

CSC 110: Fundamentals of Programming I

Assignment #7: Objects, searching, sorting

Due date

Wednesday, April 6th, 2016 at 11:55 pm via submission to connex.

Specification Document

How to hand in your work

Submit the file **BowieSongCreator.java** in the Assignment #7 link on connex.

Learning outcomes

When you have completed this assignment, you will understand:

- How to create an *instance* of a *class* (object instantiation).
- How to invoke an *object's instance methods*.
- How to create and use an *array of objects*.
- How to *search* and *sort* through data in an array.

The late David Bowie used a cut-up technique when writing songs. In this technique, text is cut-up and rearranged in different orders to create the lyrics for a song.

You will write methods for **BowieSongCreator.java**. Your program will use **SongLine** objects. The **SongLine.java** class has been written for you. Download and save the **SongLine.java** file into the same directory as the **BowieSongCreator.java** file.

SongLine objects have three instance variables, **String keyword**, **int lineNumber**, and **String words**, as shown below. As an example, suppose a line from the “Happy Birthday” song was represented as an instance of a **SongLine** called **birthdayLyrics**:

SongLine	birthdayLyrics: SongLine
String keyword int lineNumber String words	keyword: Celebration lineNumber: 1 words: Happy Birthday to you..

In this assignment, you will use a Scanner to read through a text file containing song line data, and fill an array of **SongLine** objects with this data. You will then create methods to print, search and re-order the lines of song lyrics contained in the array.

Your program must:

1. Follow the [specification document](#).
2. Ask the user to enter the name of the input file. Note that the marker can use a different data file than the examples provided in the Appendix. The data file will have exactly the same format (i.e., same columns, with each row having the same number of items) but it may contain more or fewer rows. ***Continue to ask for another input filename if the one entered is not valid.***

Input file format (example: file1.txt)

- The first line contains the number of SongLine data elements are contained in the file (the total number of items to put into the array)
 - Each remaining line contains a keyword, followed by a line number. The remaining words on the line make up lyrics of the line of a song.
3. After linking a Scanner to an input file, write the method **makeArray**, that accepts the Scanner as a parameter, and then creates and fills a SongLine array with all of the data contained in the file the Scanner is linked to. The method then returns the array.
 4. Using the array returned from the **makeArray** method, create and test all of the other methods according to the specification document. Examples of how to test these methods are shown in the Appendix.

Marking

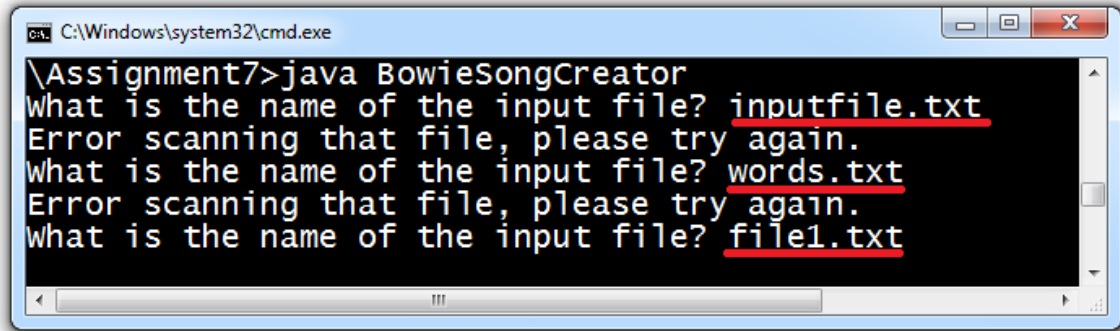
Your mark will be based on the following criteria:

- Your code *must compile and run*. Some examples of how to test your methods, along with expected output, are outlined in **Appendix A**.
- Your code must conform to all the requirements mentioned in the [specification document](#).
- Test each of the required methods to ensure each one functions correctly.
- Your code must follow the guidelines outlined in Style_Guidelines.pdf, found through the Lectures & Stuff link in the Lab Resources folder on connex. You may notice that the specification document provides some very nice comments you are welcome to borrow.

Appendix A – Testing your code

Getting started setting up the *main* method:

Your main method will have two Scanners. The first Scanner is used to read in user input from the console (user entries underlined in red):



```
C:\Windows\system32\cmd.exe
\Assignment7>java BowieSongCreator
what is the name of the input file? inputfile.txt
Error scanning that file, please try again.
what is the name of the input file? words.txt
Error scanning that file, please try again.
what is the name of the input file? file1.txt
```

The second Scanner scans the contents of a file. If the file cannot be found, the program prompts the user to enter another file name, until the second Scanner is successfully linked to a file (as shown above – as smallList.txt was found).

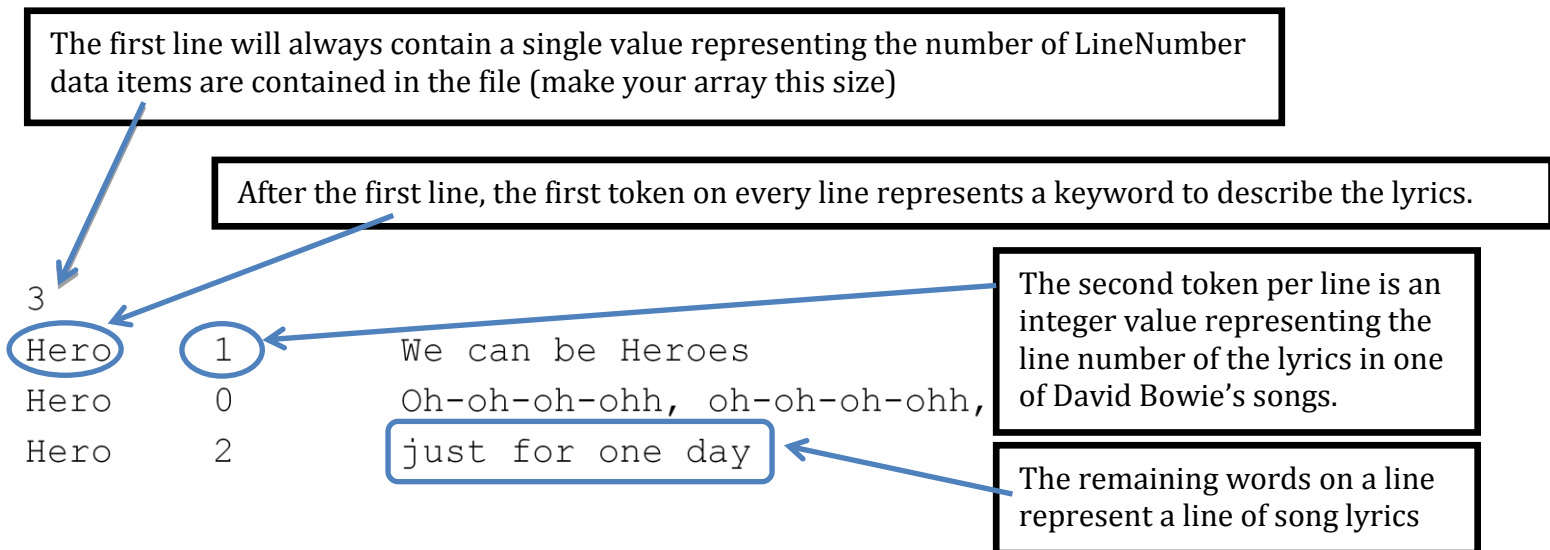
Testing the *makeArray* method:

Now that you have a Scanner linked to a text file, the next step is to copy all of the contents of the file into an array of SongLines.

Input files will all have the following format:

- The first line contains the number of SongLine data elements that are contained in the file (the total number of SongLines to put into the array)
- Each remaining line contains a keyword, followed by a line number. The remaining words on the line make up lyrics of the line of a song.

Assuming we have linked a Scanner to the file **smallExample.txt**:



Each line of data can be used to create a single instance of a SongLine. For example, the first line would create the following SongLine object (in this case named *test*):

test: SongLine
keyword: "Hero" lineNumber: 1 words: "We can be heroes"

Then each SongLine is inserted into an array, creating the following array of SongLine objects:

keyword: "Hero" lineNumber: 1 words: "We can be Heroes"	keyword: "Hero" lineNumber: 0 words: "Oh-oh-oh-ohh, oh-oh-oh-ohh,"	Keyword: "Hero" lineNumber: 2 words: "just for one day"
0	1	2

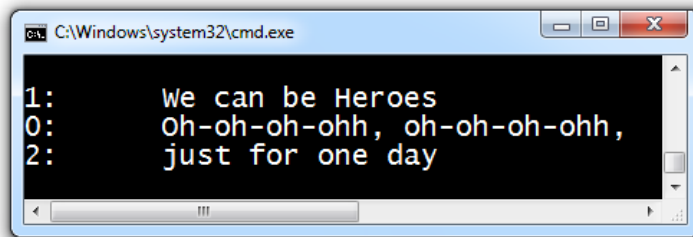
Testing the `printArray` method:

Given an array of `SongLine` objects, the method prints out information about each element in the array.

Given the following array (read in from **smallExample.txt**):

Hero	Hero	Hero
1	0	2
We can be Heroes	Oh-oh-oh-ohh, oh-oh-oh-ohh	just for one day
0	1	2

Calling this method and passing in the array above produces the following output:

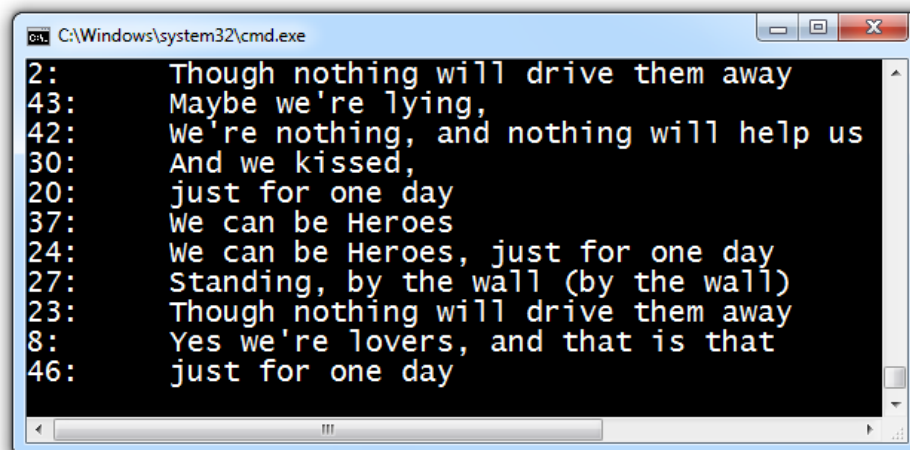


```
C:\Windows\system32\cmd.exe
1: We can be Heroes
0: Oh-oh-oh-ohh, oh-oh-oh-ohh,
2: just for one day
```

Given the following array (read in from **file1.txt**):

Hero	Hero	Hero		Hero	Hero
1	15	47		8	46
And you, you will be queen	Like the dolphins, like dolphins can	Oh-oh-oh-ohh, oh-oh-oh-ohh,	...	Yes we're lovers, and that is that	just for one day
0	1	2	...	47	48

Calling this method and passing in the array above produces the following output:



```
C:\Windows\system32\cmd.exe
2: Though nothing will drive them away
43: Maybe we're lying,
42: We're nothing, and nothing will help us
30: And we kissed,
20: just for one day
37: We can be Heroes
24: We can be Heroes, just for one day
27: Standing, by the wall (by the wall)
23: Though nothing will drive them away
8: Yes we're lovers, and that is that
46: just for one day
```

Testing the `listSongsByKeyword` method:

Some input file may contain SongLine data with different keywords. Given an array of SongLines and a keyword, this method prints out all SongLines in the array with the given keyword

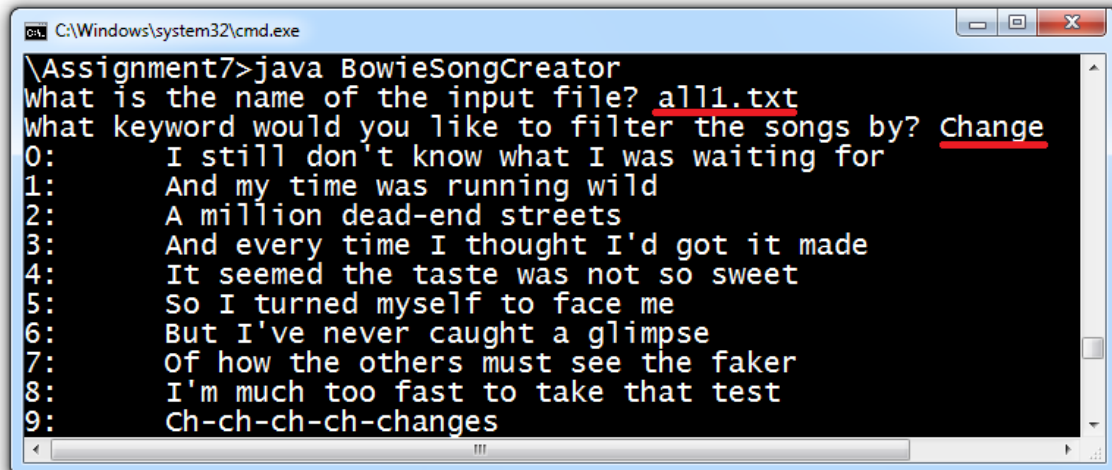
Example:

Given the following array (read in from **list2.txt**):

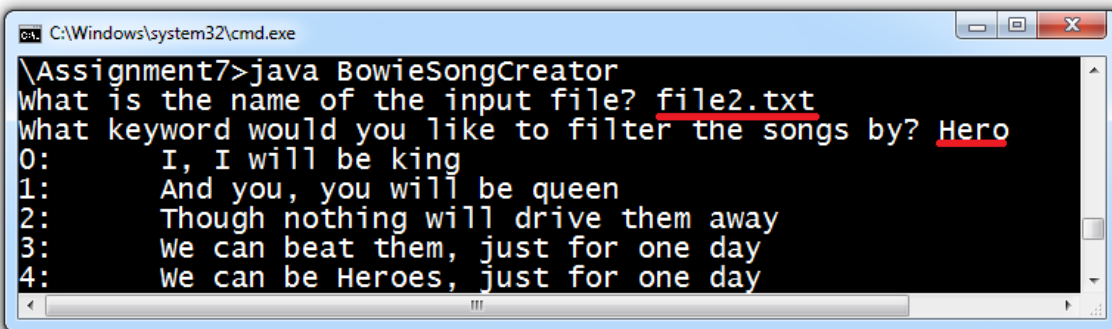
Space 31 she knows	Hero 28 And the guns shot above our heads	Space 15 If you dare	Change 18 But I can't trace time	...	Space 16 This is Major Tom to ground control	Change 28 Ch-ch-ch-ch- changes
0	1	2	3	...	145	146

```
sortByLineNumber(myArray);  
System.out.print("What keyword would you like to filter the songs by? ");  
String filterWord = userReader.next();  
listSongsByKeyword(myArray, filterWord);
```

Given the above array of SongLines, and 4 lines of code, the following output is produced (input entered by the user into the console is underlined in red):



```
C:\Windows\system32\cmd.exe  
Assignment7>java BowieSongCreator  
What is the name of the input file? all1.txt  
What keyword would you like to filter the songs by? Change  
0: I still don't know what I was waiting for  
1: And my time was running wild  
2: A million dead-end streets  
3: And every time I thought I'd got it made  
4: It seemed the taste was not so sweet  
5: So I turned myself to face me  
6: But I've never caught a glimpse  
7: Of how the others must see the faker  
8: I'm much too fast to take that test  
9: Ch-ch-ch-ch-changes
```



```
C:\Windows\system32\cmd.exe  
Assignment7>java BowieSongCreator  
What is the name of the input file? file2.txt  
What keyword would you like to filter the songs by? Hero  
0: I, I will be king  
1: And you, you will be queen  
2: Though nothing will drive them away  
3: We can beat them, just for one day  
4: We can be Heroes, just for one day
```

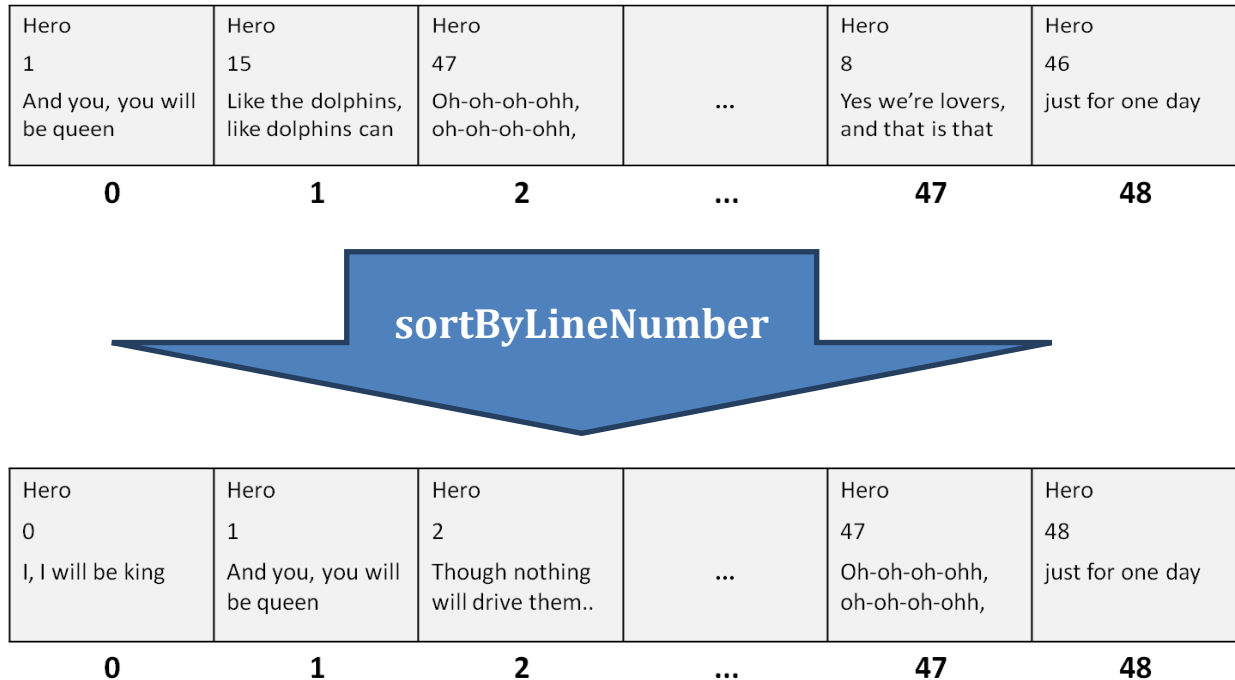
Note: The example outputs only show the first few lines of output.

Testing the `sortByLineNumber` method:

Given an array of `SongLines`, this method sorts the array by the **lineNumber** instance variable. In the example below, the **lineNumber** instance variable is shown as the second line in each element of the array (ranging from 0 to 48).

Example:

Given the following array (read in from **list1.txt**):



Call your completed **printArray** method before and after sorting to ensure the array has been properly sorted. By sorting by line number and then printing you should find the lyrics to a David Bowie song!

Testing the *makeSong* method:

Given an array of *SongLines*, this method should search through and select only specific elements in the array and print out their words to create the lyrics to a custom song that is 21-lines long. You must create your own cut-up technique to choose lyrics. Document (through code comments) the method you used to choose which items in the array were printed to create your own custom song.

Given the following array (read in from **list3.txt**):

Dance 23 Back in the USSR	Dance 40 Dancing in the Street	Space 17 Hey, that's far out so you heard him..	Change 31 Don't tell them to grow up and out of..	...	Space 0 Didn't know what time it was and the..	Space 14 Now it's time to leave the capsule
0	1	2	3	...	308	309

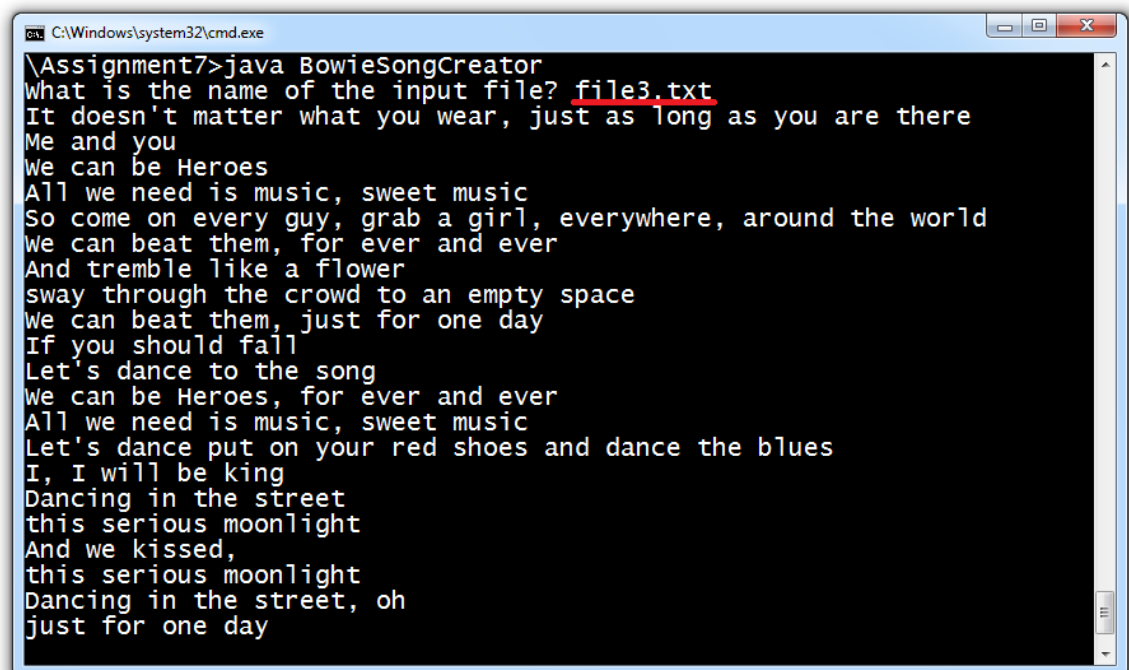
```
SongLine[] myArray = makeArray(fileReader);  
makeSong(myArray);
```

The 21-line song below uses the following algorithm:

- 1) Using the *Random* class, a random number is generated to select an index number between 0 and the last index in the array (309 in the example array above)
 - 2) Print out the *words* of the *SongLine* object at the random index if it has *keyword* "Dance". If the *keyword* is not "Dance", run step 1) again to go to a different index.
 - 3) Repeat Step 2) to print out a second line with *keyword* "Dance"
 - 4) Repeat Step 3) but instead search for *keyword* "Hero".
 - 5) Repeat the steps 1) to 4) seven times. (As they produce 3 song lines each time).
- The overall pattern is the *words* from a *SongLine*'s with *keyword* "Dance", followed by another line with *keyword* "Dance" followed by one with "Hero", repeated 7 times.

Note: the *lineNumber* instance variables are not printed, only "*words*".

The 21-line song produced using the algorithm explained above. Create your own algorithm to produce song lyrics. Be creative and **have fun!**



```
C:\Windows\system32\cmd.exe  
Assignment7>java BowieSongCreator  
What is the name of the input file? file3.txt  
It doesn't matter what you wear, just as long as you are there  
Me and you  
We can be Heroes  
All we need is music, sweet music  
So come on every guy, grab a girl, everywhere, around the world  
We can beat them, for ever and ever  
And tremble like a flower  
sway through the crowd to an empty space  
We can beat them, just for one day  
If you should fall  
Let's dance to the song  
We can be Heroes, for ever and ever  
All we need is music, sweet music  
Let's dance put on your red shoes and dance the blues  
I, I will be king  
Dancing in the street  
this serious moonlight  
And we kissed,  
this serious moonlight  
Dancing in the street, oh  
just for one day
```