Name: _____ Lab Section: 07

CS161 Fall 2015 Programming Exam

Write your name above and return this paper to your lab instructor when you submit your exam. You must finish and submit your work using Check-In by the end of this lab.

Do your own work. Only look at your own computer screen. Do not browse, email, or use cell phones and other digital technology other than the computers in the lab.

During the exam you may use your Java text book (or ebook version if you let your instructor know), the Java API documentation website and the course website, but no other web pages.

Your grade will be based on:

- Correctness of each method
- Clarity of code and comments. Your code should be easy to read, and if there are non-obvious steps, they should be explained with comments.
- Object-oriented design: Does your code capture the idea of encapsulation
- Testing: Does your main method adequately test all aspects of your code?

You will write four java classes for keeping track of music groups. The names of the classes are Groups, MusicGroup, Band and Orchestra. The classes Band and Orchestra must extend the class MusicGroup, to allow an instance of Groups to use a single ArrayList to hold instances of Band and Orchestra.

Your classes must implement the following methods.

Groups: class that is the primary application that you will run.

[**3 points**] **constructor:** Creates an empty ArrayList for holding items of type MusicGroup.

[**3 points**] **addGroup:** Add a new MusicGroup, either a Band or an Orchestra.

[**3 points**] **numberOfGroups:** Returns the number of MusicGroups that have been added.

[**3 points**] **findGroup:** Given a String argument that is the name of a MusicGroup, search the ArrayList for this MusicGroup. Return the MusicGroup if found, otherwise return null.

[**3 points**] **toString:** Returns a string representing all Bands and Orchestras that have been added, by implicitly calling the toString method of each.

[**2 points**] **main:** A main method that performs the test specified below.

[**4 points**] **MusicGroup:** parent class of Band and Orchestra.

[**4 points**] **constructor:** Has three String arguments, a MusicGroup name, the name of a person in the MusicGroup, and the kind of music the MusicGroup plays. Assigns these values to protected instance variables.

[**3 points**] **getName:** Returns name.

[**3 points**] **setName:** Changes the value of name.

[**3 points**] **getPerson:** Returns person's name.

[**3 points**] **setPerson:** Changes the value of person's name.

[**3 points**] **getMusictype:** Returns the music type.

[**3 points**] **setMusictype:** Changes the value of music type.

[**3 points**] **equals:** Returns true if two objects being compared have the same name. This must correctly override the default equals method.

[**3 points**] **toString:** Returns a string like `MusicGroup("Beatles","Paul McCartney","rock")`

[**2 points**] **main:** A main method that tests each method in this class.

[**4 points**] **Orchestra:** child class of MusicGroup

[**4 points**] **constructor:** Has three String arguments, for the name, key person, and music type for the Orchestra. Just call the parent class's constructor to do all the work.

[**4 points**] **equals:** Returns true if two objects being compared have the correct type and the same name. This must correctly override the default equals method.

[**4 points**] **toString:** Returns a string like `Orchestra("Fort Collins Symphony","Wes Kenny","classical")`

[**3 points**] **main:** A main method that tests each method in this class.

[**4 points**] **Band:** child class of MusicGroup

[**4 points**] **constructor:** Has three String arguments, for the name, key person, and music type for the Band. Just call the parent class's constructor to do all the work.

[**4 points**] **equals:** Returns true if two objects being compared have the correct type and the same name. This must correctly override the default equals method.

[**4 points**] **toString:** Returns a string like `Band("Beatles","Paul McCartney","rock")`

[**3 points**] **main:** A main method that tests each method in this class.

Remember, you are required to implement a `main` method in every class that tests the methods of that class. After your tests in the main methods of each class work, change the main method in your Groups class to perform the following steps.

[**4 points**] Compiles.

1. Add these to your Sky in the order given.
   - The Band named "Great Big Sea", with key person "Alan Doyle" and music type "folk".
   - The Band named "Styx, with key person "Dennis DeYoung", and music type "rock".
   - The Orchestra named "New York Philharmonic, with key person "Alan Gilbert" and music type "classical".
   - The Orchestra named "Hipster Orchestra", with key person "Jingle Punks" and music type "modern".
   - The Band named "Spearhead", with key person "Michael Franti" and music type "reggae".

[**3 points**] 2. Print the groups using the toString method in Groups.

[**1 points**] 3. Print the result of calling method numberOfGroups.

[**3 points**] 4. Print the result of using method findGroup to find the group named "Spearhead".

Here are some reminders about how to use ArrayLists.

- Remember to `import java.utils.ArrayList;`
- Constructing an instance: `ArrayList<Type> list = new ArrayList<Type>();`
- Add an object `o` of type Type doing `list.add(o)`
- Get element at index `i` by doing `list.get(i)`
- The `indexOf(x)` method returns the index of the first occurrence of object `x` in the list, determined by the `equals` method for the type of `x`. It returns -1 if it does not occur in the list.

When you are done, combine your files into a **jar file** named `groups.jar`

```
jar cvf groups.jar *.java
```

and check-in your jar file using the Checkin page at CS161 web site. And you must turn in this exam form with your name on it to your instructor before you leave recitation.