Week5Lab – 10 pts

Pre-lab questions

1. Overloading methods can be challenging to understand. Write the following methods, all with the same method name:
    a. Write a method called address that takes in one string.

```
public static void address(String address){
    System.out.println(address);
}
```

    b. Write a method called address that takes in three strings, one for each line of the address.

```
public static void address(String street, String city, String state){
    System.out.println(street+", "+city+", "+state);
}
```

    c. Write a method called address that takes in three strings and an int, where the three strings represent each line of the address and the int is the 5-digit zip code.

```
public static void address(String street, String city, String state, int zip){
    System.out.println(street+" "+city+", "+state+", "+zip);
}
```

    d. Write method calls for each of the overloaded address methods

```
address("3784 Ocean Avenue, Los Angeles, CA, 90012");
address("3784 Ocean Avenue", "Los Angeles", "CA");
address("3784 Ocean Avenue", "Los Angeles", "CA",90012);
```

    e. When do you think this could be useful?

    This would be useful if you want to give the user flexibility in how they enter the address. They would not need to guess as much in how to enter the address or consult documentation when using the program. The compiler knows which method was used.

Choose one of the following to develop into a program that uses at least one class outside of the driver class that contains the main method, and an interface. Once chosen, do the following:

Understand the problem (restate in your own words, make any assumptions clear):

UML diagrams of any classes needed, including the one with main and the interface:

Pseudocode of any non-trivial methods in each class (no pseudocode needed for basic setters and getters or no args constructors):

Name of files (.java) submitted:

White box test plan and results for ONE of the classes:

Bodega

A small Bodega needs an inventory program to help them track their stock. The items they sell come in two categories: food and non-food. All items have a brand name, a price, a desired stock quantity, an actual quantity, and a location. Non-food items are taxed, but food items are not. Some food items are perishable and have a sell by date. They like to move items that are within one week of their sell by date to an impulse buy location near the register, so they need to be able to get a list of what is about to go out. They also want a list of items they have more in stock than the desired quantity, again, so they can put them up front. For this first pass include milk, paper towels, Oreos, Doritos, scissors, yogurt, canned beans and lighter. You can generate reasonable stock quantity and randomly generate actual quantity.

Band

The coach of the marching band wants to have a program that will help his students learn the basics of each instrument and prop. Ultimately this will be a GUI app with lots of recordings, but at this stage, you are just planning out how everything will work together. You can use text for the sounds instruments make as placeholders for the audio recordings. He wants all the musical instruments and the batons and flag included in this program. Only musical instruments make sounds, but all musical instruments make sounds. Each instrument and prop should have a name and a String representation. They will each belong to one of the following categories: prop, woodwind, brass, percussion. For this first pass, you need to have two instruments in each category and two props.

Interface Instrument:

```
playSound();
record(sound);
```

Class Woodwind:

```
def recordsound(String sound):
```

add custom sound/ note to recordings list (attribute of class)

Class Brass:

def playSound():

print sound

def recordsound(String sound):

add custom sound/ note to recordings list (attribute of class)

Class Percussion:

def playSound():

print sound

def recordsound(String sound):

add custom sound/ note to recordings list (attribute of class)

Class Prop:

def playSound():

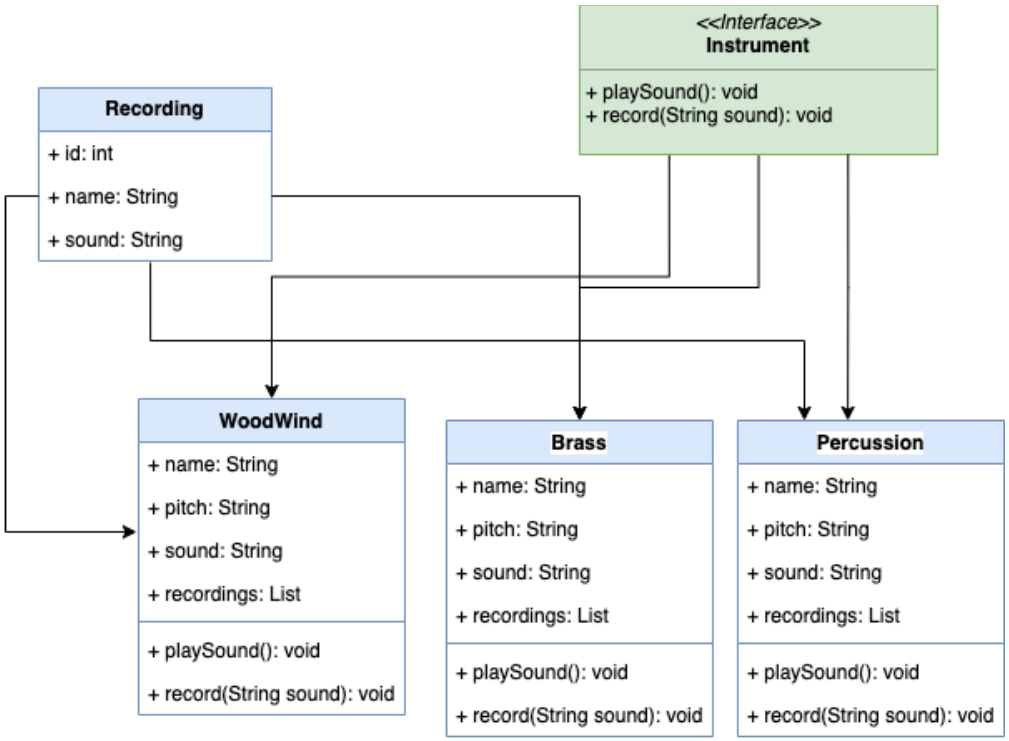
print sound

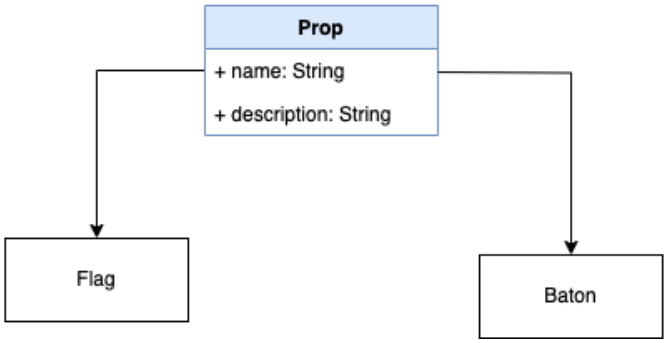
def recordsound(String sound):

add custom sound/ note to recordings list (attribute of class)

Class Recording:

This is a data class needed to convert a sound into an object and add it to the recordings list for each category of instruments


<<Interface>>
Instrument
+ playSound(): void
+ record(String sound): void
Recording
+ id: int
+ name: String
+ sound: String
WoodWind
+ name: String
+ pitch: String
+ sound: String
+ recordings: List
+ playSound(): void
+ record(String sound): void
Brass
+ name: String
+ pitch: String
+ sound: String
+ recordings: List
+ playSound(): void
+ record(String sound): void
Percussion
+ name: String
+ pitch: String
+ sound: String
+ recordings: List
+ playSound(): void
+ record(String sound): void


Prop
+ name: String
+ description: String
Flag
Baton

```
Played Sound: A
Played Sound: B#
Played Sound: C#
Woodwind{name: flute, pitch: high, sound: toot, recordings: 0}
Woodwind{name: clarinet, pitch: mid, sound: toot, recordings: 0}
Recording Added
Recording Added
Recording Added
Woodwind{name: flute, pitch: high, sound: toot, recordings: 2}
[Recording{id: 1, name: flute, sound: Bb}, Recording{id: 2, name: flute, sound: Ab}]
[Recording{id: 1, name: clarinet, sound: D}]
Recording Removed
Recording Not Found
[Recording{id: 1, name: flute, sound: Bb}]
```

Files submitted:

Prop.java

BandTester.java (driver file)

Brass.java

Woodwind.java

Percussion.java

Instrument.java (Interface)

Dungeon Crawl

Your friend has a great idea for a new dungeon crawler game but he's not sure if his ideas for weapon values can be done in a program. He wants to create several base weapons, such as dagger, sword, hammer, ax, mace, and halberd. Each will do a different amount of damage related to the common dice in D&D. The dagger would do 1-4, the sword 1-6, etc. The dice are 4d, 6d, 8d, 10d, 12d, and 20d. The complicating part is that many weapons can be enchanted, and enchantments add more randomness to the weapon values. Daggers and halberds cannot be enchanted. An enchanted weapon gets to add a second random number to the base weapon number. Some weapons that can be enchanted have an enchantment value of 0, they always return 0 to add to the weapons value. Some weapons that can be enchanted have an enchantment value that is a random number that gets added to the "roll" and others act as a second "roll" to be added to the weapons roll. A given enchantment may add an effect to the weapon. Each weapon should have a toString() that may return something like this: Flaming Sword, 8 or Thor's Hammer, 20. He wants to be able to create weapons with random enchantments and to create weapons and add specific enchantments.