

## PG II

# Programação Orientada aos Objectos em Java

### Java.util:

- Introdução
- Collections Framework – Interfaces
- Collections Framework – Classes
- Exemplos de aplicações com contentores do package java.util

# O que é uma *Collection*?

- Uma *collection*, também vulgarmente designada por *container*, é um objecto que agrupa múltiplos elementos num única unidade.
- As collections são usadas para:
  - armazenamento, retorno e manipulação de dados;
  - Transmissão de dados de um método para outro.
- Uma *collection* representa tipicamente um conjunto elementos que formam um agrupamento natural, tais como um directório de emails, uma agenda de contactos, etc.

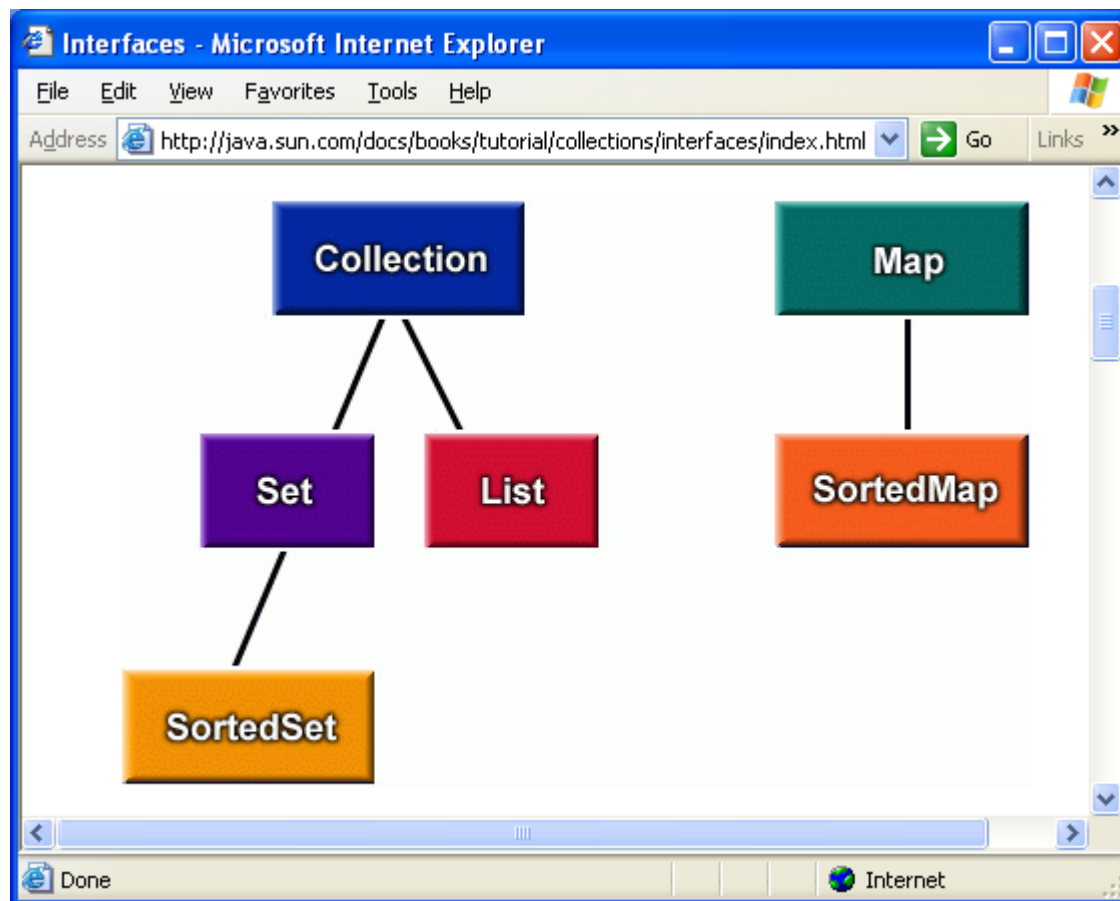
## O que é uma *Collections Framework*?

É uma arquitectura que facilita a representação e manipulação de colecções, constituída por:

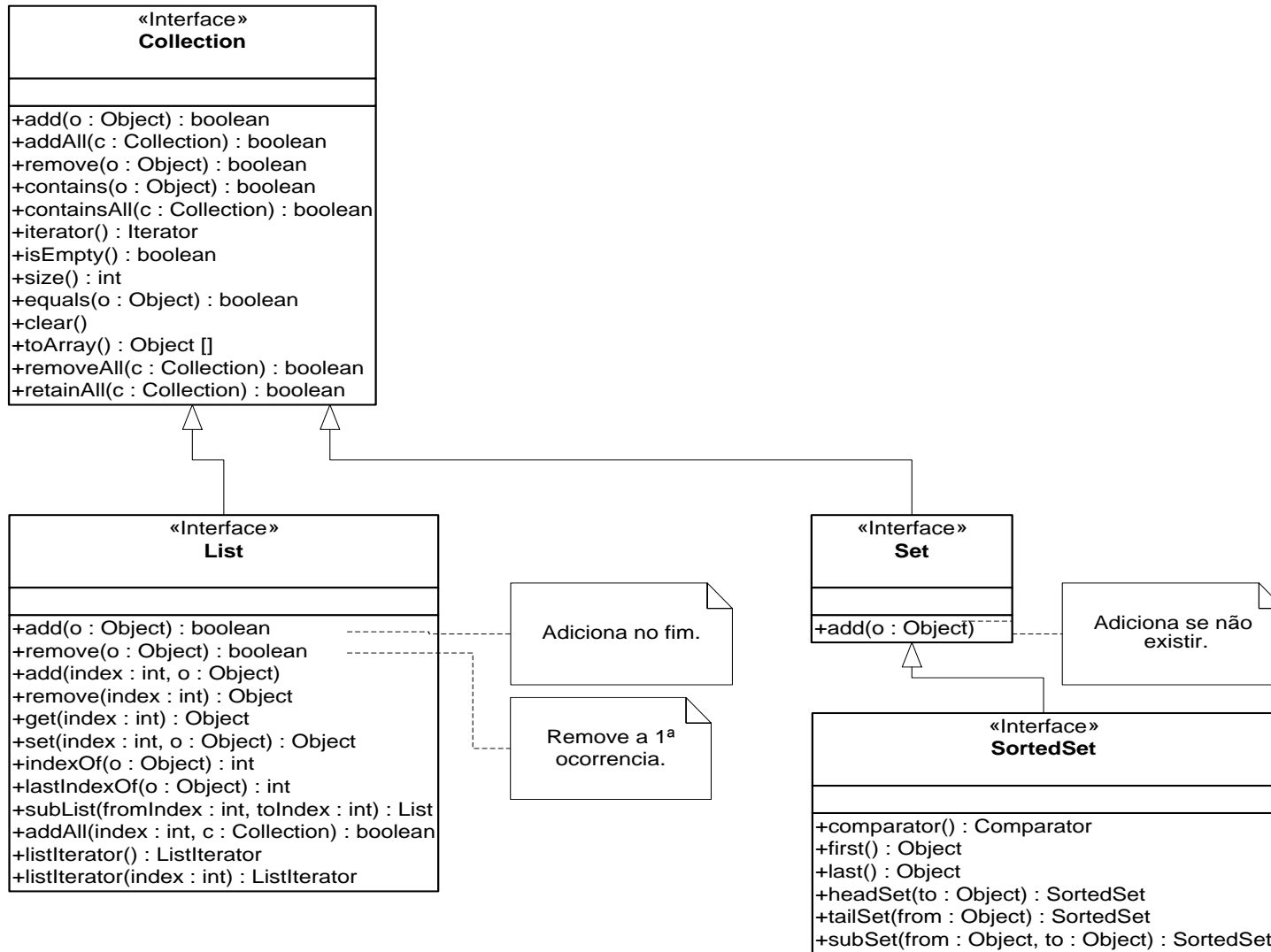
- **Interfaces** – Possibilita a manipulação de colecções independentemente dos detalhes de implementação;
- **Classes** – Implementações concretas dos interfaces das colecções.
- **Algoritmos** – métodos que desenvolvem computações úteis, como pesquisa e ordenação, em implementações de colecções.

# Collections Framework - Interfaces

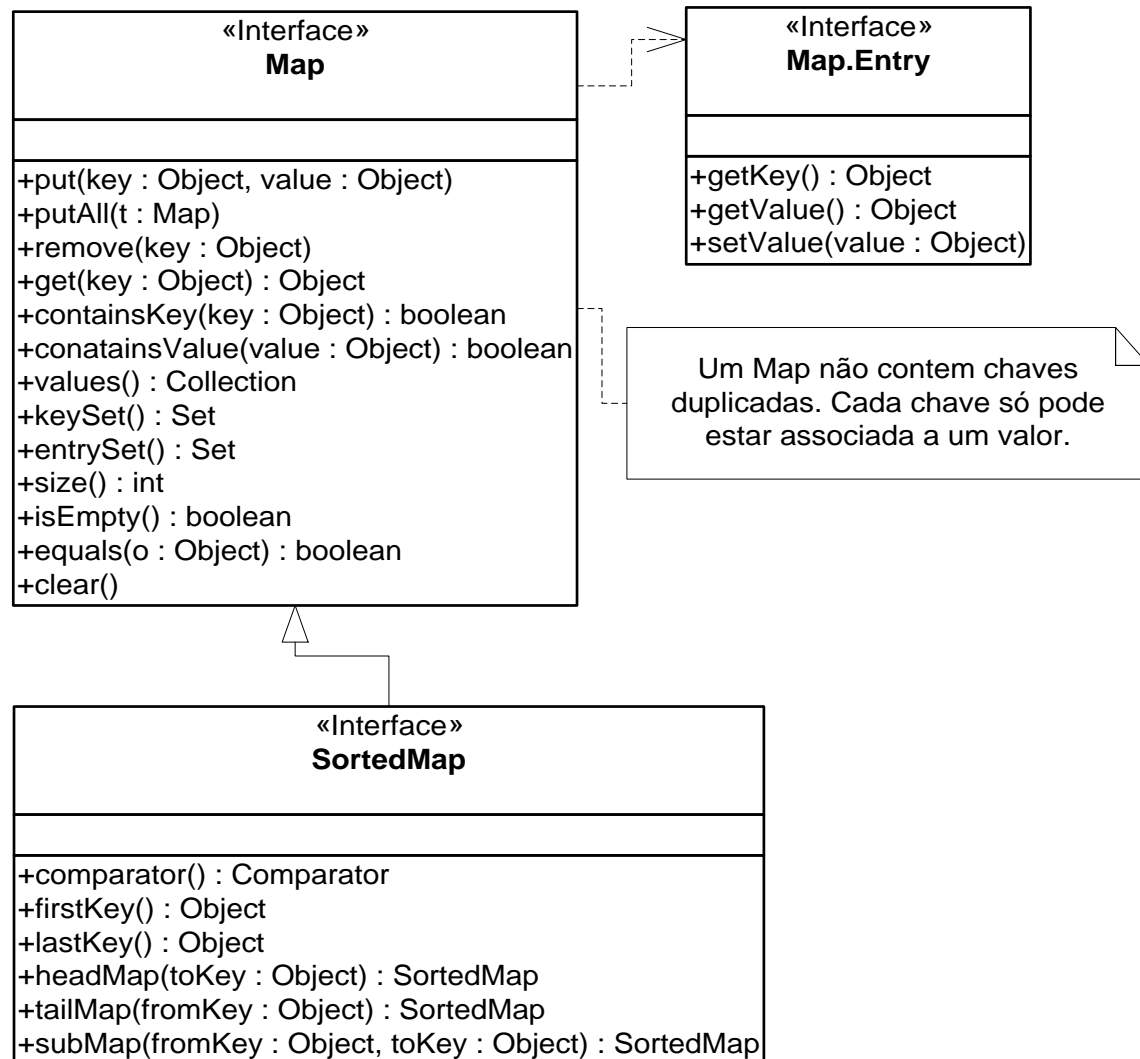
O "core" da *Collections Framework* são as *interfaces* usadas na manipulação das *Collections*:

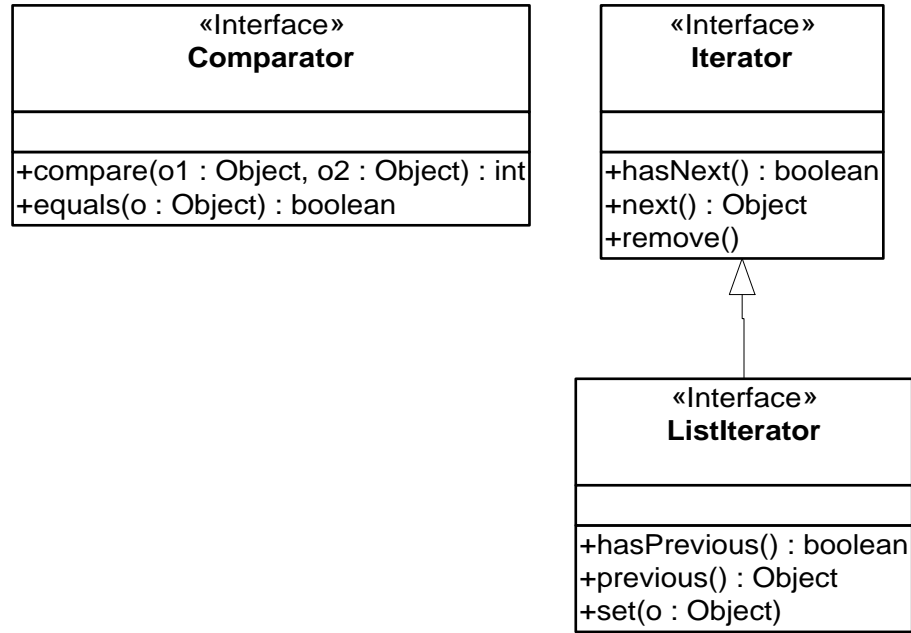


# ...Collections Framework - Interfaces




# ...Collections Framework - Interfaces





## ...Collections Framework - Classes

|   |      |                 |                 |               |             |
|---|------|-----------------|-----------------|---------------|-------------|
|  |      | Implementations |                 |               |             |
|   |      | Hash Table      | Resizable Array | Balanced Tree | Linked List |
| Interfaces  | Set  | HashSet         |                 | TreeSet       |             |
|   | List |                 | ArrayList       |               | LinkedList  |
|   | Map  | HashMap         |                 | TreeMap       |             |

### Wrapper Implementations - Collections:

```

static int binarySearch(List list, Object key)
static int binarySearch(List list, Object key, Comparator c)
static void copy(List dest, List src)
static Enumeration enumeration(Collection c)
static void fill(List list, Object o)
static Object max(Collection coll)
static Object max(Collection coll, Comparator comp)
static Object min(Collection coll)
static Object min(Collection coll, Comparator comp)
static List nCopies(int n, Object o)
static void reverse(List l)
static Comparator reverseOrder()
static void shuffle(List list)
static void shuffle(List list, Random rnd)
static Set singleton(Object o)
static List singletonList(Object o)

```

```

static Map singletonMap(Object key, Object value)
static void sort(List list)
static void sort(List list, Comparator c)
static Collection synchronizedCollection(Collection c)
static List synchronizedList(List list)
static Map synchronizedMap(Map m)
static Set synchronizedSet(Set s)
static SortedMap synchronizedSortedMap(SortedMap m)
static SortedSet synchronizedSortedSet(SortedSet s)
static Collection unmodifiableCollection(Collection c)
static List unmodifiableList(List list)
static Map unmodifiableMap(Map m)
static Set unmodifiableSet(Set s)
static SortedMap unmodifiableSortedMap(SortedMap m)
static SortedSet unmodifiableSortedSet(SortedSet s)

```

# Problema: Número de ocorrências das palavras num ficheiro de texto

```

WordCounter
package pg2.aula07;
import java.util.HashMap;
import pg2.io.*;
import java.io.*;

class Counter {
    int i = 1;
    public String toString() {
        return Integer.toString(i) + "\n";
    }
}

```

```

WordCounter
}

public class WordCounter {
    public static void main (String args[]){
        In cin = null;
        int tries = 0;
        String file;
        if (args.length > 0){
            file = args[0];
            tries = 3;
        }
        else {
            IO.cout.writeln("\nIntroduza o caminho e o nome para "
                + "o ficheiro de entrada:");
            file = IO.cin.readLine();
        }
        while (cin == null){
            try {
                cin = new In(file);
            }
            catch ( IOException e){
                IO.cout.writeln("\n" + e +
                    "\n > Nome de ficheiro ou caminho invalido.\n");
                if (tries++ > 2)
                    System.exit(1);
                IO.cout.writeln("Introduza o caminho e o nome para "
                    + "o ficheiro de entrada:");
                file = IO.cin.readLine();
            }
        }
    }
}

```

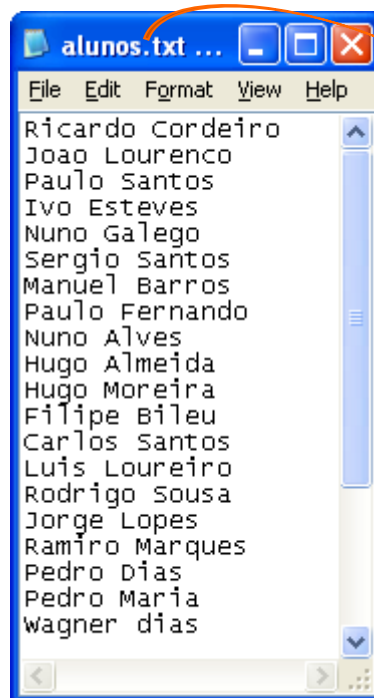


## ... Problema: Número de ocorrências das palavras num ficheiro de texto

```
public class WordCounter {
    public static void main (String args[]){
```

...

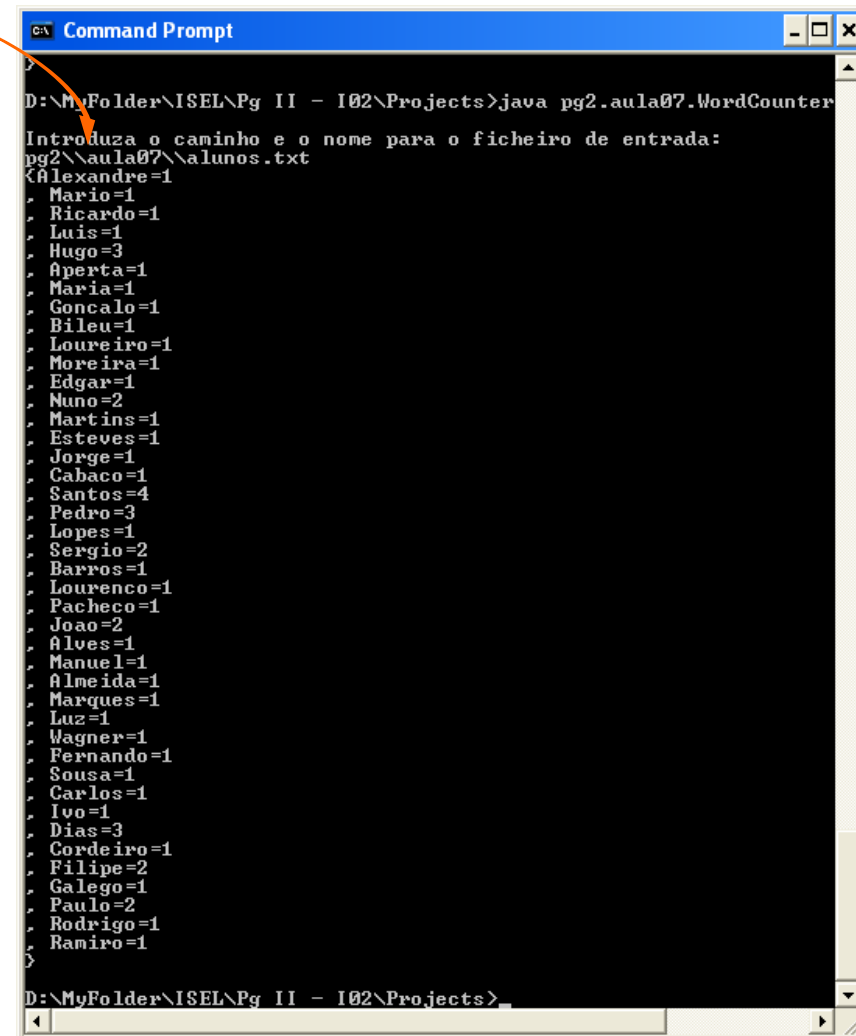
```
    HashMap hm = new HashMap(5);
    while (!cin.eof()) {
        String s = cin.readWord();
        if (hm.containsKey(s))
            ((Counter) hm.get(s)).i++;
        else
            hm.put(s, new Counter());
    }
    IO.cout.writeln(hm.toString());
}
```



alunos.txt ...

File Edit Format View Help

Ricardo Cordeiro  
Joao Lourenco  
Paulo Santos  
Ivo Esteves  
Nuno Galego  
Sergio Santos  
Manuel Barros  
Paulo Fernando  
Nuno Alves  
Hugo Almeida  
Hugo Moreira  
Filipe Bileu  
Carlos Santos  
Luis Loureiro  
Rodrigo Sousa  
Jorge Lopes  
Ramiro Marques  
Pedro Dias  
Pedro Maria  
Wagner dias



Command Prompt

D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula07.WordCounter

Introduza o caminho e o nome para o ficheiro de entrada:  
pg2\aula07\alunos.txt

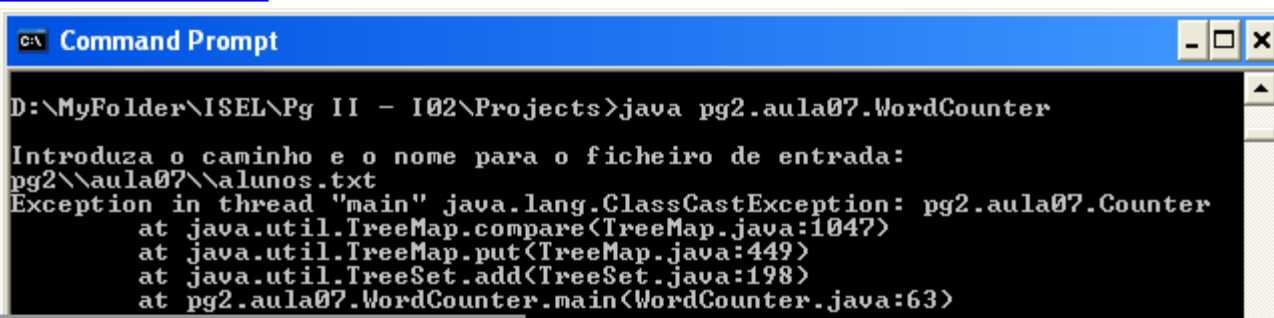
(A)lexandre=1  
Mario=1  
Ricardo=1  
Luis=1  
Hugo=3  
Aperta=1  
Maria=1  
Goncalo=1  
Bileu=1  
Loureiro=1  
Moreira=1  
Edgar=1  
Nuno=2  
Martins=1  
Esteves=1  
Jorge=1  
Cabaco=1  
Santos=4  
Pedro=3  
Lopes=1  
Sergio=2  
Barros=1  
Lourenco=1  
Pacheco=1  
Joao=2  
Alves=1  
Manuel=1  
Almeida=1  
Marques=1  
Luz=1  
Wagner=1  
Fernando=1  
Sousa=1  
Carlos=1  
Ivo=1  
Dias=3  
Cordeiro=1  
Filipe=2  
Galego=1  
Paulo=2  
Rodrigo=1  
Ramiro=1

D:\MyFolder\ISEL\Pg II - I02\Projects>

## Problema: agrupar as palavras por número de ocorrências

**Objectivo:** A partir do contentor ("hm") instanciado no problema anterior obter um novo contentor que agrupe, ordenadamente, as palavras por número de ocorrências.

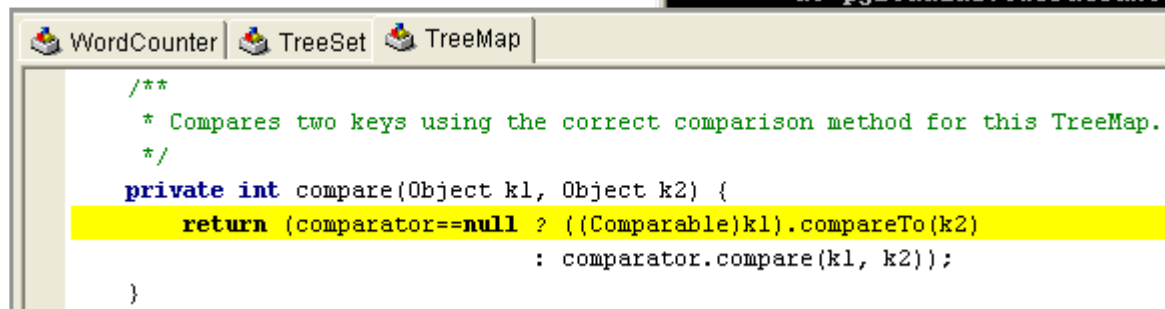
```
TreeSet st=new TreeSet();
for (Iterator i=hm.values().iterator(); i.hasNext();)
    st.add(i.next());
```



```
C:\ Command Prompt

D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula07.WordCounter

Introduza o caminho e o nome para o ficheiro de entrada:
pg2\aula07\alunos.txt
Exception in thread "main" java.lang.ClassCastException: pg2.aula07.Counter
    at java.util.TreeMap.compare(TreeMap.java:1047)
    at java.util.TreeMap.put(TreeMap.java:449)
    at java.util.TreeSet.add(TreeSet.java:198)
    at pg2.aula07.WordCounter.main(WordCounter.java:63)
```



```
WordCounter | TreeSet | TreeMap
/**
 * Compares two keys using the correct comparison method for this TreeMap.
 */
private int compare(Object k1, Object k2) {
    return (comparator==null ? ((Comparable)k1).compareTo(k2)
        : comparator.compare(k1, k2));
}
```

Ao usarmos um contentor ordenado tem que existir um critério de ordenação para os seus elementos, dado pela sua ordem natural (interface *Comparable*) ou por um outro critério dado no momento da instanciação do contentor (*Comparator*).

## ... Problema: agrupar as palavras por número de ocorrências

Neste caso optamos por introduzir na classe Counter a interface Comparable:

```
class Counter implements Comparable {
    int i = 1;

    public String toString() {
        return "\n" + Integer.toString(i);
    }

    public int compareTo(Object o) {
        Counter aux = (Counter) o;
        return aux.i == i ? 0 :
            (aux.i < i ? 1 : -1);
    }
}
```

Um método mais eficiente de obter um novo contentor ordenado pelo número de ocorrências seria através da utilização directa do construtor:

```
TreeSet st = new TreeSet(hm.values());
```

### Constructor Summary

#### TreeSet ()

Constructs a new, empty set, sorted according to the elements' natural order.

#### TreeSet (Collection c)

Constructs a new set containing the elements in the specified collection, sorted according to the elements' *natural order*.

#### TreeSet (Comparator c)

Constructs a new, empty set, sorted according to the given comparator.

#### TreeSet (SortedSet s)

Constructs a new set containing the same elements as the given sorted set, sorted according to the same ordering.

## ... Problema: agrupar as palavras por número de ocorrências

Para obter o contendor final que responde ao requisito do problema, não é necessário obter num passo intermédio as chaves correspondentes ao número de ocorrências. O objectivo pode ser atingido da seguinte forma:

```
WordCounter
{
    TreeMap tm = new TreeMap();
    for (Iterator i=hm.entrySet().iterator(); i.hasNext();){
        Map.Entry entry = (Map.Entry) i.next();
        Object key = entry.getValue();
        StringBuffer value = (StringBuffer) tm.get(key);
        if (value == null)
            tm.put(key,new StringBuffer((String) entry.getKey()))
        else
            value.append(" " + entry.getKey());
    }
    IO.cout.writeln(tm);
}
```

```
Command Prompt
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula07.WordCounter
Introduza o caminho e o nome para o ficheiro de entrada:
pg2\aula07\alunos.txt
1=Lobato Freitas Daniel Santo Edgar Cristina Lopes Matias Pacheco Cabaco Carval
o Martins Rodrigo Massano Erra Aguiar Galego Fernandes Jose Teixeira da Almeida
Carapau Harish Moreira Sousa Bruno Saraiva Botelho Wagner Valadares Lajas Cruz
eitao Norberto Ana Frederico Rechena Loureiro Esteves Luis Mendes Rato Cordeiro
Tania Moura Catia Nelson Patricia Bicho Micael Bhatt Mario Aperta Samuel Barros
Alves Andrade Rui Paulino Angelo Maria Nunes Bileu Cabral Fernando Lourenco Ram
ro Rocha Alexandre Ivo dias Graca Luz,
2=Sergio Filipe Silva Jorge Dias Diogo Marques Carlos Pereira Tiago Manuel,
3=Ferreira Goncalo Paulo,
4=Joao Ricardo,
5=Pedro Nuno,
6=Santos Hugo>
```

## ... Problema: agrupar as palavras por número de ocorrências

**Problema: Quais as alterações necessárias se quisermos apresentar o resultado anterior por ordem inversa ou seja, do maior para o menor número de ocorrências ??**

```
WordCounter CmpInvCounter Comparator

package pg2.aula07;
import java.util.Comparator;

public class CmpInvCounter implements Comparator {
    public int compare(Object o1, Object o2){
        return (-1)*((Counter) o1).compareTo(o2);
    }
}
```

```
WordCounter CmpInvCounter Comparator

TreeMap tm = new TreeMap(new CmpInvCounter());
for (Iterator i=hm.entrySet().iterator(); i.hasNext();){
    Map.Entry entry = (Map.Entry) i.next();
    Object key = entry.getValue();
    StringBuffer value = (StringBuffer) tm.get(key);
    if (value == null)
        tm.put(key, new StringBuffer((String) entry.getKey()));
    else
        value.append(" " + entry.getKey());
}
IO.cout.writeln(tm);
```

```
Command Prompt

D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula07.WordCounter

Introduza o caminho e o nome para o ficheiro de entrada:
pg2\aula07\alunos.txt
{
6=Santos Hugo,
5=Pedro Nuno,
4=Joao Ricardo,
3=Ferreira Goncalo Paulo,
2=Sergio Filipe Silva Jorge Dias Diogo Marques Carlos Pereira Tiago Manuel,
1=Lobato Freitas Daniel Santo Edgar Cristina Lopes Matias Pacheco Cabaco Carvalh
o Martins Rodrigo Massano Erra Aguiar Galego Fernandes Jose Teixeira da Almeida
Carapau Harish Moreira Sousa Bruno Saraiva Botelho Wagner Valadares Lajas Cruz L
eitao Norberto Ana Frederico Rechena Loureiro Esteves Luis Mendes Rato Cordeiro
Tania Moura Catia Nelson Patricia Bicho Micael Bhatt Mario Aperta Samuel Barros
Alves Andrade Rui Paulino Angelo Maria Nunes Bileu Cabral Fernando Lourenco Rami
ro Rocha Alexandre Ivo dias Graca Luz}
```

- class java.lang.Exception
  - class java.lang.RuntimeException
    - class java.util.ConcurrentModificationException
    - class java.util.EmptyStackException
    - class java.util.MissingResourceException
    - class java.util.NoSuchElementException
  - class java.util.TooManyListenersException

Dar Load Factor

Classe Collections e Arrays

Fazer um outro exercício com um dos Comparadores já existentes.

`String.Comparator`