títulos menores que o nó corrente sejam inseridos à esquerda, e os maiores, à direita.

Parâmetros:

livro: O objeto da classe Book a ser inserido na árvore.

Retorno: Void (sem retorno).

buscarPorTitulo(String titulo)

Busca e exibe todos os livros na árvore que contenham, parcial ou completamente, o título fornecido. O método percorre a árvore de forma recursiva.

Parâmetros:

titulo: Uma string representando o título ou parte do título a ser buscado.

Retorno: Void (exibe os resultados da busca no console).

buscarPorAno(int ano)

Busca e exibe todos os livros que foram lançados no ano especificado. A busca também é feita de forma recursiva.

Parâmetros:

ano: Um inteiro que representa o ano de lançamento dos livros a serem buscados.

Retorno: Void (exibe os resultados da busca no console).

buscarPorAutor(String autor)

Busca e exibe todos os livros escritos pelo autor especificado. Assim como os outros métodos de busca, este também é recursivo.

Parâmetros:

autor: Uma string representando o nome completo do autor a ser buscado.

Retorno: Void (exibe os resultados da busca no console).

Métodos Privados

inserirRecursivo(No raiz, Book livro)

Método auxiliar que realiza a inserção recursiva de um livro na árvore. O livro é comparado com o nó corrente e inserido no local apropriado (à esquerda ou à direita).

Parâmetros:

raiz: O nó corrente da árvore.

livro: O livro a ser inserido.

Retorno: O nó atualizado após a inserção.

buscarTituloRecursivo(No raiz, String titulo)

Método recursivo utilizado para percorrer a árvore e buscar livros que contenham o título fornecido.

Parâmetros:

raiz: O nó corrente da árvore.

titulo: O título ou parte do título a ser buscado.

Retorno: Void (exibe os resultados no console).

buscarAnoRecursivo(No raiz, int ano)

Método recursivo utilizado para percorrer a árvore e buscar livros lançados no ano especificado.

Parâmetros:

raiz: O nó corrente da árvore.

ano: O ano de lançamento dos livros a serem buscados.

Retorno: Void (exibe os resultados no console).

buscarAutorRecursivo(No raiz, String autor)

Método recursivo utilizado para percorrer a árvore e buscar livros de um autor específico.

Parâmetros:

raiz: O nó corrente da árvore.

autor: O nome do autor a ser buscado.

Retorno: Void (exibe os resultados no console).

Funcionamento Geral

A Arvore é uma estrutura que permite armazenar, buscar e organizar livros de maneira eficiente. Os livros são inseridos com base no título, mas a busca pode ser feita por título, ano ou autor. O comportamento da árvore é recursivo, ou seja, todos os métodos de inserção e busca percorrem a árvore explorando os filhos à esquerda e à direita conforme necessário, garantindo que a árvore mantenha suas propriedades estruturais.

Complexidade

Inserção: O tempo de inserção em uma árvore binária de busca é proporcional à altura da árvore. Na melhor das hipóteses (árvore balanceada), a complexidade é O(log n), onde n é o número de nós. Na pior das hipóteses (árvore degenerada), a complexidade pode ser O(n).

Busca: Assim como a inserção, a busca tem complexidade O(log n) na melhor das hipóteses e O(n) na pior.