```java
import java.util.Comparator;
import java.util.LinkedList;
import java.util.Queue;

public class BTree<E>
{
    private Node<E> root;
    private Comparator<E> comparator;
    private int order;

    /**
     * Creates an empty tree from given order and comparator
     * @param theOrder - given order of the tree
     * @param theComp - given comparator of the tree
     */
    public BTree(int theOrder, Comparator<E> theComp)
    {
        this.order = theOrder;
        this.comparator = theComp;
        this.root = new Node<>(theOrder, theComp);
    }

    /**
     * Adds the given item into the tree
     * @param item - item to be added into the tree
     */
    public void add(E item)
    {

    }

    /**
     * Finds the leaf node in the tree rooted at the given node
     * @param curr - the node to start the traversal from
     * @param item - the item should be inserted at the node
     * @return the leaf node in the tree
     */
    private Node<E> findLeaf(Node<E> curr, E item)
    {
        return curr;

    }

    /**
     * Determines if the tree contains the given item
     * @param item - the item to be determined if the tree contain
     * @return true if the tree contains the item
     */
    public boolean contains(E item)
    {
        return findNode(root, item) != null;
    }

    /**
     * Finds the node containing the specified item if it exists in tree
     * @param curr - the node to start the traversal from
     * @param item - the item to be found in the tree
     * @return the node containing the specified item if it exists in tree
     */
    private Node<E> findNode(Node<E> curr, E item)
    {
        return curr;

    }

    /**
     * Performs inorder traversal of the tree
     * @param visitor - given visitor to start traverse inorder through
     */
    public void inorder(Visitor<E> visitor) {
        inorder(visitor, root);
    }

    /**
```

```java
     * Performs inorder traversal of the tree
     * @param visitor - given visitor to start traverse inorder through
     * @param curr - node where the traversal start from
     */
    private void inorder(Visitor<E> visitor, Node<E> curr) {

    }

    /**
     * Returns a string representation of this tree in sorted order
     * @return a string representation of this tree in sorted order
     */
    public String toStringSorted() {
        return "";
    }

    /**
     * Returns a string representation of this tree in level-order traversal
     * @return a string representation of this tree in level-order traversal
     */
    public String toString() {
        return "";
    }
}

```