ECE 325 LAB Assignment 2: Managing your equipment inventory

Your neoclassical jazzhopmetal band is becoming more popular, which is great! But, this popularity comes at a cost: you get invited to play lots of shows. To make sure that you don't forget any of your equipment after a show, you decide to create an application that helps you keep track of your equipment. In this assignment, you will implement the application and answer some questions about the design.

Part 1: Implement the application

1) The Equipment classes:
The following types of equipment exist: instruments (guitars and keyboards) and furniture (stools and chairs). You want to be able to distinguish between these types (e.g., because you only have to bring instruments to some concert venues that already have the furniture you need). Design these classes so that it is easy to add new types of Equipment later on. To make keeping track of the number of equipment in the inventory easier, make sure to add a toString() method that returns the type of the equipment (e.g. "Guitar" or "Equipment"). Implement the Equipment class and add the other classes that you need.

2) The EquipmentInventory class:
The EquipmentInventory consists of two components:

- An ArrayList<Equipment> to keep track of all your Equipment,
- A HashMap<String, Integer> to keep track of the number of every type of equipment that you have (so that you can easily display these numbers). Check the Java docs to see how to use a HashMap – it is basically a dictionary of key-value pairs. In this case, the key is a String and the value is an Integer. In your application you decide to use the toString() method of a piece of Equipment as the key. So, an example key-value pair could be "Guitar", 3.

Implement the EquipmentInventory class. Make sure to follow the specification in the comments in the provided EquipmentInventory class.

Hints:
- Make sure you cannot accidentally add the same equipment twice.
- Make sure to update the HashMap after adding/removing a piece of equipment.

3) Do a test run of your application:
You own the following equipment (so add these to your EquipmentInventory in your application's main() method):

3 guitars, 2 keyboards, 3 stools, 1 chair

After adding the equipment, print the contents of the inventory and verify that the numbers are correct. Your output should look as follows:

```
[EquipmentInventory: Guitar: 3, Stool: 3, Chair: 1, Keyboard: 2]
```

(the order of the equipment types may differ, as long as the numbers and the String formatting are the same).

4) Show that you can also remove equipment from the inventory:
You end up selling one keyboard and one stool. Update your inventory and show it again.

Part 2: Questions about design choices

Please answer these questions about the implementation/design choices. There is no need to change your implementation based on these questions, a textual explanation is fine.*

1) Why do we choose to implement add(Equipment e) instead of a more specific method like add(Guitar g)?
2) Why do we use a String instead of an Equipment object in the HashMap?
3) Why do we use an Integer instead of an int in the HashMap?
4) Why do we need to define a custom toString method for the Equipment types? What happens if we use the default method?
5) What happens if we only define a custom toString method for the Equipment or Instrument class (but not for any of the subclasses)?

Rubric

Implementation of the application 15 points total.

Questions 1 point each.

(20 points total)

Please submit:
1) A zip file containing your code and a PDF with the answers to the questions above.
Name the file 'FirstName_ID_lab_asg2.zip' and keep the exact same file structure as the zip that was provided for the assignment. So, for example:
Filename: Cor-Paul_1234567_lab_asg2.zip

```
|------- solution.pdf
|------- src
|        |------- ece325
|        |        |------- labs
|        |        |        |------- lab2
|        |        |        |        |------- *.java
```

2) A screencast/movie that shows the following steps:
- Open your Canvas with your name shown
- Open your IDE
- Show your code briefly
- Execute your code and demonstrate that your class works correctly.

Please submit the screencast as a **separate** file to Canvas.

Please do not modify any of the names/methods we've defined in the provided *.java files.