Tutorial 6

Objectives

Practice using polymorphism and abstract things.

Attendance Quiz

Please log on to cuLearn using one of the computers in the tutorial room and complete the attendance quiz. You can only access the quiz if you log in using one of the computers in the lab. You cannot use a laptop for this. This is a time limited quiz. Be sure to do this as soon as you arrive.

At the end of the tutorial a TA will assign you a grade, call it G, which is 0, 1 or 2, depending on the progress you make during the tutorial. If you spend your time reading mail and chatting with your friends you will receive 0. If you have completed the attendance quiz on time then G will be your tutorial grade. If you have not completed the attendance quiz on time, then your tutorial grade will be max(0, G - 1/2). Each tutorial grade will therefore be one of 0, 0.5, 1.5 or 2.

In order to receive full marks for this tutorial, you must fully complete parts 1 and 2, and make good progress into part 3.

---

0) Make a directory called **comp1406t6**. Download all the tutorial files to this directly. Run the **javadoc** program to generate the API for the classes.

1) Consider the *abstract* **Animal** class that is provided. Read the class (and the API) and see what it provides. Override Object's **toString()** method in the **Animal** class so that it prints out the animals name and *current age*. For the current age, just use the age in years that the animal will be this year. For example, if a dog was born in 2000 and it is currently 2019, the dog's current age is 19 years. The program should not hard-code the current year (see the Note below for help on this).

Note: assuming your computer's clock is correct, we can get the current year (as an int) using two classes of Java's **time** package as follows:

```
ZonedDateTime.now( ZoneId.of("America/Montreal")).getYear()
```

Note: *Animal* is abstract. You cannot create an *Animal* object using the *new Animal(...)*. In order to test your *Animal* class, you need to create a **concrete** child class that overrides all abstract methods in *Animal*.

---

2) The **Cat** and **Dog** classes should each **extend** the animal class. Add any needed constructors and methods to make these work. The classes should be ***concrete*** classes.

The noise that a cat makes should be *meow* or *prrr* (randomly chosen each time it's noise method is called), and the noise a dog makes should be *woof* or *grrrr* (randomly chosen each time it's noise method is called).

The **AnimalApp** program generates an array of animals and calls the noise method of each. You can use this to test your **Animal**, **Cat** and **Dog** classes.

3) Create an **Owl** class in the same package that also extends the **Animal** class. The class should have a three-argument constructor that takes a string (name), an integer (birth year) and a boolean (that determines if it is a *wise* owl or not). The string representation of an owl (i.e., the output of *toString()*) should include its name, age and whether or not it is a wise owl.

An owl's noise is *hoot*.

Modify the **AnimalApp** program so that some owl objects are also created.

4) Create two new subclasses for each of the **Cat**, **Dog** and **Owl** classes. For each pair of new classes, have one introduce a new noise that is specific to that specific kind of cat, dog or owl.