1) zWhy do we choose to implement add(Equipment e) instead of a more specific method like add(Guitar g)?

   Implementing the code this way allows for more flexibility in our code. If in the future we wish to also deal with, for example, microphones, as long as we implement them as a subclass of the Equipment class we won't have to change the code in our EquipmentInventory class to keep track of this other object, it can be dealt with in the same way as all the other Equipment objects.

2) Why do we use a String instead of an Equipment object in the HashMap?

   Strings, unlike Equipment objects, are immutable. This means our HashMap can run faster because it doesn't need to recheck the value of each string every time it's called. Being immutable, they also stop you from accidentally breaking your own code. Additionally using strings as keys is standard so if another developper was working on your code and you used a type other than string as your key it could make it more challenging for them to understand your code.

3) Why do we use an Integer instead of an int in the HashMap?

   Hashmaps treat both keys and values as objects. int is a primitive type and so will cause bugs if a Hashmap method tries to treat it as an object. Using a wrapper class, such as Integer, allows you to treat a primitive, such as int, as an object.

4) Why do we need to define a custom toString method for the Equipment types? What happens if we use the default method?

   A custom method allows each individual piece of equipment to print its own subclass name. From a default equipment method it would be more challenging to determine which subclass name to print because superclasses don't have access to subclass information. If you did find a way you would run into further challenges because you would have to modify this code if you added new types of equipment.

5) What happens if we only define a custom toString method for the Equipment or Instrument class (but not for any of the subclasses)?

   Without a unique toString method for each subclass you would need to implement some other getter to return the subclass name of each object if you wanted to keep track of specific objects in the inventory and inventoryCount objects. This would likely result in much more complex code.