CSC110 Assignment 7:
Two Dimensional Arrays and File I/O

Objectives

Upon completion of this assignment, you need to be able to:

- Become familiar with the source code required for a standard Java *class* description.
- Create an *instance* of a class, also known as *object instantiation*.
- Invoke an object's instance methods.
- Create and use an array of objects.
- Search through and sort the objects of an array.

Introduction

The late David Bowie (and others) used the cut-up technique for writing some of his song lyrics. We will simulate part of this technique by rearranging lines from songs to create a new song.

For this assignment, you are given a completed class, called `SongLine`, and will create a class called `CutupSongCreator` that can create new songs by using the cut-up technique.

Quick Start

(1) Download this pdf file and store it on your computer.
(2) Create a directory / folder on your computer specific to this assignment. for example, `CSC110/assn7` is a good name.
(3) Save the file called `SongLine.java` in your directory. Look at this file, but do not change any of the methods.
(4) You can save the following practice *text* files in the same directory. Use these for testing your program.
    - `file1.txt`

- `file2.txt`
- `file3.txt`

(5) Create a new file called `CutupSongCreator.java` in the same directory.
(6) Using the information in the `CutupSongCreator` specification document, create the program shell.
   - We recommend that you cut and past the method headers specified and write comments above each that are similar to the specification details.
   - This class that you create will make use of a completed class: the source code contained in `SongLine.java` and the documentation contained in its own specifications.
(7) Start filling in the code for each method, starting with the easiest ones first, remembering to test your code early and often.

Detailed Instructions

In the following subsections, we provide a detailed description of the given `SongLine` and how to approach the implementation of each of the methods named in the `CutupSongCreator` class:

the `SongLine` class

Most of the methods will be handling `SongLine` objects. The source code and specifications are provided for you. Note that, similarly to standard Java documentation, you do not need to know anything about the actual source code. However, this class is implemented using standard Java practices and is therefore a good example for beginner Object-oriented programmers.

Basically a `SongLine` object has three *fields*[*]:

genre: This is a single word that categorizes a song type. For instance, the word *celebration* may be a genre that would be used for lines of songs that celebrate something.
lineNumber: The line number (from 1 to the number of actual lines in a song) of a particular song line.
words: This actual words in the complete line. For instance, in the song, called *Happy Birthday*, the words of line number $1, 2$ and 4 are all "Happy birthday, to you.".

[*]The definition of the field can be found on page 536 of the textbook.

Using the `Scanner`

This assignment requires you to initialize and use two `Scanner` objects. Both of these need to be instantiated in the `main` method. The first one is used to access text from the console. The second one is used to access text from a particular text file. Both `Scanner`s parse the text, and know where to look for it, because they have access to the *file handle*[†]

The first `Scanner`

The user provides the name of the text file to look through for `SongLine` items. The first `Scanner` links up to `System.in` to get that filename.

The second `Scanner`

The second `Scanner` object is initialized (in `main`) using the filename provided by the user. Then, it is passed to the `makeArray` method, where it helps extract all the information from the lines of text to create an array of `SongLine` objects. Remember, after the call to `makeArray`, to close the second `Scanner` object (in this example it is called `reader`), which in turn closes the open text file.

```
reader.close();
```

Note that the first `Scanner` should not be `closed`, because the `Scanner` will try to close `System.in`.[‡]

The input file

Three input files are provided for testing the `CutupSongCreator` methods; each file has the same format. We recommend you do not alter these files; if you do, note that the first line must accurately state how many song lines follow. Each song line contains:

(1) a single-word genre, followed by
(2) a line number, followed by
(3) the remaining words, which provide the single line of a song's lyrics.

[†]A file handle is actually a number, assigned by the operating system, to an open file. In Java, `System.in` is also treated as a file, as is `System.out`

[‡]It is the writer's opinion that this is a bug in Java.

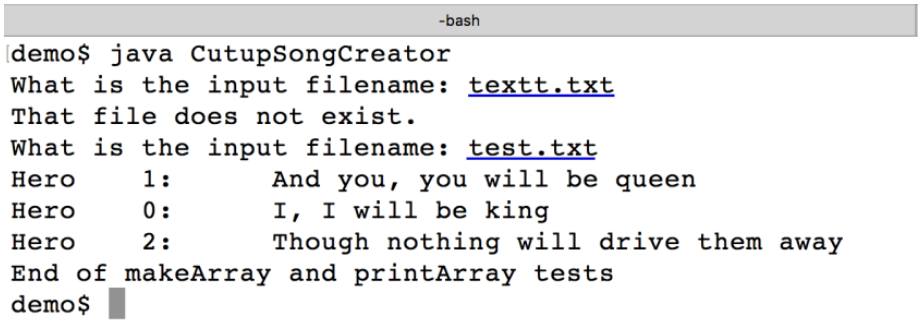

FIGURE 1. Error checking, makeArray, and printArray example

Testing the code in `main`: Some examples

In this section, we provide some examples of user interaction with the program. Note that in all the examples, the user input is underlined in blue. The example in Figure 1 assumes that a file named `test.txt` resides in the current directory and contains the following text:

```
3
Hero  1  And you, you will be queen
Hero  0  I, I will be king
Hero  2  Though nothing will drive them away
```

The examples in Figure 2 and 3 both act on a file called `test2.txt` that contains the following lines:

```
5
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
Change   4   It seemed the taste was not so sweet
```

Creating a song

This example shows a random song of 21 lines, taken from `file3.txt`. Note that when this method is called, a new song is created.

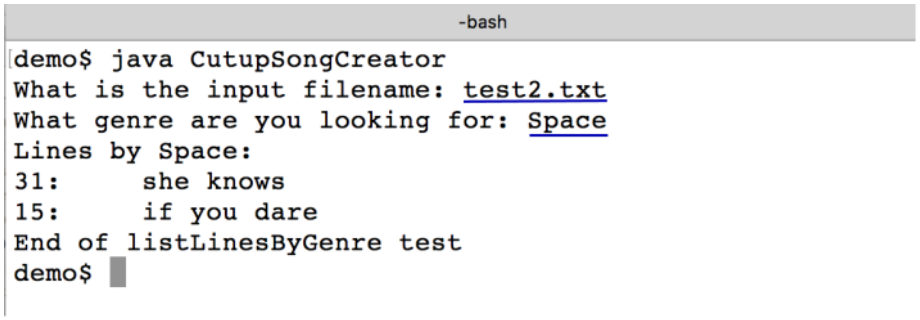

FIGURE 2. listLinesByGenre example

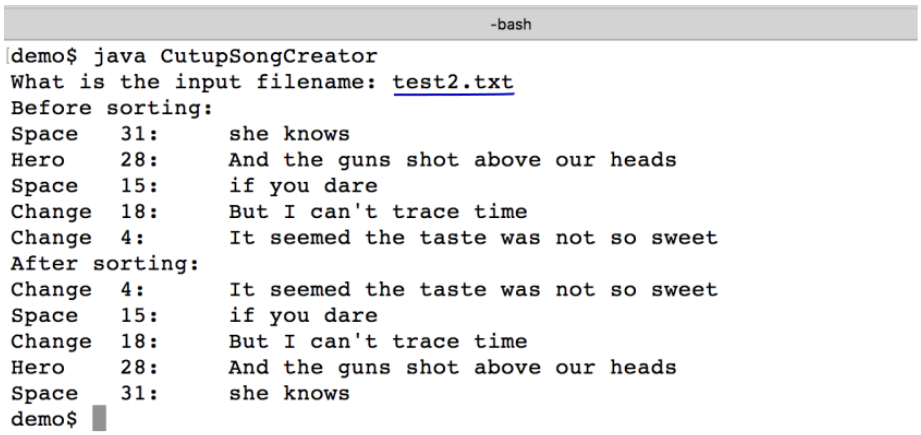

FIGURE 3. sortByLineNumber example

Submission

Submit the following completed file to the Assignment folder on conneX.

- `CutupSongCreator.java`

Please make sure you have submitted the required file(s) and conneX has sent you a confirmation email. Do not send [`.class`] (the byte code) files. Also, make sure you *submit* your assignment, not just save a draft. Draft copies are not made available to the instructors, so they are not collected with the other submissions. We *can* find your draft submission, but only if we know that it's there.

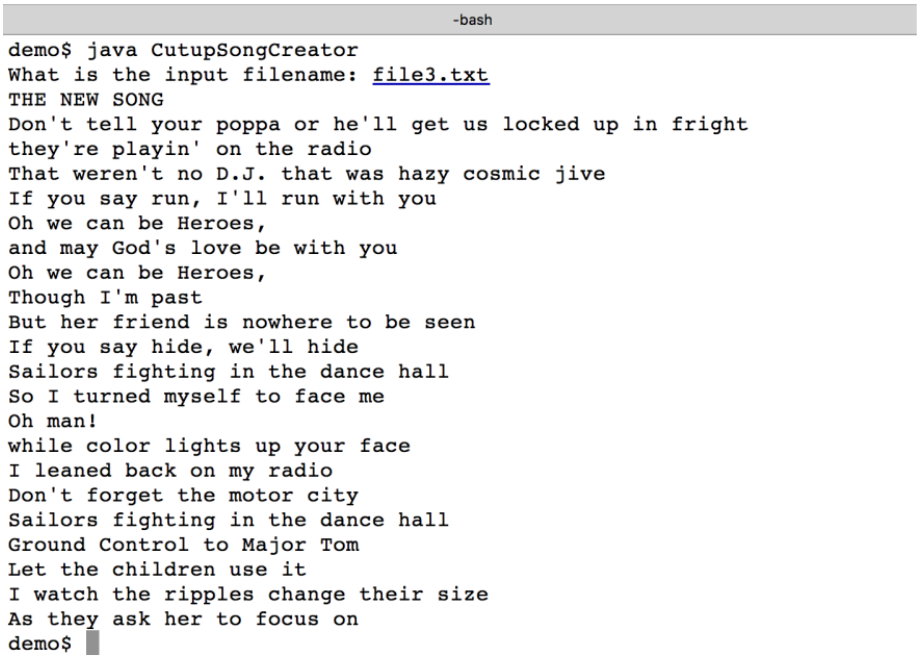

FIGURE 4. makeSong example

A note about academic integrity

It is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

Grading

Marks are allocated for . . .

- No errors during compilation of the source code.
- The method headers must be exactly as specified and the methods must perform as specified. Be sure to read the **Method Details** in the specification document.
- The user is allowed to make a mistake with the filename, and can make a correction, as shown in Figure 1.
- The `main` method creates only two `Scanner` objects and tests each of the methods in `CutupSongCreator`.
- Style of the source code meets the requirements outlined in the Coding conventions document available in the Lab Resources folder of conneX.