COP2800C Module 7 Graded Programming Assignment

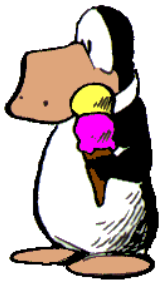
I recommend that you use the Horizon system for this assignment unless you are comfortable working with environment variables and execution search paths. All necessary configurations are already in place on the Horizon system.

Be sure to review the practice exercise for design notes and implementation details used in that exercise that will also apply to this assignment.

For this assignment we will incorporate inheritance and interfaces into our Penguin Rookery application design.

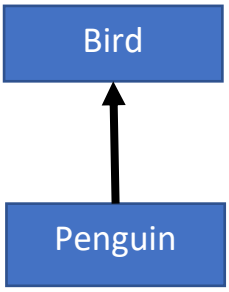
The data file for this assignment has not changed.

Our chief scientist has asked us to think about the possibility of creating a future rookery application which allows us to study data for other types of birds in addition to penguins (a rookery, according to Webster, can be loosely defined as "a breeding ground of gregarious birds"). After doing some research using The Google™ we have determined that penguins are indeed birds, but they are a specialized type of bird known as a "flightless waterfowl":



https://www.berkeleybreathed.com/

Based on this information, we can use the strong "is-a" relationship between birds (a generalization) and penguins (a specialization) to create the following inheritance hierarchy:



Look at the instance variables we have specified for our Penguin class. Which of these attributes could moved to a generalized Bird class? I've highlighted them here:

```
private Integer sampleNum = 0;
private PenguinSpecies species = PenguinSpecies.NO_SPECIES;
private Double culmenLen = 0.0,
```

```
                    culmenDepth = 0.0,
                    flipperLen = 0.0;
    private Integer bodyMass = 0;
    private Character sex = '\0';
```

Note that "flipperLen" is actually a wing length that we could generalize on as well, but we'll keep that in the Penguin class and continue to refer to it as a flipper measurement. We could generalize on a species of bird in the Bird class, but our application is referring specifically to the Palmer Penguin species so we will keep that in the Penguin class as well. "culmen" is actually a generic term for a bird's bill that is not specific to penguins so that can be moved to the Bird class, as can bodyMass, sex, and sampleNum.

For this assignment you will create a Bird class with the attributes specified above. Include an overloaded constructor, a toString method, an equals method, and accessors as appropriate.

Change the Penguin class to inherit from your Bird class and remove the accessors for any instance variables that have been moved into the Bird class. Modify the constructor, the toString method, and the equals method as necessary so that the superclass's methods are used as demonstrated in the practice exercise.

Create a "TableFormatter" interface in a separate package: edu.fscj.cop2800c.table and change the PenguinRookery class to specify that it implements that interface (note that these changes are identical to those made in the practice exercise). Remember that you will need to import the TableFormatter interface in the PenguinRookery class.

Modify the main method in the PenguinRookery class to test the toString and equals modifications as well as verifying the table method still works as before (this is known as "regression testing" – we need to make sure we did not break that method with our changes).

Create a JAR file for your project which contains all source code and class files.

Attach and submit the following three items:

- A screen snip showing the **top** lines of your command line output after running the JAR file. The snip should show your execution of the java command with the jar file and the subsequent visible output lines (enter a “cls” command before running to clear your screen and move the prompt to the top of the screen).
- A screen snip showing the **bottom** lines of your command line output after running your JAR file.
- Your executable JAR file containing your class files and your .java source code files. Do not submit your source code files separately, they must be contained in the JAR file.

The expected output should include a test of the Penguin's equals method results (testing == vs. the equals method), output from the PenguinRookery's toString method, and output from the table method. Note that the toString output and table output shown below has been compressed but your program should include all of the output.

Expected Output:

```
penguins p1 and p2 ==? false
penguins p1 and p2 equal? true
```

Our rookery contains the following penguins:

```
1,39.1,18.7,3750,M,181.0,ADELIE
2,39.5,17.4,3800,F,186.0,ADELIE
3,40.3,18.0,3250,F,195.0,ADELIE
...
340,50.4,15.7,5750,M,222.0,GENTOO
341,45.2,14.8,5200,F,212.0,GENTOO
342,49.9,16.1,5400,M,213.0,GENTOO
```

Palmer Penguin Rookery:

SampleNum	Species	CulmenLen	CulmenDepth	FlipperLen	BodyMass	Sex
1	ADELIE	39.1	18.7	181.0	3750	M
2	ADELIE	39.5	17.4	186.0	3800	F
3	ADELIE	40.3	18.0	195.0	3250	F
...						
340	GENTOO	50.4	15.7	222.0	5750	M
341	GENTOO	45.2	14.8	212.0	5200	F
342	GENTOO	49.9	16.1	213.0	5400	M