```java
/**
package dataStructures;

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

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    protected BSTNode<K, V> root;
    protected Stack<BSTNode<K, V>> path;

    /**
     *
     */
    public BSTInorderIterator(BSTNode<K, V> root) {

        this.root = root;
        rewind();
    }

    @Override
    public boolean hasNext() {

        return (!path.isEmpty());
    }

    @Override
    public Entry<K, V> next() throws NoSuchElementException {

        if (!hasNext())
            throw new NoSuchElementException();

        BSTNode<K, V> node = path.pop();
        Entry<K, V> entryToReturn = node.getEntry();
```

```
43          // Se tem filho direito, achar o mínimo desse filho e
  empilhar
44          if (node.getRight() != null)
45              minNode(node.getRight());
46
47          return entryToReturn;
48      }
49
50      @Override
51      public void rewind() {
52
53          path = new StackInList<BSTNode<K, V>>();
54          minNode(root);
55      }
56
57      /**
58       * Realiza o percurso até ao nó cuja chave tem o valor mínimo.
  Empilha os
59       * nós por onde vai passando até chegar ao fim.
60       *
61       * @param node
62       *             Nó de partida.
63       * @return Nó cujo valor da chave é mínimo.
64       */
65      private void minNode(BSTNode<K, V> node) {
66
67          while (node != null) {
68              path.push(node);
69              node = node.getLeft();
70          }
71      }
72  }
73
```