ECE 325 Lab 3 Part 2

--Jaskeerat Singh

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

We choose to implement add(Equipment e) instead of a more specific method like add(Guitar g) so that we don't have to define methods for each type of equipment. For example, if we were to design more specific method, we would have to add it to each and every new type of equipment if added in the future. Moreover, we would have to write roughly the same code multiple times resulting in code dependency. By taking Equipment e as a parameter, it can be used on subclasses to Equipment as well, avoiding the above issues.

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

We have a string instead of an Equipment object in the HashMap because having a string in the HashMap explicitly tells us the type of the equipment instead of simply telling us the object. It gives a description which is useful to us when we count the number of items in each type and helps us classify and keep track of types of items in a more intuitive way.

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

We use an Integer instead of an int type in the HashMap because as mentioned in the lecture slides, we are not sure if the value of a primitive type like int is initialized to a number or if it has the default value. This makes things confusing while coding. However, this is not an issue with Integer, because Integer as an objects default is set to *null* making it easier to distinguish whether it exists in the HashMap or not. It allows us to see more clearly what is and is not existing in the inventory because by setting Integer to *null*, we can tell very easily whether it exists or not.

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

We need to define a custom toString method for the Equipment types to give a more intuitive representation of the type of the objects. By defining the toString method for the Equipment types, we are able to better control the way the objects are displayed when the method is invoked. If we were to use the default toString method, it would return the class name with the hash code of the object which would be confusing to deal with.

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

If we only define a custom toString method for the Equipment of the Instrument class and not for any of the subclasses, whenever we call the toString method, it will be invoked from the respective superclass. This would result in us not being able to distinguish the different types of Equipment/Instrument since if we called the function, it would just return the same string for all the subclasses. This is why overriding the method in each subclass is important.