java 2: 29 (exemples), 48, 53,54

RESUM: 63-71

Les classes contenidores tenen variables útils de classe.

Integer.MAX\_VALUE, Integer.MIN\_VALUE

Double.MAX\_VALUE, Double.MIN\_VALUE, Double.NaN,

Double.NEGATIVE\_INFINITY, Double.POSITIVE\_INFINITY

També tenen mètodes útils de classe

Double.parseDouble (String) ‐ converteix una cadena a un doble

Integer.parseInt (String) ‐ converteix una cadena a un enter

--------

 ArrayList<T>: Implementació de la llista

amb una taula redimensionable. Accés més rapid.

 LinkedList<T>: Implementació de la llista

amb punters. Insercions i eliminacions més

ràpides

List<String> noms = new LinkedList<String>;

 Diverses implementacions de la interfície

Map<T>:

 HashMap: Implementació del Map amb taules

de hast. [Més usat]

 TreeMap:Implementació del Map amb arbres

balancejats. Manté un ordre.

 Exemple:

Map<String,Integer> noms = new HashMap<String,Integer>;

Map map = new HashMap();

Pag 53,54

Map map = new HashMap(); // instantiate a concrete map

// ...

map.put(key, val); // insert a key‐value pair

// ...

// get the value associated with key

Object val = map.get(key);

map.remove(key); // remove a key‐value pair

// ...

if (map.containsValue(val)) { ... }

if (map.containsKey(kay)) { ... }

Set keys = map.keySet(); // get the set of keys

// iterate through the set of keys

Iterator iter = keys.iterator();

while (iter.hasNext()) {

Key key = (Key) iter.next();

// ...

}

HashMap<String, Integer> frequency(String[] names) {

HashMap<String, Integer> frequency =

new HashMap<String, Integer>();

for(String name : names) {

Integer currentCount = frequency.get(name);

if(currentCount == null) {

currentCount = 0; }

frequency.put(name, ++currentCount);

}

return frequency;

}

public class Person {

public String name;

boolean equals(Object o) {

return (o instanceof Person && ((Person)o).name.equals(name));

}

public int hashCode() {

return name.hashCode();

}

}

iteeeeeeeeraaaaatooorssss

static void printAll (Collection coll) {

Iterator iter = coll.iterator( );

while (iter.hasNext( )) {

System.out.println(iter.next( ) );

}

}

 Mètodes implementats

boolean hasNext( );

Object next( );

void remove( );

 Mètodes implementats addicionals pels

ListIteration

boolean hasPrevious();

Object previous();

int nextIndex( );

int previousIndex( );

 Hi ha constructores de iteradors específics per llistes:

ListIterator listIterator( );

ListIterator listIterator(int index);

la darrera comença en la posició index

Són mètodes estàtics que implementen algorismes

usuals en llistes:

 sort: Ordena fent servir mergesort

 shuffle: Barreja aleatòriament els elements de la llista

 reverse: Inverteix l’ordre

 rotate: Rota la llista el nombre de vegades que vulguem

 fill: Reescriu tots els elements de la llista amb l’element que volem.

 copy: Copia una llista sobre una altra

 binarySearch: Cerca fent servir cerca binaria

 indexOfSubList: Retorna el índex de la primera subllista trobada

 lastIndexOfSubList: Retorna el índex de la darrera subllista trobada

 max: Retorna el màxim de la llista

 min: Retorna el mínim de la llista

 Negatiu: quan el primer sigui menor que el segon

 Zero: quan siguin iguals

 Positiu

public class CountComparator implements Comparator {

public int compare(Object o1, Object o2) {

if (o1 != null &&

o2 != null &&

o1 instanceof Count &&

o2 instanceof Count) {

Count c1 = (Count) o1;

Count c2 = (Count) o2;

return (c2.i ‐ c1.i);

} else {

return 0;

}

}

}

public class WordFrequency4 {

static public void main(String[] args) {

....

List list = new ArrayList(words.values());

Collections.sort(list, new CountComparator());

Iterator iter = list.iterator();

while (iter.hasNext()) {

count = (Count) iter.next();

word = count.word;

System.out.println(word +

(word.length() < 8 ? "\t\t" : "\t") +

count.i);

}

}

}