Coleções

ArrayList

A classe ArrayList é uma matriz redimensionável, que pode ser encontrada no pacote java.util.

A diferença entre um array integrado e um ArrayList em Java é que o tamanho de um array não pode ser modificado (se você deseja adicionar ou remover elementos de / para um array, você deve criar um novo). Enquanto os elementos podem ser adicionados e removidos de um ArrayList sempre que você quiser. A sintaxe também é um pouco diferente:

LinkedList

Anteriormente descrevi sobre ArrayList. O LinkedList é uma classe idêntica ao ArrayList.

ArrayList vs. LinkedList

O LinkedList pode conter na sua coleção muitos objetos do mesmo tipo, como o ArrayList. Elas se parecem pois tem métodos e implementam a mesma interface List, em que você pode adicionar, remover e limpar da mesma forma. Contudo mesmo que elas possa parece idênticas foram construídas de forma diferente.

Como funciona o ArrayList

A classe ArrayList possui um array regular dentro dela. Quando um elemento é adicionado, ele é colocado na matriz. Se o array não for grande o suficiente, então um novo array maior é criado para substituir o antigo e o antigo é removido.

Como funciona o LinkedList

O LinkedList armazena seus itens em "contêineres". A lista possui um link para o primeiro container e cada container possui um link para o próximo container na lista. Para adicionar um elemento à lista, o elemento é colocado em um novo contêiner e esse contêiner é vinculado a um dos outros contêineres da lista.

Quando usar ArrayList:

- Você deseja acessar itens aleatórios com frequência.
- Você só precisa adicionar ou remover elementos no final da lista.

Quando usar LinkedList:

- Você só usa a lista percorrendo-a em vez de acessar itens aleatórios.
- Você frequentemente precisa adicionar e remover itens do início ou meio do
  Lista.

Conclusão em relação ao ArrayList e LinkedList

Para muitos casos, o ArrayList é mais eficiente, pois é comum, pois se precisar de acesso a itens aleatórios na lista, mas o LinkedList fornece vários métodos para fazer certas operações com mais eficiência.

HashSet

Um HashSet é uma coleção de itens em que cada item é único e pode ser encontrado no pacote java.util. Apoiada por uma tabela hash que é, na verdade, uma instância HashMap.

Os itens em um HashSet são na verdade objetos. Lembrando que uma String em Java é um objeto, e não um tipo primitivo. Para usar outros tipos, como int, você deve especificar uma classe de wrapper equivalente: Integer. Para outros tipos primitivos, use: Boolean para booleano, Character para char, Double para double, entre outros.

TreeSet

O TreeSet é uma das implementações mais importantes, pois utiliza uma árvore de armazenamento. A ordem dos elementos é mantida por um conjunto usando sua ordem natural, seja ou não fornecido um comparador. Isso deve ser consistente igual, como se fosse para implementar corretamente a interface Set. Ele também pode ser solicitado por um Comparador fornecido no momento da criação do conjunto, dependendo de qual construtor é usado. O TreeSet implementa uma interface NavigableSet herdando a classe AbstractSet.

HashSet vs. TreeSet

Utilizando HashSet para operações como pesquisa, inserção e exclusão. Leva um tempo constante para essas operações, em média, sendo HashSet mais rápido que TreeSet. HashSet é implementado usando uma tabela de hash. Para outros tipos de operações o TreeSet será mais rápido, pois utiliza busca em árvore binária.

Utilização de comparadores para HashSet e TreeSet

HashSet usa o método equals (), para comparar dois objetos em Set e para detectar duplicatas. TreeSet usa o método compareTo () para o mesmo propósito. Se equals () e compareTo() não forem consistentes, ou seja, para dois objetos iguais, equals deve retornar verdadeiro enquanto compareTo () deve retornar zero, então quebrará o contrato da interface Set e permitirá duplicatas em implementações Set como TreeSet.

HashMap

Um HashMap, entretanto, armazena itens em pares "chave / valor" e você pode acessá-los por um índice de outro tipo. Objeto é usado como uma chave índice para outro objeto valor. Ele pode armazenar diferentes tipos: chaves de string e valores inteiros, ou o mesmo tipo, como: chaves de string e valores de

string. Para usar essa classe e seus métodos, você precisa importar o pacote java.util.HashMap ou sua superclasse.

Use o método keySet () se quiser apenas as chaves e use o método values () se quiser apenas os valores.