

Assignments

Assignment - In progress

Add attachment(s), then choose the appropriate button at the bottom.

In progress
Submitted
Returned
Title
A09-Binary Search Tree
Due
Apr 10, 2022 11:55 PM
Number of resubmissions allowed
Unlimited
Accept Resubmission Until
Apr 13, 2022 11:55 PM
Status
Not Started
Grade Scale
Points (max 270.00)

Instructions

Binary Search Tree

You will implement a BST. To make this easier you will be using the attached CD.java and CDException.java classes. **They should NOT be modified.**

You must submit the following classes

BSTNode.java which should implement the following Interface

```
public interface BSTNodeI{

    /**
     * This method will set newNode as
     * the left child of the current node
     * @param BSTNode new Node to add into a BST
     */
    abstract void setLeftChild(BSTNode newNode);
    /**
     * This method will set newNode as
     * the right child of the current node
     * @param BSTNode new Node to add into a BST
     */
    abstract void setRightChild(BSTNode newNode);
    /**
     * This method will return the left child
     * of the current node
     * @param no paramteters
     * @return BSTNode left child
     */
    abstract BSTNode getLeftChild( );
    /**
     * This method will return the right child
     * of the current node
     * @param no paramteters
     * @return BSTNode right child
     */
    abstract BSTNode getRightChild( );

    /**
     * This method return the information
     * inside the current node CD
     * @param none
     * @return CD object
     */
    abstract CD getCD( );

    /**
     * This method return a printable version
     * of the CD inside the current node
     * @param none
     * @return String with the CD information
     */
    abstract String toString( );
```

}

CDBST.java which should implement the following interface

```
/**
 * CDBSTI Interface to be used for CDBST.java
 */
public interface CDBSTI {

    /**
     * Inserts node into binary search tree.
     * @param CD cd
     * @exception Any exception.
     * @return boolean.
     */
    abstract boolean addNode(CD cd);

    /**
     * Looks for a node within the binary search tree.
     * @param CD cd
     * @exception Any exception.
     * @return boolean.
     */
    abstract boolean findNode(CD cd);

    /**
     * Prints binary search tree in inorder
     * traversal using recursion.
     * @param BSTNode root.
     * @exception Any exception.
     * @return No return value.
     */
    abstract void printBSTree(BSTNode root);

    /**
     * Overloads method of the same name to
     * access private variable root.
     * @param None.
     * @exception Any exception.
     * @return No return value.
     */
    abstract void printBSTree( );
}
```

You may test your code using the following DriverClass.

I suggest creating a different driver class because I will be modifying it to test even more cases

```
//Driver class by Blanca Polo

public class CDBSTDriver{
    public static void main(String[ ] arg) throws Exception{
        final int SIZE = 6;
        CDBST tree = new CDBST( );
        CD cd1 = null;
        CD [ ] cdarr = new CD[SIZE];
        cdarr[0] = new CD ("Coldplay", "Album24", 24.99, 5);
        cdarr[1] = new CD ("Maroon 5", "Album2", 2.99, 5);
        cdarr[2] = new CD ("Ariana Grande", "Album43", 43.99, 5);
        cdarr[3] = new CD ("Madonna", "Album6", 6.99, 5);
        cdarr[4] = new CD ("Bon Jovi", "Album55", 55.12, 5);
        cdarr[5] = new CD ("Bon Jovi", "Album55", 55.12, 5);
        for(int i = 0; i< cdarr.length; i++){
            if(tree.addNode(cdarr[i])) {
                System.out.println("added");
                System.out.println(cdarr[i].toString( ));
                System.out.println("-----\n");
            }
            else{
                System.out.println("duplicate");
            }
        }
    }
}
```

```
System.out.println("\n=====\\n");
System.out.println("printing all CDs, inorder by price");
tree.printBSTree( );

System.out.println("\\n=====Find Albums=====");
System.out.println("=====+++++++=\\n");

CD cd2 = new CD ("Rocker", "dont find me", 12.09, 2);
if(tree.findNode(cd2)) {
    System.out.println("\\nfound\\n*****\\n");
}
else{
    System.out.println(cd2.toString( ));
    System.out.println("\\nnot found\\n*****\\n");
}
cd2 = new CD ("Madonna", "Album6", 6.99, 5);
if(tree.findNode(cd2)) {
    System.out.println(cd2.toString( ));
    System.out.println("found\\n====");
}
else{
    System.out.println(cd2.toString( ));
    System.out.println("not found\\n====");
}
}
```

If used as is, it will result in the following outcome:

```
---jGRASP exec: java CDBSTDriver
added
Album: Album24
Artist: Coldplay
Price: 24.99
Currently 5 in stock
-----

added
Album: Album2
Artist: Maroon 5
Price: 2.99
Currently 5 in stock
-----

added
Album: Album43
Artist: Ariana Grande
Price: 43.99
Currently 5 in stock
-----

added
Album: Album6
Artist: Madonna
Price: 6.99
Currently 5 in stock
-----

added
Album: Album55
Artist: Bon Jovi
Price: 55.12
Currently 5 in stock
-----

duplicate

=====

printing all CDs, inorder by price
Album: Album2
Artist: Maroon 5
Price: 2.99
Currently 5 in stock
```

Album: Album6
Artist: Madonna
Price: 6.99
Currently 5 in stock

Album: Album24
Artist: Coldplay
Price: 24.99
Currently 5 in stock

Album: Album43
Artist: Ariana Grande
Price: 43.99
Currently 5 in stock

Album: Album55
Artist: Bon Jovi
Price: 55.12
Currently 5 in stock

=====Find Albums=====

Album: dont find me
Artist: Rocker
Price: 12.09
Currently 2 in stock

not found

Album: Album6
Artist: Madonna
Price: 6.99
Currently 5 in stock
found
=====

Comments: Comment all your classes and methods. You may use the comments supplied in the interface code. Do not forget to add your name to your program code.

Program requirements and structure:

You will implement the BSTNodeI.java and the CDBSTI.java interfaces. **Submit only the BSTNode.java and CDBST.java** programs. I already have CD an CDEException, both the interfaces, as well as the driver class.

constructor	The constructors are not specified in the interface neither for the node class nor for the BST class. It is up to you how you design them. However for the BST you will need at least one PRIVATE instance variable, the root . No need to manipulate it when creating the tree but you will need a pointer to the root in order to make your code work. A reference to the root is the most important part.
CDBST public boolean addNode(CD cd) public boolean findNode(CD cd)	<ul style="list-style-type: none">• Since CD.java and CDEException.java do not allow the creation of invalid CDs we will not validate for that anymore.• To find or to insert the CD use the compareTo method in the CD class. Please read and understand it so you know how it works.• return true if element inserted or found, false if the element was a duplicate and was not inserted. The exact opposite will take place when looking for a node. If the element is found your code should return true and if not found it should return false.
public void printBSTree(BSTNode root) public void printBSTree()**	<pre>public void printBSTree(BSTNode root){ if(root != null){ printBSTree(root.getLeftChild()); System.out.println(print the CD info here); printBSTree(root.getRightChild()); } }</pre> <p><i>This method assumes to get the root as a parameter and will help you print your tree in inorder traversal. This method, it is good for debugging purposes. Please feel free to use it</i></p> <p>Overloaded method to access private data field root to send to recursive method of the same name. If you wish to implement this differently it is ok. The root of your tree should not be public.</p>
BSTNode methods	Please follow the interface given. Most of the methods are straight forward and you should be able to use your previous ICS211 programming experience to get these methods going. Note that there is not setCD method. This is on purpose to avoid anyone from changing the information inside a node and making the BST invalid. If we were implementing the removeNode method then the setCD method would be implemented but it must be private if that were the case.

You must implement the provided interfaces. You may add other methods but the ones specified here must be exactly as specified (except the last method)**

You may copy and paste the recursive printBSTree method into your BSTree class. If you need to add other methods in order to make this work add them and make sure you comment them well.
you will be graded as follows:

BSTNode class *(total 70 points)*
setLeftChild *(10 points)*
setRightChild *(10 points)*
getLeftChild *(10 points)*
getRightChild *(10 points)*
getCD *(10 points)*
toString *(20 points)*
Failure to implement the class will result in no points

CDBST class *(total 200 points)*
addNode *(100 points)*
findNode *(100 points)*
printBSTree *(0 points. The necessary methods are provided for you)*



Remember to **draw your code** to avoid mistakes!! Start ASAP!
You do NOT need a driver class.... but you may use the one provided to run and test your code. Feel free to modify the driver class just for testing purposes.

Learning Outcomes:

- For this assignment, you will practice/learn:
- Creating and using binary search trees.
 - Use binary tree traversals.
 - Practice your skills in creating and communicating several classes.

If you have any questions please e-mail your instructor: blanca@hawaii.edu or email Preston our Tutor!

Additional resources for assignment

-  [CDException.java](#) (1 KB; Nov 26, 2021 2:19 pm)
-  [CD.java](#) (3 KB; Nov 26, 2021 2:19 pm)

Submission
Attachments

No attachments yet


Select a file from computer No file chosen

Proceed

Preview

Save Draft

Cancel

 Don't forget to save or proceed!