```
class TreeNode {
    String key;
    int priority;
    TreeNode left, right;

    public TreeNode(String key, int priority) {
        this.key = key;
        this.priority = priority;
        this.left = this.right = null;
    }
}

public class project5 {
    private TreeNode root;

    public PriorityQueueBST() {
        this.root = null;
    }

    public void insert(String key, int priority) {
        root = insertRec(root, key, priority);
    }

    private TreeNode insertRec(TreeNode root, String key, int priority) {
        if (root == null) {
            return new TreeNode(key, priority);
        }

        if (priority < root.priority) {
            root.left = insertRec(root.left, key, priority);
        } else {
            root.right = insertRec(root.right, key, priority);
        }

        return root;
    }

    public String removeMax() {
        if (root == null) {
            return null;
        }

        TreeNode maxNode = findMax(root);
        root = removeMax(root, maxNode.key);
        return maxNode.key;
    }

    private TreeNode findMax(TreeNode root) {
        while (root.right != null) {
            root = root.right;
        }
        return root;
    }

    private TreeNode removeMax(TreeNode root, String key) {
```

```
        if (root == null) {
            return null;
        }

        if (key.compareTo(root.key) < 0) {
            root.left = removeMax(root.left, key);
        } else if (key.compareTo(root.key) > 0) {
            root.right = removeMax(root.right, key);
        } else {
            if (root.left == null) {
                return root.right;
            } else if (root.right == null) {
                return root.left;
            }

            TreeNode temp = findMax(root.left);
            root.key = temp.key;
            root.priority = temp.priority;
            root.left = removeMax(root.left, temp.key);
        }

        return root;
    }

    public String getMax() {
        if (root == null) {
            return null;
        }

        return findMax(root).key;
    }

    public static void main(String[] args) {
        PriorityQueueBST pq = new PriorityQueueBST();
        pq.insert("Task1", 3);
        pq.insert("Task2", 1);
        pq.insert("Task3", 2);

        System.out.println("Current Max Priority Task: " + pq.getMax());  // Output: Task2
        System.out.println("Removing Max Priority Task: " + pq.removeMax());  // Output: Task2
        System.out.println("Current Max Priority Task after removal: " + pq.getMax());  // Output: Task3
    }
}
```