skooter500 / **OOP-2018-Lab-Test-2**

No description, website, or topics provided.

14 commits | **1** branch | **0** releases | **1** contributor | MIT

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

skooter500 bla — Latest commit `67b0835` 16 days ago

ie/dit	bla	17 days ago
.gitignore	Initial commit	17 days ago
LICENSE	Initial commit	17 days ago
README.md	bla	16 days ago
hnj0.abc	bla	17 days ago

README.md

Object Oriented Programming Lab Test 2 2018

For todays' lab test, you will be writing a parser for ABC music files in Java. ABC music files are plain text ascii files that store sheet music notation. ABC music files can contain the notation for multiple tunes in one file and tunes are seperated by one or more blank lines. The file you will be parsing today has 100 tunes in it. Each tune consists of headers which are the letters X, T, R etc followed by a :. The actual music notes follow the headers. The X header is called the index number of the tune and is always an integer. The T header is the tune title. Tunes always begins with an X header and there is always at least one title, though there can be optionally multiple T headers as tunes can have multiple titles. There are other headers, but we are only interested in the tites and the index numbers for the purpose of this test.

Here is an extract from the file you will be parsing today that shows the ABC notation for two tunes:

```
X:3
T:Banish Misfortune
R:jig
H:First bar also played |^fed cAG|
D:Tommy Keane & Jacqueline McCarthy: The Wind among the Reeds
D:Chieftains Live.
Z:id:hn-jig-3
M:6/8
K:Dmix
fed cAG | AGd cAG | ~F3 DED | ~F3 GFG |
~A3 cAG | AGA cde | fed cAG | Ad^c d2e :|
|: f2d d^cd | f2a agf | e2c cBc | ece gfe |
f2g agf | e2f gfe | fed cAG | Ad^c d2e :|
|: f2g e2f | ded cdc | ~A3 GAG | ~F3 ded |
c3 cAG | AGA cde | fed cAG | Ad^c d2e :|
P:variations
|: =fed cAG | A2d cAG | F2D DED | FEF ~G3 |
AGA cAG | ~A3 cde | fed cAG | Ad^c d2e :|
|: f2d d^cd | f2g agf | e2c cBc | e2f gfe |
f2g agf | e2f gfe | fed cAG | Ad^c d2e :|
|: f2g e2f | d2e c2d | ABA GAG | F2d ded |
c3 cAG | AGA cde | fed cAG | Ad^c d2e :|

X:4
T:Piper's Chair, The
T:Cathaoir an Ph\'iob\'aire
R:jig
D:Bobby Gardiner: His Master's Choice.
Z:id:hn-jig-4
M:6/8
K:G
DGG GFD|c2c cAc|ded cAG|FAG FEF|DGG GFD|c2c cAc|ded cAF|1 AGF G3:|2 AGF GBd||
|:~g3 agf|d2g gfg|GFG =fef|A2B cBA|GBd g2f|d2e fdc|Bdd cAF|1 AGF GBd:|2 AGF G3||
```

If we take the second tune, "the Pipers Chair" as an example, we can see that tune has an index number of 4 (X:4) and has two titles "Piper's Chair, The" (T:Piper's Chair, The) and "Cathaoir an Ph'iob'aire" (T:Cathaoir an Ph'iob'aire).

Instructions:

- Create a new Java git repository on github and call it JavaTest. Make sure you specify Java as the language on github so you get a Java .gitignore file
- Clone the repository to your computer
- Create the folder structure for a java package in the project you cloned, called `ie.dit` and download this abc file to the root of the folder structure you created.
- Create a class called `Tune` in the package ie.dit with private fields for `x`, `title`, `altTitle` and `notation`. `x` should be of type `int`, `title`, `altTitle` and `notation` should be `String`s. Create public accessor methods for these private fields.
- Write a `toString` method on the `Tune` class. This should return the fields formatted as "x, title, altTitle". If the tune does not have an `altTitle`, then you should leave this part out. For example, the above tunes would be printed as:

```
3, Banish Misfortune
4, Piper's Chair, The, Cathaoir an Ph\'iob\'aire
```

- Write a class called `TuneBook` that has a field called `tunes` of type `ArrayList` of `Tune` objects. To use an `ArrayList`, you should type `import java.util.ArrayList;` at the top of your Java file.
- Write a constructor for the `TuneBook` class that takes a single `String` as a parameter representing the name of the abc file to load. Write the code in this constructor that loads the file line by line and populates the tunes `ArrayList` from the contents of the ABC file. There should be one `Tune` object in the `ArrayList` for each tune in the file and you should set the `x`, `title`, `altTitle` and `notation` fields on each `Tune` object from the file. You can use the accessor methods for this. `title` should be set from the first T header in each tune and the `altTitle` field should be set from the second T header if one exists. If a tune has only one T header, then the `altTitle` field of the tune should be null. If there are more than two titles for a tune (more than two T headers), then you can ignore the third and subsequent titles. `notation` should contain the full tune including the headers and the music notes.

  Here is some example code that loads and prints a text file you can use to get started writing this method. Also fee free to have a look at the code from Monday's class

```
BufferedReader inputStream = null;
try {
    inputStream = new BufferedReader(new FileReader("words.txt"));

    String l;
    while ((l = inputStream.readLine()) != null)
    {
        System.out.println(l);
    }
}
catch (IOException e)
{
    e.printStackTrace();
}
finally
{
    if (inputStream != null) {
        try
        {
            inputStream.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

- Write a `toString` method on the `TuneBook` class that returns a `String` version of the tunes `ArrayList`, with each element of the `ArrayList` on a seperate line in the returned `String`.
- Write a method `public Tune findTune(String title)` on the `TuneBook` class that returns the first matching `Tune` from the ArrayList that contains the parameter `title` in the title of the tune.

- Create an `interface` called `Player` that has one method called `void play()`
- Implement the interface on the `Tune` class. The play method should just print the notation for the tune to the console.
- Put a `main` method on the `TuneBook` class that has the following code on it to test your solution. It's best to test your code as you go along and don't wait until you have all the parts completed before compiling and running your code.

```
public static void main(String[] args)
{
    TuneBook tb = new TuneBook("hnj0.abc");
    System.out.println(tb);

    Tune t = tb.findTune("Scotsman over the Border");
    t.play();
}
```

Below is sample of what your program should output:

```
1, Bride's Favourite, The
2, Irish Washerwoman, The
3, Banish Misfortune
4, Piper's Chair, The, Cathaoir an Ph\'iob\'aire
5, Scotsman over the Border, The
...
(lines omitted)
...
97, O'Broin's Flightcase
98, Wheels of the World, The
99, Peter O'Byrne's Fancy, Peter Byrne's Fancy
100, Humours of Drinagh, The

X:5
T:Scotsman over the Border, The
R:jig
H:Related to The Carraroe Jig, #181
D:Music at Matt Molloy's.
D:Noel Hill & Tony Linnane.
D:Molloy, Peoples, Brady.
Z:id:hn-jig-5
M:6/8
K:D
DED FDF | AFA d2A | ~B3 BAB | dgf edB |
ADD FDF | AFA d2A | ~B3 AFA | dAF ~E3 :|
|: dfa afa | bag fef | dfa afe | def edB |
dfa afa | bag fed | B2B AFA | dAF ~E3 :|
P:variations
|: ~D3 FDF | AFA d2A | ~B3 BAB | def edB |
ADD ~F3 | AFA d2A | BdB AFA |1 dAF EFE :|2 dAF EFA ||
|: dfa afa | bag fge | dfa afe | def edB |
dfa af/g/a | bag fge | d2B AFA |1 dAF EFA :|2 dAF EFE ||
```

Commit your code whenever you get something completed. Submit the URL to your git repository here.

Marking Scheme

Description	Marks
Writing the `Tune` class, with private fields & public accessors	15 marks
`toString` method on the `Tune` class	10 marks
`TuneBook` class with an `ArrayList` of Tune objects	10 marks
Loading tunes into the `ArrayList`	25 marks
`toString` method on the TuneBook	10 marks
Writing the `findTune` method	10 marks
Writing and implementing the interface	10 marks
Correct use of git	10 marks

Description Marks