Assignment 2 BST Nov. 2, 2015

Class BST

java.lang.Object
    BST

```
public class BST
extends java.lang.Object
```

Binary Search Tree class of Albums.

The purpose of this exercise is to give you practice with Binary Search Trees. If you just copy/paste/modify code from the book or from online, you might get it to work, but you will not benefit from the exercise. You will likely not be able to answer questions about this on the Final.

BST is a Binary Search Tree that stores Albums (see the Albums API). It is somewhat similar to the Binary Search Tree of our book, but there are differences. You will not be able to just copy and paste the book's implementation. Please read the method descriptions carefully. Post questions on piazza as soon as you think of them. Remember that you can ask concept questions but you may not share or examine code that is not your own.

Be sure that you can do this assignment WITHOUT looking at the book. You can start by studying the book, and maybe referring to it the first time you're writing the code, but you should get to a point where you don't need to look at the book to write this code.

You are also required to write a JUnit test class for your implementation. Please see the sample files on piazza. Look in the resources tab for src.zip. This zip file will expand to MySet.java, ArrayMySet.java, ArrayMySetTest.java and a README. Study this for an example of how to write a JUnit test class.

Where to submit

http://web-cat.cs.vt.edu

What to submit

All your source code files, including your JUnit file. These must be zipped or jarred. You will submit the .zip or the .jar file.

Practicing submitting and using JUnit

You can use the MySet files to practice submitting to web-cat. Use the MySet Practice assignment. The files can be downloaded from piazza. Look for src.zip in the Resources/resources section.

Version:

102815 initial version

Author:

acsiochi (based upon cshaffer's Binary Search Tree interface)

Constructor Summary

Constructors

Constructor and Description
BST() A newly instantiated BST is empty (it contains no Tnodes).

Method Summary

Methods

Modifier and Type	Method and Description
boolean	**find**(**Album** album) Determine if the specified album is in this BST.
java.lang.String	**inOrder**() The inorder traversal consists of the keys of each node delimited by the pipe (|) symbol.
void	**insert**(**Album** value) Insert the specified value into this BST.
boolean	**isEmpty**() Determine if this BST is empty.
void	**remove**(**Album** a) Remove the specified album from this tree.
Album	**retrieve**(java.lang.String artist, java.lang.String title) Retrieve the album with the given artist and album title.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BST

public BST()

A newly instantiated BST is empty (it contains no Tnodes).

Method Detail

isEmpty

```
public boolean isEmpty()
```

Determine if this BST is empty.

Returns:

true if this BST is empty, false otherwise.

insert

```
public void insert(Album value)
```

Insert the specified value into this BST.

Parameters:

`value` - is the Album to be inserted

inOrder

```
public java.lang.String inOrder()
```

The inorder traversal consists of the keys of each node delimited by the pipe (|) symbol. For example if this BST contained two Tnodes with the following Albums:

```
< "n1", "t1", songs1 >
< "n2", "t2", songs2 >
```

then this inOrder() method would return

```
|n1t1||n2t2|
```

An empty BST would return the empty string "".

Returns:

String representation of the inorder traversal of this BST

find

```
public boolean find(Album album)
```

Determine if the specified album is in this BST.

Parameters:

`album` - is the Album to search for in this BST

Returns:

tue if album is in this BST, false otherwise

retrieve

```
public Album retrieve(java.lang.String artist,
             java.lang.String title)
```

Retrieve the album with the given artist and album title.

Parameters:

`artist` - is the name of the artist of this album

`title` - is the title of this album

Returns:

the Album matching the given artist and title.

remove

```
public void remove(Album a)
```

Remove the specified album from this tree. If the Album was not in the tree, do nothing.

Parameters:

`a` - is the Album to remove

Assignment 2 BST **Nov. 2, 2015**

Class Album

java.lang.Object
    Album

```
public class Album
extends java.lang.Object
```

Album is a class that stores information about a music album like a CD or an LP. It is a collection of songs by an artist. It has a title. Its key is a concatenation of the artist and the title.

You will use this class in your Binary Search Tree implementation. The source code is not provided, only the .jar file. To use this file you will need to add the .jar file to your class path. In Eclipse this is done with the Build Path popup menu (right click the project and select Build Path...)

Version:

102815

Author:

acsiochi

Constructor Summary

Constructors

Constructor and Description
Album(**Album** a) make a clone of the specified album.
Album(java.lang.String artist, java.lang.String title, java.lang.String[] songs) Set the attributes of this Album to those specified in the parameters

Method Summary

Methods

Modifier and Type	Method and Description
java.lang.String	**artist**()
int	**compareTo**(**Album** a) Albums are compared by their keys (see the key() method of this class).

boolean	**equals**(java.lang.Object o) Albums are equal if they have the same artists, titles, and songs.
java.lang.String	**key**() The key of an album is a unique identifier for this Album.
void	**setArtist**(java.lang.String artist) Set the artist of this album to the specified artist.
java.util.ArrayList<java.lang.String>	**songs**()
java.lang.String	**title**()

Methods inherited from class java.lang.Object

getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Album

```
public Album(java.lang.String artist,
     java.lang.String title,
     java.lang.String[] songs)
```

Set the attributes of this Album to those specified in the parameters

Parameters:

`artist` - is the string to store in this Album's artist field

`title` - is the string to store in this Album's title field

`songs` - is the ArrayList to store in this Album's songs field

Album

```
public Album(Album a)
```

make a clone of the specified album. the clone will contain new Objects rather than references to the Objects in the specified album.

Parameters:

`a` - is the Album to clone

Method Detail

artist

```
public java.lang.String artist()
```

Returns:

the artist of this Album

setArtist

```
public void setArtist(java.lang.String artist)
```

Set the artist of this album to the specified artist.

Parameters:

`artist` - is the name of the artist to set in this Album

title

```
public java.lang.String title()
```

Returns:

the title of this Album

songs

```
public java.util.ArrayList<java.lang.String> songs()
```

Returns:

the ArrayList of songs of this Album

key

```
public java.lang.String key()
```

The key of an album is a unique identifier for this Album. It is the concatenation of the artist and the title. For example, if the artist is "joe" and the title is "cool" the key is "joecool"

Returns:

the key of this Album

compareTo

```
public int compareTo(Album a)
```

Albums are compared by their keys (see the key() method of this class). For example:

```
if we have
Album a;
Album b;
Album c;

and stuff is done to set up a and b so that a.key() == "joecool"
and b.key() == "billykid", and we set c = new Album(a);
so that c is a clone of a, then

a.compareTo(b) > 0 is true
b.compareTo(a) < 0 is true
c.compareTo(a) == 0 is true
```

Parameters:

`a` - is the other Album to compare this Album to

Returns:

less than 0, equal to 0, or greater than 0 according as this Album's key is lexicographically less than, equal to, or greater than a's key.

equals

`public boolean equals(java.lang.Object o)`

Albums are equal if they have the same artists, titles, and songs.

Overrides:

`equals` in class `java.lang.Object`

Parameters:

`o` - is the Object to compare to this Album

Returns:

true if this Album is equal to Album o, false otherwise.