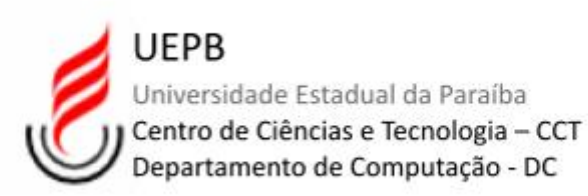


Disciplina: Laboratório de Estrutura de Dados

Professor: Fábio Luiz Leite Júnior

Caio Sérgio Ramalho Lima

Projeto UT2

Justificativa e Aplicação das Estruturas de Dados no Projeto

Campina Grande - PB
2025

Justificativa das Estruturas de Dados Utilizadas

Lista Encadeada (ListaEncadeadaImpl)

Justificativa

Utilizar uma lista encadeada simples para armazenar sequências de elementos com possibilidade de inserção e remoção dinâmica, superando a limitação e rigidez de arrays estáticos.

Vantagens

- Crescimento dinâmico, sem necessidade de definir tamanho fixo;
- Inserção de novos elementos no final da lista sem custo de realocação;
- Fácil iteração e impressão dos elementos na ordem de inserção;
- Estrutura leve, composta por nós simples que apontam para o próximo elemento.

Exemplo no código

```
@Override  154 usages
public void insert(T chave) {
    NodoListaEncadeada<T> novo = new NodoListaEncadeada<T>(chave);
    NodoListaEncadeada<T> atual = cabeca;
    while (atual.getProximo() != cauda) {
        atual = atual.getProximo();
    }
    atual.setProximo(novo);
    novo.setProximo(cauda);
    tamanho++;
}
```

Facilita o armazenamento e manipulação de listas com tamanho variável, eliminando o problema de arrays fixos que podem desperdiçar espaço ou necessitar realocação custosa.

Fila Encadeada (MinhaFilaEncadeada)

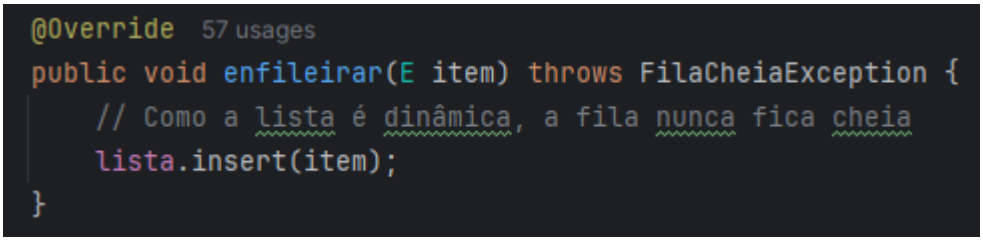
Justificativa

A fila dinâmica foi implementada para garantir uma estrutura que respeite a ordem FIFO (primeiro a entrar, primeiro a sair), essencial para gerenciar processos, filas de tarefas ou dados, sem limitações de tamanho pré-definido.

Vantagens

- Operações enfileirar e desenfileirar em tempo constante (O(1));
- Sem necessidade de deslocar elementos (problema comum em arrays fixos usados para filas);
- Uso eficiente da memória, crescendo conforme a demanda;
- Métodos claros para consultar a cabeça (primeiro elemento) e a cauda (último elemento).

Exemplo no código



Garante a organização dos elementos em ordem sem limitação por tamanho fixo, sem custo de cópia ou deslocamento, evitando gargalos em aplicações que dependem de filas dinâmicas.

Conjunto Dinâmico Encadeado

(MeuConjuntoDinamicoEncadeado)

Justificativa

Implementar uma coleção de elementos inteiros únicos que permita inserção, remoção e busca eficientes, além de operações adicionais como obter sucessor, predecessor, mínimo e máximo, gerenciando a estrutura de forma dinâmica.

Vantagens

- Garante unicidade dos elementos, evitando duplicatas;
- Gerenciamento dinâmico da memória, sem necessidade de arrays fixos;
- Permite operações avançadas (exemplo: predecessor, sucessor), facilitando consultas estruturais;
- Flexibilidade para crescer ou reduzir conforme o número de elementos.

Exemplo no código

```java
@Override  48 usages
public void inserir(Integer item) {
    if (item == null)
        throw new IllegalArgumentException("Elemento não pode ser null");
    Node novo = new Node(item);
    if (cabeca == null) {
        cabeca = cauda = novo;
    } else {
        cauda.prox = novo;
        cauda = novo;
    }
    tamanho++;
}
```

Facilita o controle de um conjunto de inteiros com operações dinâmicas e complexas, evitando limitações dos arrays e atendendo às necessidades de operações de conjunto matemático.

Conclusão

Durante a implementação das estruturas de dados, percebi que usar listas encadeadas foi uma escolha bastante eficiente para lidar com listas, filas e conjuntos. Elas se mostraram muito úteis principalmente por não precisarem de um tamanho fixo e por permitirem inserções e remoções de forma simples, sem ficar movendo os elementos como acontece em arrays.

Além disso, elas se adaptam bem a situações em que os dados mudam com frequência. Então, cada estrutura que usei se comportou como esperado e atendeu bem ao que cada TAD precisava.