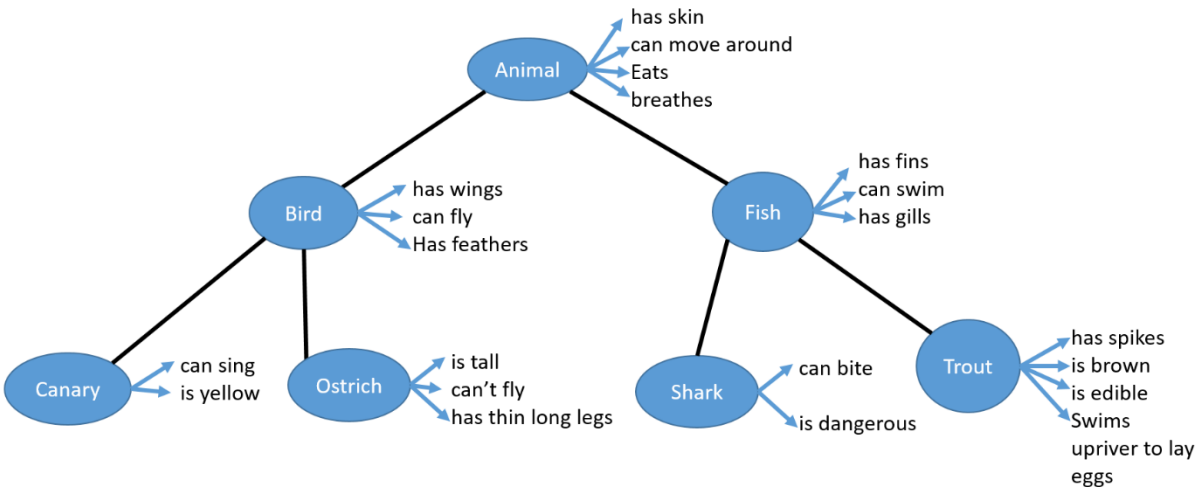
Assignment 3 – Inheritance

Instructions:

- Download the zip code attached.
- Create a new project in BlueJ.
- Add the classes in the zip file to the project
    - From the Blue J workbench menu select: Project-> Add Zip/Jar
- Answer the questions below - there are **two** parts to the assignment
- Submit your answer making sure to follow the submission instructions.

The attached code provides a Java implementation of the left-hand side of the hierarchy in the figure below:



Part 1: Implement the full animal class hierarchy

a) You are required to complete a Java implementation of the full inheritance hierarchy above making full use of what you know so far about inheritance. Each of the circles in the figure above represents a class. As in the example code I provided, some of the attributes of a class above are encoded in Java as fields and some as methods (e.g. I have chosen hasSkin to be a field and 'can move around' to be a method, move). You should make similar decisions with the remaining classes. Avoid unnecessary repetition of fields and methods. Inheritance is about inheriting fields and methods from the superclass and overriding them only when necessary. The example code provided illustrates how fields and method are inherited from superclasses and you should try to follow its example.

b) For every leaf class (Canary, Ostrich, Shark, Trout), I want you to implement an *equals* and *toString* method that correctly overrides the method inherited from the superclass. Every toString() method returns a String value describing the state of the object. Each of the toString methods that you implement should include values of fields inherited from the object's superclass. The example code contains a partially completed toString method for the Canary class. Remember to annotate all overriden methods with *@Overriden*, as demonstrated in the example code

c) For an Ostrich object, your code should implement the simplest way to get it to output "I am a bird but cannot fly" when it is called upon to move.

   Hint: This does not involve creating or overriding a new method. See lecture slides.

Part 2: Test the classes in your hierarchy

The second part of the assignment involves creating a class to demonstrate that the classes you have defined in Part 1 actually work. This is similar to the idea used in Assignment 2 where a class was used to test different transaction scenarios.

Create a class called *AnimalTest* with a main method and two test methods, which you will call from the main method. The first test method will demonstrate the output of the toString() method. The second test method will demonstrate that your equals() methods work as expected. Choose a suitable name for each method.

a) **test1 method**: this method will demonstrate that your toString() method works correctly. It should do this by populating an array with animal objects and then looping over the elements of the array and printing out the output of the toString() method of each element. If you are unsure how to use an Array, consult the tutorial notes on Arrays in the week 6 folder.

First of all create an Array of Animal references with size 4.

Animal[] animals = new Animal[4]; *// This declares and allocates memory for an array of Animal references with 4 spaces*

In each element of the Array I want you to put a reference to an object of the four leaf animals in your hierarchy (Canary, Ostrich, Shark, Trout). There are 4 spaces in the array, one for each type of animal.
As demonstrated in the lecture and in the Arrays tutorial PDF, each animal object (Canary, Ostrich, Shark, Trout) will be referenced by an element of the Animals array.

E.g.
animals[0] = new Canary("Bluey"); //*Assigning a reference to a Canary object to the first position in the array. Remember the first position in any array is always at index 0*

Now add a reference for each animal object to the remaining spaces in the Array. When finished you will have an array of four Animal references. Each reference points to an object belonging to an Animal subclass (e.g. Ostrich, Canary, Shark, Trout).
Then, write a loop and iterate over the array calling System.println to print out the contents of each element in the array (see the arrays tutorial, if you have difficulties).

E.g.
System.out.println(animals[i]); //*Where i is the value of the loop variable*

If you have written your toString() method correctly, the String value of the object at each position in the array will be output

Note: Do not call toString() directly as in this example:
System.out.println(animals[i].toString());

When you try to print an object out, Java will automatically look for its toString() method. So marks will be lost if you directly call the toString() method. This is never done.

b) **test2 method**: this method demonstrates that your equals methods work correctly. Use an array, like the one used in test1, and demonstrate the output of the different elements.

Note: An array is not necessary to demonstrate that equals works (or toString()) - however, I want you to get some practice in using Arrays.

I am leaving the specific implementation of this method for you to demonstrate.
Use comments to explain your code.

What to submit:

One PDF with the following code. Don't use a screenshot of the code; copy and paste it from BlueJ. As always, you should include comments and code description to demonstrate your understanding of the concepts. Add content in the following order to the PDF:

- Assignment Description
- AnimalTest code
- AnimalTest test1 output
- AnimalTest test2 output
- Canary code
- Ostrich code
- Fish code
- Shark code
- Trout code
- Brief explanation of your code - e.g. explaining how you got the Ostrich to printout "I am a bird but cannot fly". You are also required to upload each of the .java classes in your program