07-Coleções-Parte-2

Conteúdo: Coleções em Java. Set. Map.

Hash - Tabela Dispersão

HashCode de Object

- int hashCode(): retorna um valor inteiro que é o hash do objeto, para uso com coleções com esse recurso.
- Não é um número único para cada objeto. Vários objetos diferentes (!equals) podem ter o mesmo código de hash
- Objetos iguais devem retornar o mesmo hash.

Coleção Set (1)

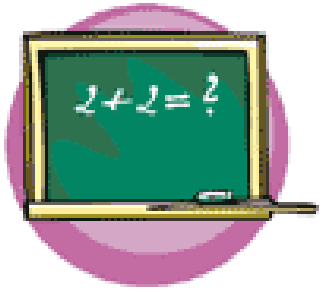
- não pode ter objetos-elemento duplicados
  - O método equals de um objeto já armazenado retorna true tendo como parâmetro o objeto sendo inserido
  - Implementações:
    - HashSet: iteração entre os elementos de forma não ordenada
    - LinkedHashSet: iteração entre os elementos de forma ordenada, pela ordem de inserção
    - TreeSet: armazenamento, busca, e iteração ocorrem em um ordem especificada (ordem natural ou personalizada)

Coleção Set (2)

- Exemplo usando o HashSet

```
public class Main {
  public static void main(String[] args) {
    Set<String> unicos=new HashSet<String>();
    Set<String> dups=new HashSet<String>();
    for(String valor:args)
      if(!unicos.add(valor)) dups.add(valor);
    unicos.removeAll(dups);
    System.out.println(unicos);
    System.out.println(dups);
  }
}
```

Define o tipo dos elementos



O que produz?

Coleção Map (1)

- Interface Map
  - Cada chave (única) mapeia a um objeto
  - V put(T key, V value), V get(T key), V remove(T key), boolean containsKey(T key), boolean containsValue(V value), int size(), boolean isEmpty(), putAll(Map), clear(), Set keySet(), Collection values(), Set entrySet()
  - Se invocar put para uma chave que exista, o objeto é substituído
  - Implementações: HashMap, TreeMap

Valor anterior

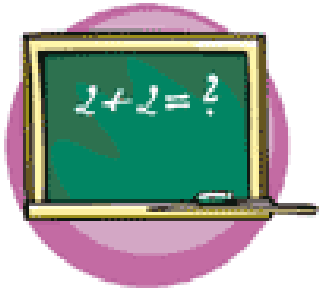
Quantidade de entradas chave-valor

Retorna uma visão ordenada de elementos Map.Entry (chave-valor) num Map.

Coleção Map (2)

```
import java.util.*;
public class TestaTreeMap {
  public static void main(String[] args) {
    Map<String,Integer> m=new TreeMap<String,Integer>();
    for(int i=0;i<args.length;i++) {
      Integer freq=m.get(args[i]);
      m.put(args[i],freq==null ? 1 : ++freq );
    }
    System.out.println(m.size()+" tokens distintos.");
    Set<String> s=m.keySet();
    for(String str:s) {
      System.out.println(str+":"+m.get(str));
    }
  }
}
// invocado com:
// $ java  TestaTreeMap ana pedro ana augusto pedro ana leo
```

O que produz?



Ordenação em Coleções

- Ordenação
  - Collections.sort(List)
  - Objetos devem implementar interface Comparable
    - Wrappers e String, Date, etc implementam
    - Classes próprias devem implementar Comparable, caso contrário é lançada exceção ClassCastException

```
public interface Comparable<T> {
  public int compareTo(T);
}
```

Retorna -1, se o objeto comparado (parâmetro) for menor que o atual, 0 se igual, e 1, se maior

    - Caso a classe não implemente Comparable, ou seja, deseja-se outra ordenação, pode-se usar o Comparator

Ver código Coleções TestaSortCollections.java

Set com Ordenação

- Interface SortedSet (extensão de Set)
  - elementos ordenados por ordem padrão ou definido em um Comparator.
  - Deve-se usar a implementação TreeSet.
  - SortedSet subSet(T deElemento,T ateElemento)
  - SortedSet headSet(T e), SortedSet tailSet(T e)
  - T first(), T last(), Comparator comparator()
  - O Iterator (de Set) varre o em ordem
  - O toArray() retorna elementos ordenados

Map com Ordenação

- Interface SortedMap (extensão de Map)
  - elementos ordenados pela chave na ordem padrão ou definido em um Comparator
  - Deve-se usar a implementação TreeMap
  - SortMap subSet(T deElemento,T ateElemento)
  - SortMap headSet(T e), SortMap tailSet(T e)
  - T first(), T last(), Comparator comparator()

Comparando Coleções (1)

- Detalhes das Implementações das coleções
  - Vetor redimensionável (implementa List)
    - ArrayList
    - LinkedList (usa lista duplamente encadeada)
  - Usam tabela de Hash
    - HashSet (Interface Set) e HashMap (interface Map)
    - Exigem sobreposição do equals e hashCode
  - Árvore Rubro-Negra:
    - TreeSet e TreeMap (interface Set e Map, respectivamente)
    - Exigem que os elementos sejam comparáveis

Comparando Coleções (2)

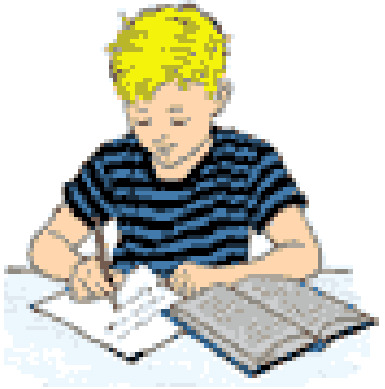
- ArrayList x LinkedList
  - ArrayList: acesso posicional mais rápido
  - LinkedList: melhor quando tem inserções/remoções no meio da lista
- HashSet x TreeSet
  - HashSet: melhor performance em todos os casos
  - TreeSet: ordenação. Para uso com SortedSet.
- HashMap x TreeMap
  - HashMap: geralmente a melhor
  - TreeMap: para uso com SortedMap

Classe Utilirária

- Classe Collections
  - .sort(List)
  - .sort(List,Comparator)
  - .shuffle(List)
  - .reverse(List)
  - .fill(List,T)
  - .copy(List dest,List ori)
  - .binarySearch(List,T)
  - .min(Collection)
  - .max(Collection)

Exercícios Fixação

- 07-Coleções-Parte-2
- T5 (sobre coleções)



Humor


NÃO IMPORTA QUANTOS RECURSOS VOCÊ TEM
SE VOCÊ NÃO SABE COMO USÁ-LOS,
NUNCA SERÁ SUFICIENTE.