Program cs06 CS 5245: Data Programming
Fall 2022: Updated October 3, 2022 R. M. Parry

Read chapter 13 and 14 in Fundamentals of Python Programming and complete the following exercises.

Complete computer setup by following the directions here:
`http://cs.appstate.edu/~rmp/cs5245/setup.pdf`

Create a file called `cs06.py` to store your code for this lab.

This assignment comes with starter code available here:

`http://cs.appstate.edu/~rmp/cs5245/cs06.py`

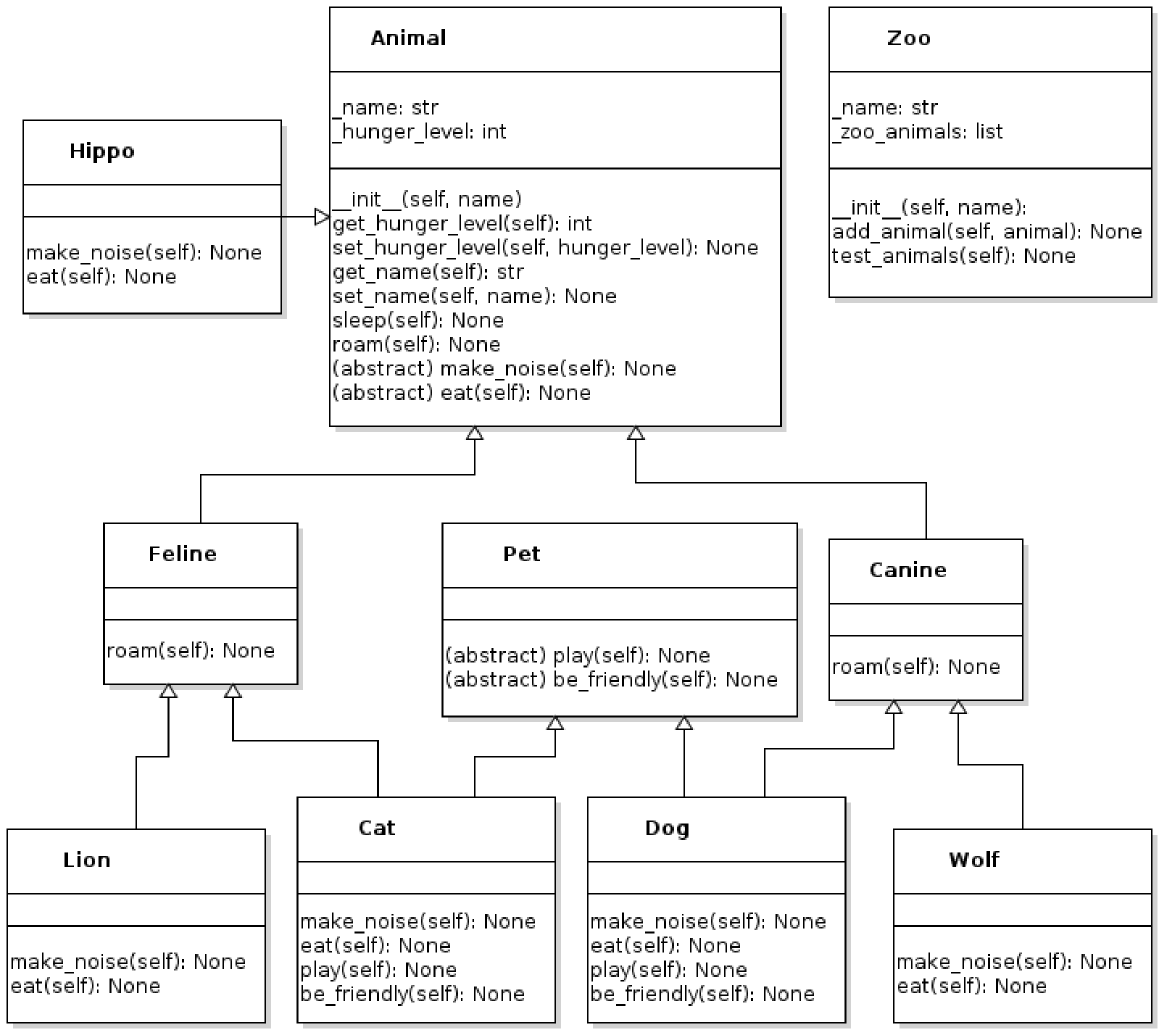
Complete the following activities using the **exact function names** and save the files using the **exact file names** as specified.

To test your programs, run them from IDLE and check their output.

1 Unified Modeling Language (UML) Class Diagram

Large projects need to coordinate class names, field names, method names, and inheritance (among other things). We will be using the diagram below for this assignment (click it to download the full version). Each box represents a `class` divided into three sections. The top section indicates the class's name. The second section lists any fields. (A leading underscore indicates that it is private.) The third section indicates the methods that the class implements (name, input arguments, and return type). An "abstract" method should raise an the `NotImplementedError`The information in the UML diagram specifies how these classes interact and can be used. However, it does not show how to implement it.

Understanding the class relationships allows you to create new classes that fit into the same software "ecosystem" and utilize or contribute to a larger project.



2 zoo

3 Animal class: Maintaining hunger_level

An Animal has a _name and a _hunger_level. Its _name should default to None. Each Animal's hunger level ranges from 0 (not hungry) to 10 (extremely hungry). "Get" and "set" methods should be provided in the Animal class. However, they do not make sure the hunger level stays in this range (between 0 and 10). Modify the code so that invalid hunger_levels get clamped to 0 or 10 (setting to -4 results in 0 and setting to 11 results in 10). When an animal sleeps its hunger_level should be set to 10. When an animal roams its hunger_level should increase by 1. The Animal class provides two abstract methods (make_noise and eat). These must be implemented by subclasses to avoid the NotImplementedError.

4 Other classes

- Pet provides two "abstract" methods play and be_friendly. These methods must be implemented in any subclasses to avoid the NotImplementedError.

- `Canine` and `Feline` overrides the `roam` method with a more specific print statement. Modify both methods so that they still increment hunger by one (like the `Animal` class).
- `Wolf` is a `Canine` that makes the noise: `'growl...'`. To eat, it should `'rip with teeth...'` and decreases hunger by 2.
- `Cat` is a `Feline` that makes the noise: `'meow...'`. To eat, it should `'pick...'` and decreases hunger by 3.
- `Dog` is a `Canine` that makes the noise: `'bark...'`. To eat, it should `'slop...'` and decrease hunger by 3.
- `Lion` is a `Feline` that makes the noise: `'roar...'`. To eat it should `'rip with teeth...'` and decrease hunger like a `Wolf`.
- `Hippo` makes the noise: `'blub...'`. To eat it should `'slurp...'` and decrease hunger by one.

5 Polymorphism

- `Zoo` has two fields: the name of the zoo (`_name`) and a `list` containing all the animals in the zoo (`_zoo_animals`). The `_name` should default to `None`. It has two methods: `add_animal` and `test_animals`. `add_animal` takes an `Animal` as an input argument and adds it to its `list` of animals. `test_animals` prints information about the zoo and simulates activities at the zoo:
  1. print the name of the zoo
  2. print the number of animals in the zoo
  3. for each `Animal` in the zoo,
     (a) print its name,
     (b) have it go to sleep,
     (c) they will be hungry, so have them make noise
     (d) eat until they are full
     (e) print the animal's hunger level
     (f) roam about
- Write a `main()` function that creates one animal of each type and adds them to a zoo. Then, call the zoo's `test_animals` method to see if it looks like the example.
- At the bottom of your Python file, you should call the `main` function as long as this is the "main" Python file being executed by the Python interpreter:

```python
if __name__ == '__main__':
    main()
```

Running your program should look something like this (possibly with different names and order of animals):

```
$ python p06.py
zoo name: zoo
number of animals: 5
name: dan the dog
sleeping...
bark...
slop...
slop...
slop...
slop...
hunger_level: 0
canines like to roam in packs...
-------------------
name: wanda the wolf
sleeping...
growl...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
hunger_level: 0
canines like to roam in packs...
-------------------
name: clyde cat
sleeping...
meow...
pick...
pick...
pick...
pick...
hunger_level: 0
felines like to roam alone...
-------------------
name: harry the hippo
sleeping...
blub...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
hunger_level: 0
moving around...
-------------------
name: lily the lion
sleeping...
roar...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
hunger_level: 0
felines like to roam alone...
-------------------
```

6 Interfaces

An interface is a class that merely specifies some (unimplemented) abstract methods. If a class inherits from it, it must implement these methods. The `Pet` class represents an interface. Modify your `Cat` and `Dog` class to inherit from `Pet` in addition to the class they already inherit from, and implement the extra `Pet` methods for them:

- Make sure the cats `'frolic...'` when they play and `'purr...'` when they are friendly.
- Dogs `'scamper...'` when they play and `'sniff...'` when they are friendly.
- Test this out in the `Zoos` `test_animals` method by having pets play and be friendly after roaming about. Because not all animals implement the `Pet` interface, you will have to check that an animal is a `Pet` before calling these functions using the built-in `isinstance` function.

After modifying your `test_animals` method, running your program should look something like this (possibly with different names and order of animals):

```
$ python p06.py
zoo name: zoo
number of animals: 5
name: dan the dog
sleeping...
bark...
slop...
slop...
slop...
slop...
hunger_level: 0
canines like to roam in packs...
scamper...
sniff...
-------------------
name: wanda the wolf
sleeping...
growl...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
hunger_level: 0
canines like to roam in packs...
-------------------
name: clyde cat
sleeping...
meow...
pick...
pick...
pick...
pick...
hunger_level: 0
felines like to roam alone...
frolic...
```

```
purr...
-------------------
name: harry the hippo
sleeping...
blub...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
slurp...
hunger_level: 0
moving around...
-------------------
name: lily the lion
sleeping...
roar...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
rip with teeth...
hunger_level: 0
felines like to roam alone...
-------------------
```

Submit to Web-CAT to compute your score!

1. Create a ZIP file for your `cs06` folder by right-clicking the folder and selecting:
   - **Send to** → **Compressed (zipped) folder** on Windows
   - **Compress Items** on MacOS
   - **Compress** on Linux

   You should find the new ZIP file in the same directory where `cs06` resides.

2. Login to `http://webcatvm.cs.appstate.edu:8080/Web-CAT` and submit your ZIP file for grading. You may submit as many times as you want before the deadline.