The purpose of this lab is to get more practice with setting up java classes, particularly writing *methods* and using *encapsulation.*

Note: You can use the same java project and package within the project as you did last week – or set up a *new* package within the same project. Or set up a new project and package. Up to you…

Part 1 – Explore Java API - 0.2

The Java API is a library of "free" java classes that do common things that save you writing all your code from scratch. The API for your version of java is available on the Oracle website.
e.g. The Java API (for Java version 15) is here

https://docs.oracle.com/en/java/javase/15/docs/api/index.html

Java classes are grouped into modules

Exercise:
Go the Java API page above
Find the **JButton** class
(Use the search bar on top right to help)

a. What **module** and **package** is it in?
b. What is the purpose of this class
c. How many "Constructors" does JButton have?

Find the **ArrayList class**

1. What **module** and **package** is it in?
2. How many constructors does it have?
3. Find the method is used to empty out all elements in the arraylist.

Part 2 – Java class – 0.4

Create a new class called Animal. Give it the following class members:

3 attributes in your Animal class - (1) name, (2) breed, (3) domesticAnimal (or not)

For now, DON'T mark the attributes as `private`.

Add 2 constructors in your Animal class

- One constructor that sets up just the animal name;
- Another constructor that sets up all 3 attributes.

a toString() method in your Animal class

As we did last week in the lab, add a toString() method to your Animal class that returns a String that contains the object data in a readable way e.g. "This animal is called Jerry, is a dog and it is true that it is a domestic animal"

Create another java class (call it Control) and put a main method in it. In this "main" method, instantiate the following Animal objects, including:

A domestic dog called Spot
An animal called "Leo";
… more of your own choice.

From the main method, print out each object your create (Using `System.out.println(objectname).`

PART 3 – Encapsulation — 0.6

In your "Main" method of your control class, print out the attribute values of any object you have created e.g.

```
System.out.println(whateveryourobjectnameis.name);
System.out.println(whateveryourobjectnameis.domestic);
```

UNLESS your attributes in your Animal class have been marked private, you'll be allowed to access these attributes of your object from another class, and to see their values. See if you can update them to new values. i.e. objectname.name = ".. whatever".

Now mark each attribute in your Animal class as private – e.g.

Eg. `Private String name ;`

Try printing the attributes again from the main method using:

```
System.out.println(whateveryourobjectnameis.name);
```

Marking the attributes as "private" in the class is the first step to *encapsulating* them.

PART 4 – getters and setters methods to support Encapsulation 0.8

- Write a method `public void setName (String name)` that allows the name attribute to be changed to the `name` value being sent in.
- Write a method called `getName` that returns the value of the name attribute as a String

From your main method, using an animal object you set up earlier – call your new getter method and setter methods to see and change the name attribute. e.g. set the name to "Sam", now check it is changed etc.. .

From your main method, using an animal object you set up earlier – call your new getter and setter methods to get and set each of the attributes values.

PART 5 – Using methods 1

In your animal class, add a method called makeNoise();

```
Public void makeNoise()
{

}
```

In it, just print out a noise… put in an if condition to do this. If the breed is a dog, print out a Bark, if it's a cat, "miaow" etc.. for at least 3 breeds.

Test your makeNoise() method by calling it for your animal objects in the main method.