Trabalho Final Java Collections - RELATÓRIO TÉCNICO

Grupo:

Delphino Luciani de Paula Araujo Filho - 20219019597

Samia Ribeiro Gonçalves - 20219022082

**1.1 - Introdução:**

O presente relatório visa detalhar o Trabalho final do grupo, que compõe a terceira nota da disciplina de Estrutura de Dados do Departamento de Computação - CCN da Universidade Federal do Piauí, período 2022.2, ministrada pelo professor Dr. Raimundo Moura.

**1.2 - Preparação do grupo**

Todos os integrantes do grupo se reuniram em uma chamada online onde todos contribuíram para a criação do trabalho, dividindo em parte de código e simulação/ elaboração do relatório, criação de gráficos e estudo dos resultados

**1.3 - Material utilizado**

Para criação dos gráficos e tabelas foi utilizado o programa Excel e o Eclipse foi utilizado como principal plataforma de desenvolvimento e simulação em um Desktop com processador Ryzen 5 3600(6 núcleos 12 threads 3.6 Ghz)

**2.0 – Questões**

**2.1 – Questão 01**

Vector

Características: O Vector é uma classe sincronizada que armazena elementos em uma matriz dinâmica de tamanho ajustável

Vantagens: permite a inserção e remoção de elementos em qualquer posição do vetor, o que pode ser útil em certas situações. No entanto, essa flexibilidade tem um preço: a inserção e remoção de elementos no meio do vetor pode ser uma operação custosa em termos de desempenho, já que exige que todos os elementos subsequentes sejam deslocados para acomodar a mudança. Além disso, o Vector é considerado obsoleto em versões mais recentes do Java, sendo substituído pelo ArrayList.

Desvantagens: O Vector é mais lento que o ArrayList e o LinkedList, pois é sincronizado e é necessário adquirir um bloqueio antes de acessar ou modificar a lista. Além disso, o Vector não é tão flexível em termos de tamanho da lista e realocação de memória como o ArrayList.

LinkedList

Características: O LinkedList é uma lista duplamente vinculada que armazena elementos como nós. Cada nó tem um ponteiro para o próximo e o anterior nó, permitindo que a lista seja percorrida em ambas as direções.

Vantagens: O LinkedList é rápido para adicionar e remover elementos, especialmente no início e no fim da lista, pois os nós são interligados e podem ser facilmente rearranjados. Além disso, o LinkedList é uma boa escolha quando o tamanho da lista é desconhecido ou pode mudar frequentemente.

Desvantagens: O LinkedList não é tão rápido quanto o ArrayList quando se trata de pesquisar elementos, pois a lista deve ser percorrida sequencialmente para encontrar um elemento específico. Além disso, o LinkedList requer mais espaço em memória do que o ArrayList, pois cada nó precisa armazenar dois ponteiros para os nós vizinhos.

ArrayList

Características: O ArrayList é uma matriz dinâmica que armazena elementos em uma matriz contígua de memória. A matriz pode ser aumentada automaticamente à medida que novos elementos são adicionados à lista.

Vantagens: O ArrayList é rápido para pesquisar elementos, pois os elementos podem ser acessados diretamente por índice. Além disso, o ArrayList é mais eficiente em termos de uso de memória do que o LinkedList, pois não há necessidade de armazenar ponteiros adicionais para os nós vizinhos.

Desvantagens: O ArrayList é menos eficiente ao adicionar ou remover elementos no meio da lista, pois pode ser necessário realocar a matriz e copiar elementos. Além disso, o ArrayList não é sincronizado e, portanto, não é seguro para threads.

https://youtu.be/hFa8NJv7iQQ\*

**2.2 – Questão 02**

HashSet: é uma classe que implementa um conjunto baseado em tabela de dispersão (hash table). Ele não garante a ordem dos elementos e permite a inserção de valores nulos. O HashSet é eficiente para adicionar, remover e verificar se um elemento está presente na coleção. Uma das desvantagens do HashSet é que ele não garante a ordem dos elementos, o que pode ser um problema em algumas aplicações que dependem da ordem dos elementos.

LinkedHashSet: é uma classe que implementa um conjunto baseado em tabela de dispersão com uma lista encadeada que mantém a ordem de inserção dos elementos. Ele permite a inserção de valores nulos. O LinkedHashSet é eficiente para adicionar, remover e verificar se um elemento está presente na coleção, assim como o HashSet. Além disso, ele mantém a ordem de inserção dos elementos, o que pode ser útil em certas aplicações.

TreeSet: é uma classe que implementa um conjunto ordenado baseado em árvore. Ele não permite a inserção de valores nulos e garante a ordem dos elementos. O TreeSet é eficiente para adicionar, remover e verificar se um elemento está presente na coleção, desde que a ordem dos elementos seja levada em consideração. Além disso, ele garante a ordem dos elementos, o que pode ser útil em certas aplicações que dependem da ordem dos elementos. No entanto, a ordenação dos elementos pode tornar as operações mais custosas em termos de desempenho.

https://youtu.be/DcTEsdA6d7c

**2.3 – Questão 03**HashMap: Permite armazenar pares chave-valor em uma tabela hash; Não garante a ordem de inserção dos elementos; Permite acesso aleatório aos elementos;

Vantagem: é mais rápido para inserir e acessar elementos do que as outras duas classes;

Desvantagem: pode haver colisões de hash, o que pode diminuir a performance; Também pode ser mais lento para iterar sobre todos os elementos, pois não garante ordem.

LinkedHashMap: Mantém a ordem de inserção dos elementos; Permite acesso aleatório aos elementos; Implementado como uma lista ligada com uma tabela hash;

Vantagem: permite percorrer os elementos em ordem de inserção, além de possuir acesso rápido aos elementos;

Desvantagem: pode ser mais lento do que o HashMap, especialmente para inserções.

TreeMap: Armazena os elementos em uma estrutura de árvore balanceada; Os elementos são ordenados por chave;

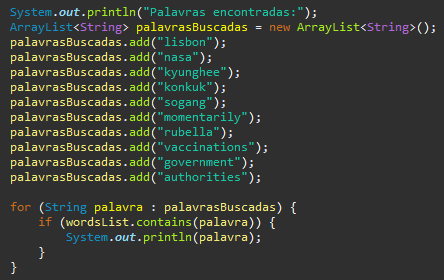
Vantagem: permite a ordenação dos elementos por chave, o que pode ser útil em algumas situações;

Desvantagem: pode ser mais lento do que o HashMap e o LinkedHashMap para inserir e acessar elementos.

Portanto, se a ordem de inserção ou a ordenação dos elementos for importante, o LinkedHashMap ou o TreeMap podem ser mais adequados. Se a performance for uma preocupação maior e não for necessário manter a ordem, o HashMap pode ser uma escolha melhor.

https://youtu.be/Z1FPQzc3rVs

**2.3 – Questão 04**

**2.3 – Questão 05**

Alteração Realizada

https://youtu.be/CMaO3OAiD2M:

**2.3 – Questão 06**

**2.3 – Questão 07**

****

https://youtu.be/I1KbVQ2FLU8

**2.3 – Questão 08**