Homework 4

Due Saturday, June 18th, at 11:59pm on Classes for 20 points.

Motivation

- Demonstrate your ability to program **using inheritance**, including **interfaces**, in the Java programming language.
- Demonstrate your knowledge of proper `equals()` implementations in the Java programming language.

Tasks

1. Create one **interface**, called `Musician`, that requires classes that subscribe to it to implement a single `void` method called `perform()`. (*5 pts*)
2. Create one **abstract class**, `BandMember`, that implements the `Musician` interface. Your `BandMember` class must have **three** private, immutable, instance attributes:
    - A **string** called `name`, representing the band member's name. Please define a public `getName()` method for it as well. (*2 pts*)
    - A **string** called `instrument`, representing the band member's instrument's name. Please define a public `getInstrument()` method for it. (*2 pts*)
    - A **boolean** value called `isLeader`, representing whether this band member is a band's leader or not. Please define a public `getIsLeader()` method for it. (*2 pts*)
3. **Four concrete classes** that inherit/extends from `BandMember` — `Guitarist`, `Bassist`, `KeyboardPlayer`, and `Drummer`. Each of these classes must:
    - Implement the `perform()` method. You're free to do whatever you want here for each class. (*1 pt / class*)
    - **Override** the `Object` superclass's `equals()` method. (*0.5 pt / class*)
    - Have at least **one** instance attribute particular to each class. This attribute must be used in the class's `perform()` and `equals()` methods. (0.5 pts / class)
    - At least **one** of these classes must have an **overloaded constructor** method. (*1 pt*)

Check out this sample behaviour using my own implementation of the instructions (you do **not** need to define a `Performance` class):

```java
public class Performance {
    public static void main(String[] args) {
        // Creating my objects
        Guitarist yui       = new Guitarist("Yui", "electric guitar", false, false, "Gibson");
        Guitarist azusa     = new Guitarist("Azusa", "electric guitar", false, true, "Fender")
        Bassist mio         = new Bassist("Mio", "bass guitar", true, 4, "Fender");
        KeyboardPlayer mugi = new KeyboardPlayer("Mugi", "keyboard", false, "organ");
        Drummer ritsu       = new Drummer("Ritsu", "drums", false, 1);

        // Calling each of their perform() methods
        mio.perform();
        yui.perform();
        azusa.perform();
        mugi.perform();
        ritsu.perform();

        // Checking a couple of equals()
        System.out.printf("\n%s and %s are%s equal.\n",
                yui.getName(), azusa.getName(),
                yui.equals(azusa) ? "" : " not");

        System.out.printf("%s and %s are%s equal.\n",
                mio.getName(), mio.getName(),
                mio.equals(mio) ? "" : " not");

        System.out.printf("%s and %s are%s equal.\n",
                mugi.getName(), ritsu.getName(),
                mugi.equals(ritsu) ? "" : " not");
    }
}
```

My output:

```
Our band leader Mio plays their 4-stringed Fender bass guitar!
Yui plays their Gibson electric guitar!
Azusa plays their distorted Fender electric guitar!
Mugi plays the organ on their keyboard!
Ritsu plays their drums!

Yui and Azusa are not equal.
Mio and Mio are equal.
Mugi and Ritsu are not equal.
```

Keep in mind that, of course, depending on the instance variables that you pick for each class, your constructors and outputs will look different to mine.

Bonus Points

- Create a second interface of your design and have one or more of your four concrete classes implement it. If you do this successfully, you will recover any points that you may have lost anywhere else in this homework assignment.

Implementation

- Ensure your code is correct by compiling and testing it.
- A portion of your grade will be based upon readability and organization of your code.
  - Follow the naming guidelines of lecture.
  - Break large functions (if you have any) into multiple functions based on logical organizations.

How to Submit

When submitting to Classes, make sure to do so by placing all of your `java` files into one folder, zipping it, and then uploading it. The zipped folder's structure must be as follows:

```
[lastName-firstName-assignment01]
    |
    |
    |----> Musician.java
    |----> BandMember.java
    |----> Guitarist.java
    |----> Bassist.java
    |----> KeyboardPlayer.java
    |----> Drummer.java
    |
    | And, if you chose to do a second interface for the bonus points:
    |
    |----> [aSecondInterface].java
```