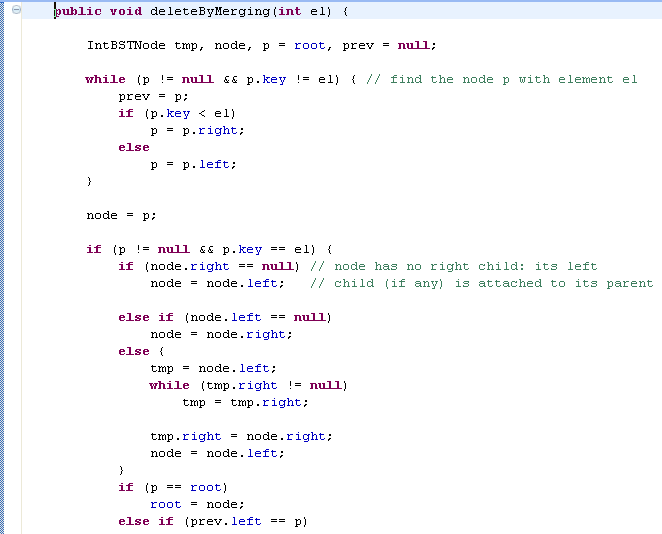
******

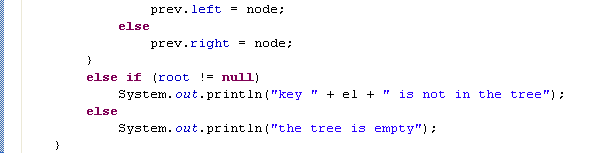
***Hands-On Exercise 7.4 [20-points]: Deleting a node in a binary tree (:15 min)***

### *Instructions:*

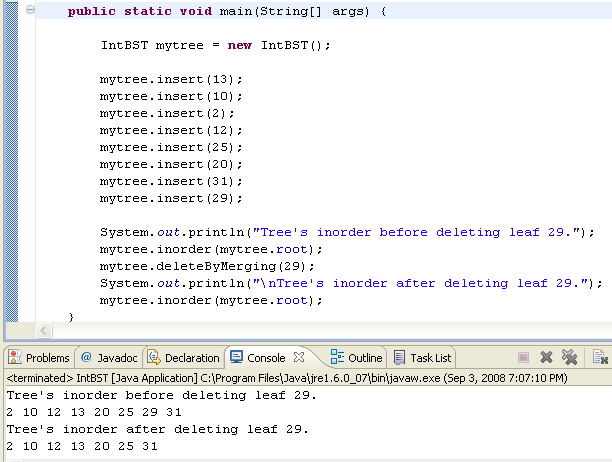
* According to some research, industry values documentation, and excellent written and oral communication skills. The purpose of this part of the class is to encourage you to gain these skills.
* Backup your work to your USB drive for this material may come out as part of your examination.
* Make a copy of this entire document and add your work into it.
* Submit to Blackboard at the same link where you got this document.
* Points will be a deducted if submitted on the wrong place, or if these instructions are not followed.

1.❑ Add the deleteByMerging() method to the IntBST class you have created before:

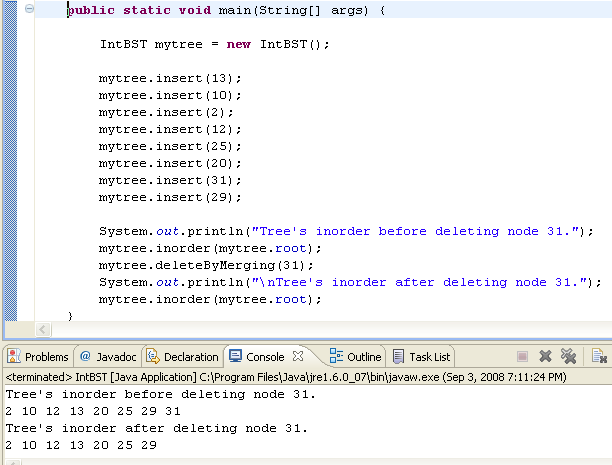




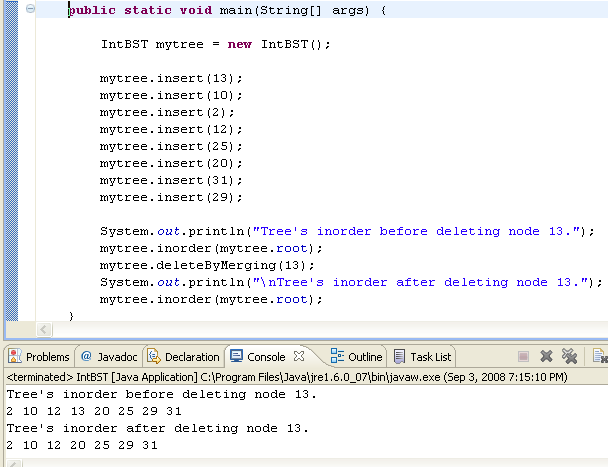
2.❑ Test the new method by deleting a leaf.



3.❑ Test the new method by deleting a node with only one child.



4.❑ Test the new method by deleting a node with two children.



5.❑ Paste your code here.

**public** **void** deleteByMerging(**int** el){

IntBSTNode tmp, node, p = root, prev = **null**;

**while** (p != **null** && p.key != el){

prev = p;

**if**(p.key < el)

p = p.right;

**else**

p = p.left;

}

node = p;

**if**(p != **null** && p.key == el){

**if**(node.right == **null**)

node = node.left;

**else** **if**(node.left == **null**)

node = node.right;

**else** {

tmp = node.left;

**while** (tmp.right != **null**)

tmp = tmp.right;

tmp.right = node.right;

node = node.left;

}

**if**(p == root)

root = node;

**else** **if** (prev.left == p)

prev.left = node;

**else**

prev.right = node;

}

**else** **if**(root != **null**)

System.***out***.println("key " + el + " is not in the tree");

**else**

System.***out***.println("the tree is empty");

}

**package** Excercise7\_1;

**public** **class** Driver {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

IntBST mytree = **new** IntBST();

mytree.insert(13);

mytree.insert(10);

mytree.insert(2);

mytree.insert(12);

mytree.insert(25);

mytree.insert(20);

mytree.insert(31);

mytree.insert(29);

System.***out***.println("Trees in order before deleting leaf 29");

mytree.inorder(mytree.root);

mytree.deleteByMerging(29);

System.***out***.println("\nTrees in order after deleteing leaf 29");

mytree.inorder(mytree.root);

mytree.deleteByMerging(31);

System.***out***.println("\nTrees in order after deleteing leaf 31");

mytree.inorder(mytree.root);

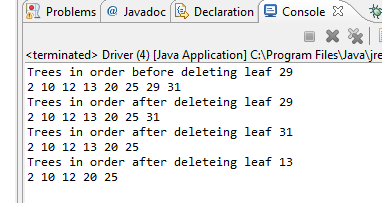
mytree.deleteByMerging(13);

System.***out***.println("\nTrees in order after deleteing leaf 13");

mytree.inorder(mytree.root);

}

6.❑ Paste your screen shot output here [Ctrl] + [PrtScn]. Make sure you magnified it. Paste one screenshot for each test case (a total of 3 screenshots).



7.❑ Write your topmost question regarding this topic.

Is there a situation where binary trees are not the best choice?

8.❑ **Critical Thinking:** If you are asked to make a test question based on this topic, what would be the question and what is your answer?

True of false: In a binary tree node deletion involving both left and right child nodes, the resulting left tree is made the child of the resulting right tree.

Answer: false

[](http://images.google.com/imgres?imgurl=www.skyscript.co.uk/im/trophy.jpg&imgrefurl=http://www.skyscript.co.uk/im/&h=214&w=180&sz=6&tbnid=ECCiP8U-7NsJ:&tbnh=99&tbnw=84&prev=/images?q=trophy&svnum=10&hl=en&lr=&ie=UTF-8&oe=UTF-8&sa=G)Congratulations! You’ve just learned how to delete a node in a binary tree.

**Submission Procedure**

1. Write your **name** here: \_\_\_Joshua LeGoff\_\_\_\_\_\_\_\_\_\_
2. Date: \_\_\_11/10/2015\_\_\_\_\_\_\_\_\_\_\_\_\_
3. **Backup** your work to your USB drive, this material may come out as part of your exam.
4. **Submit** to Blackboard at the link where you got it.

**Note:**

* Submit back to Blackboard where you get it.
* 2-points deduction if you submit it on the wrong place.
* 2-points deduction if you did not follow these instructions.
* Make sure you submit it at the correct location where you got it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GRADING RUBRIC | | | | |
| Grading Criteria | 3  **Exceeds**  *Excellent*  Epic Wow | 2  **Meets**  *Satisfactory*  O.K. | 1  **Partially Meets**  *Below Expectations*  Not Yet | 0  **Does Not Meet**  *Unacceptable*  Fail |
| **Completeness** | +5-Completed all the required work and added more examples. | +2-Completed all the work required. | +1-Partially completed the work required. | Unfortunately, did not complete the work required. |
| **Coding** | +10- Code is excellent, comments are added, and different techniques were used. | +7-Code is O.K., and program works. | +4-Code works, but still needs improvement. | Unfortunately, no coding. |
| **Output** | +5-Outputs are correct, and provided additional output cases. | +2-Output meets requirement and is readable. | +1-There is output, but not readable, and/or needs improvement. | Unfortunately, no output. |
| **Late** | Excellent, you submitted it before the deadline. | -5, unfortunately for submitting after the deadline. | -7, unfortunately for submitting several weeks after the deadline. | -10, unfortunately, for submitting very late. |