```java
/**
package dataStructures;

/**
 * @author Ricardo Gaspar Nr. 35277
 * @author Hugo António Nr. 34334 Turno P2 Docente Vasco Amaral
 */
public class SepChainHashTableIterator<K, V> implements
Iterator<Entry<K, V>> {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    /**
     * Tabela de dicionários (entries) a iterar.
     */
    private Dictionary<K, V>[] table;

    /**
     * Apontador para posição actual na tabela de dicionários.
     */
    private int current;

    /**
     * Iterador actual.
     */
    private Iterator<Entry<K, V>> it;

    /**
     * Construtor da SepChainHashTableIterator. Cria e inicializa
um iterador
     * para uma dada tabela.
     *
     * @param table
     *            Tabela de disperão para a qual ser· criado o
iterador.
     */
    public SepChainHashTableIterator(Dictionary<K, V>[] table) {

```

```
41          this.table = table;
42          this.rewind();
43
44      }
45
46      /*
47       * (non-Javadoc)
48       *
49       * @see dataStructures.Iterator#hasNext()
50       */
51      @Override
52      public boolean hasNext() {
53          if (it == null)
54              return false;
55          return it.hasNext();
56      }
57
58      /*
59       * (non-Javadoc)
60       *
61       * @see dataStructures.Iterator#next()
62       */
63      @Override
64      public Entry<K, V> next() throws NoSuchElementException {
65
66          if (!hasNext())
67              throw new NoSuchElementException();
68
69          Entry<K, V> entry = it.next();
70          if (!hasNext())
71              // É necessário avançar o current, pois esta posição na
  tabela já
72              // foi iterada
73              nextIterator(++current);
74
75          return entry;
76      }
77
78      /*
79       * (non-Javadoc)
80       *
```

```
81      * @see dataStructures.Iterator#rewind()
82      */
83     @Override
84     public void rewind() {
85         current = 0;
86         //Inicializa o it com o próximo iterador de uma lista não
   vazia.
87         nextIterator(current);
88 
89     }
90 
91     /**
92      * Avança para o próximo iterador cuja lista é não vazia. É
   necessário
93      * indicar a posição de partida.
94      *
95      * @param startPosition
96      *             Posição de partida.
97      */
98     private void nextIterator(int startPosition) {
99         // Se a table for vazia ou se não houver próximo.
100        it = null;
101        for (int i = startPosition; i < table.length; i++) {
102            if (!table[i].isEmpty()) {
103                it = table[i].iterator();
104                current = i;
105                break;
106            }
107        }
108    }
109
110}
111
```