Java 3.1: Understanding ArrayLists

1) In the `Cat` class
    a. Give it the following private instance variables
        i. `name` (String)
        ii. `weight` (double)
    b. Give it getters and setters for both `name` and `weight`.
    c. Give it a `meow()` method, which simply prints out "Meow!".

2) Create a new class called `TestCatList`. In the `main()` method:
    a. Create an ArrayList called `CatList` that contains `Cat` objects.
    b. Add four `Cat` objects to `CatList`
    c. Give each cat a unique name using the setter (pick your own cat names).
    d. Use an enhanced for loop to give each cat a random weight between 5 and 15.
        i. `Math.random()` returns a random double number between 0 and 1 (exclusive).
        ii. That number multiplied by 10 is a number between 0 and 10 (exclusive)
        iii. Adding 5 to part ii) returns a number between 5 and 15.
    e. Print out each cat's name and weight, and then have it meow on a separate line.

When finished, include both classes in one file and upload it to onCampus in the "ArrayLists: TestCatList" assignment.

<u>Bonus</u>

- Have a 25% chance of the cat making a different noise when meowing.
- Change the program so that the user can either name the cats themselves or have the program do it itself.
- Change the `meow()` method so that the cat makes a different sound based on its weight. Include three different sounds.

See the other side of this handout for help with syntax.

Useful syntax

Getters are methods with the explicit purpose of returning a private variable. They have the form

```
public vartype getVarname() {
      return varname;
}
```

Where `varname` is the name of the variable being returned, and `vartype` is its type.

Setters are methods with the explicit purpose of assigning a private variable. In the simplest case, they have the form

```
public void setVarname(newValue) {
      varname = newValue;
}
```

Where `newValue` is the value that the variable is being assigned to, and the other words have the same meaning as above.

To create an ArrayList, specify its type in angled brackets and use the `new` keyword to make a new ArrayList object:

```
ArrayList<Dog> dogList = new ArrayList<Dog>();
```

To add a new object to the ArrayList, use the `.add()` method:

```
dogList.add(new Dog());
```

This statement creates a new Dog object and places it in the list.

To access an element of the list, use the `.get()` method:

```
dogList.get(0);
```

This statement returns the zero-th element of the list, which is the Dog object we created above with new `Dog()`.

By accessing an element with `.get()`, you can use it just like you would any other variable, including accessing the object's getters & setters. If our `Dog` class has an instance variable called `Breed` with a corresponding setter, we can set the breed with this statement:

```
dogList.get(0).setBreed("Golden Retriever");
```

The enhanced for loop has the form

```
for (type elementName: listName) {
      // do something with elementName
}
```

Here `type` is the type of the array or ArrayList; `listName` is the variable name for your array/ArrayList; `elementName` is the variable name you use to access the element in `listName` (your array/ArrayList).