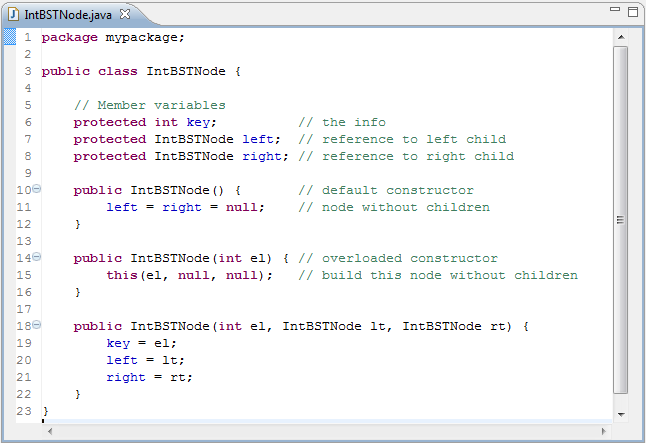
******

***Hands-On Exercise 7.1 [20-points]: Searching a node in a binary tree (:15 min)***

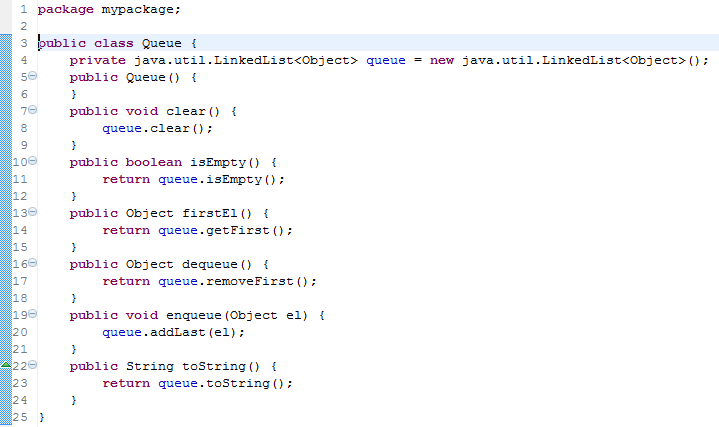
### *Instructions:*

* 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.

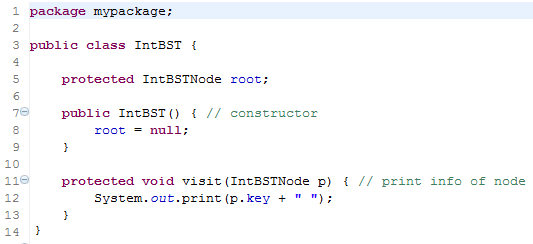
1.❑ Create the Node class below:



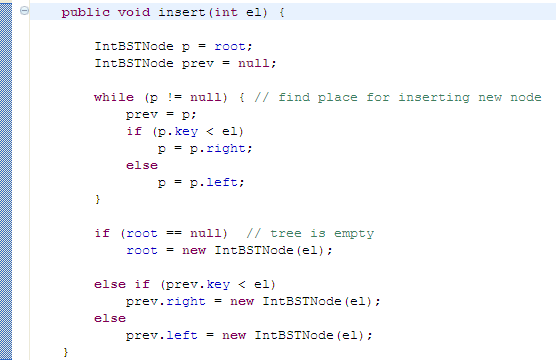
2.❑ Create the Queue class below:



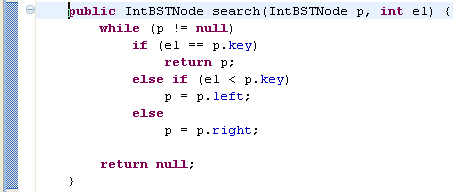
3.❑ Create the IntBST class below:



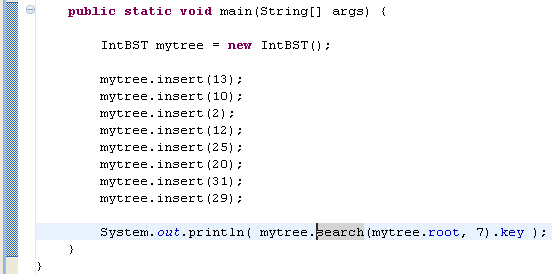
4.❑ Add the insert method to the class above.



5.❑ Add the search() method to the IntBST class you have created above:



6.❑ Add the main() method to the IntBST class you have created above and run to search for node with value 7.



7.❑ What happened if the node you’re searching is not in the binary tree?

A NullPointException is thrown.

8.❑ Paste your code here.

**package** Excercise7\_1;

**public** **class** IntBSTNode {

**protected** **int** key;

**protected** IntBSTNode left;

**protected** IntBSTNode right;

**public** IntBSTNode(){

left = right = **null**;

}

**public** IntBSTNode(**int** el){

**this**(el, **null**, **null**);

}

**public** IntBSTNode(**int** el, IntBSTNode lt, IntBSTNode rt){

key = el;

left = lt;

right = rt;

}

}

**package** Excercise7\_1;

**public** **class** IntBST {

**protected** IntBSTNode root;

**public** IntBST(){

root = **null**;

}

**protected** **void** visit(IntBSTNode p){

System.***out***.print(p.key + " ");

}

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

IntBSTNode p = root;

IntBSTNode prev = **null**;

**while** (p != **null**){

prev = p;

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

p = p.right;

**else**

p = p.left;

}

**if**(root == **null**)

root = **new** IntBSTNode(el);

**else** **if**(prev.key < el)

prev.right = **new** IntBSTNode(el);

**else**

prev.left = **new** IntBSTNode(el);

}

**public** IntBSTNode search (IntBSTNode p, **int** el){

**while** (p != **null**){

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

**return** p;

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

p = p.left;

**else**

p = p.right;

}

**return** **null**;

}

}

**package** Excercise7\_1;

**public** **class** Queue {

**private** java.util.LinkedList<Object> queue = **new** java.util.LinkedList<Object>();

**public** Queue(){

}

**public** **void** clear(){

queue.clear();

}

**public** **boolean** isEmpty(){

**return** queue.isEmpty();

}

**public** Object firstEl(){

**return** queue.getFirst();

}

**public** Object dequeue(){

**return** queue.removeFirst();

}

**public** **void** enqueue(Object el){

queue.addLast(el);

}

**public** String toString(){

**return** queue.toString();

}

}

**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);

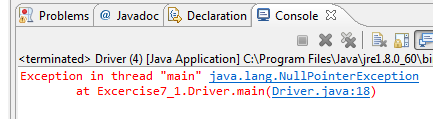
mytree.insert(29);

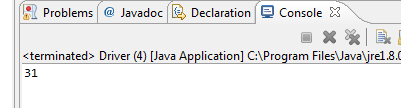
System.***out***.println(mytree.search(mytree.root, 7).key);

}

}

9.❑ Paste your screen shot output here [Ctrl] + [PrtScn]. Make sure you magnified it.





10.❑ Write your topmost question regarding this topic.

Are binary search trees the fastest data structure?

11.❑ **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?

What costs the most time during I/O?

Answer: Seek Time and Latency(Rotational Delay)

[](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 search 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. |