Homework #3

This work is based on material from Chapter 4 of *Objects First with Java - A Practical Introduction using BlueJ* by David Barnes and Michael Kolling. The goal is to allow you to practice writing class definitions that make use of collections and iteration.

You will need to be familiar with collection classes and iteration, both of which are covered in chapter 4 of the textbook. In particular, you will need to make use of the `ArrayList` class that is available in the `java.util` package, and understand the use of *for-each loops*, *while loops* and *Iterators*. You will find it helpful to study the source code of the `music-organizer` project, which we have discussed in the course and is explored in detail in chapter 4 of the textbook.

Introduction

Download the **HW3_code.zip** file, and the file on your computer. Open it in BlueJ, and compile the code.

For this Homework imagine that you have been asked to write part of the software that will be used to manage the voting for an International Singing Contest. You have been provided with the `Singer` class and you need to write the `Contest` class:

- The Singer class represents the details of a singer. It has 2 fields of type `String` called `name` and `nationality`, and 1 field of type `int` called `votes`. You can use the 'test fixture to object bench' menu option of the `ContestTest` class to put a sample of `Singer` objects onto the object bench to help you when testing your code.

- You need to write the `Contest` class. It must be able to store an `ArrayList` of `Singer` objects and provide a range of functionality through different methods.

First, we are going to have an overview of the already existing classes in the project.

ContestTest and SingerTest classes

These classes are not part of your assessment and you do not need to modify their code. You can use the 'test fixture to object bench' menu option of the `ContestTest` class to put a sample of `Singer` objects onto the object bench in order to help with testing as you develop your program.

The Singer class

This is a simple class that doesn't require any change. It consists of:

- One constructor that takes two parameters of type `String` that are used as the initial values of its `name` and `nationality` fields. The `votes` field is initialized to zero.
- An accessor method for each field.
- A method called `increaseVotes` that takes a single `int` parameter and returns nothing. The method adds the value of the parameter to the value of the `votes` field if the parameter's value is greater than zero.

- A method called `getDetails` that takes no parameters and returns a `String`. It returns a `String` containing the values of the fields in the following format: `"Boa Kwon is Korean and has 100 votes."` where `"Boa Kwon"` is an example of the value of the `name` field, `"Korean"` is an example of the value of the `nationality` field and `100` is an example of the value of the `votes` field. Different `Singer` objects will have different values for these fields.

Task

Write the `Contest` class as described below. Pay careful attention to the exact details of the requirements; such as the names of methods and the parameters required.

Part 1 – The Contest class – Basic methods **(50 Points)**

The `Contest` class must use an `ArrayList` to store any number of `Singer` objects.

Write:

a) A method called `addSinger` that takes a `Singer` object as a parameter and adds it to the `ArrayList`. You may assume that two singers with the same name will never be added to the same list and there is no need to check for this.

b) A method called `getSize` that returns the number of `Singer` objects in the `ArrayList`.

c) A method called `list` that prints out the details of all the `Singer` objects stored in the `ArrayList`, one per line. Use the `Singer` objects' `getDetails` method to format the details. The printed list must **not** include the index of each `Singer` in the list.

Part 2 – The Contest class – Iterative methods **(50 Points)**

Write:

a) A method called `voteFor` that takes a single parameter of type `String` called `name` and does not return anything. This method locates the single `Singer` object in the `ArrayList` whose `name` field *exactly* matches the contents of the method's `name` parameter, and then adds 1 to the number of votes for that singer.

Note that the match required is a character-by-character match of two strings. You must assume that at most one `Singer` will have the matching name, but there might also be the case that there is not a single matching `Singer`. If no matching `Singer` is found then the method must print an error message.

b) A method called `shortlist` that takes a single `int` parameter called `minimumVotes` and does not return anything. This method must remove from the list all the `Singer` objects that have *fewer* votes than the value of the parameter given. There could be zero or more objects removed by this method.

Recommendations

I encourage you to keep checking whether your code still compiles after each change. It will be important to adopt the same approach with this assignment.

The Contest class

Start with the initially empty class. Add your name and date to the opening class comment.

Add a field

Add a field to the `Contest` class for storing the collection of `Singer` objects. You will need to use an import statement at the top of the file. (See the `MusicOrganizer` class in chapter 4 of the book you are not sure about how to do this.)

Constructor

Complete the constructor for the class. The constructor must take no parameters. The constructor must initialize the collection field.

Methods

The order of the methods listed in the requirements section is a reasonable guide to the order in which you should try to implement them. After adding each method, compile the code, fix any errors and test that it works. Use inspectors and the debugger to give yourself confidence that the code is working as you think it should.

Submission

Submit your code, with the revised `addSinger`, `getSize`, `list`, `voteFor`, `shortlist` methods, as follows. Go to CTL, open the menu Student Activity -> Assignment, click on the button for submitting the report. Attach and submit you're a **.zip file** that contains your source file **`Contest.java`**.