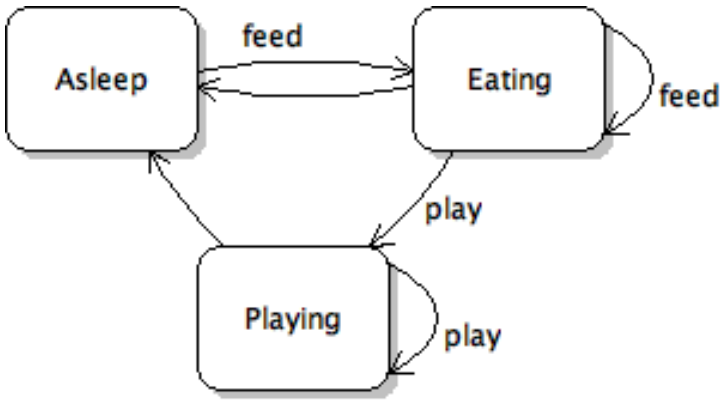
CECS 277 – Lab 14 – State

Puppy Simulator

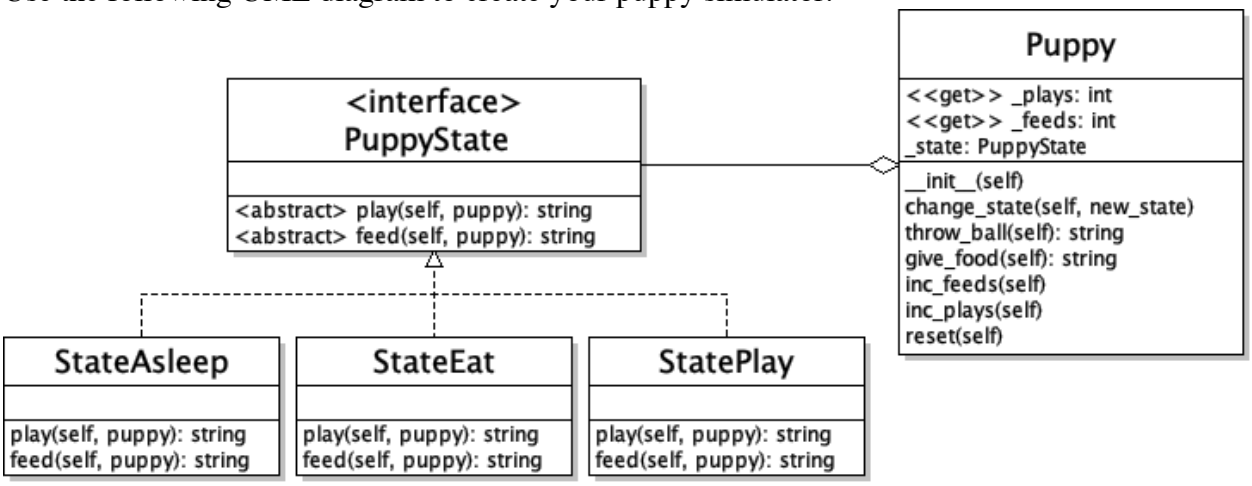
Using the State pattern, create a puppy simulator program that has two basic functions: to feed or play with the puppy. The puppy will react differently to these functions based on which state it is currently in. The puppy has three possible states: asleep, eating, or playing.

Use the following state diagram to decide how the puppy reacts to each function in each state:



- The puppy simulator should begin in the asleep state.
- When the puppy is asleep, the only way to wake it up is to feed it. It will come running when it hears its food bowl being filled.
- When the puppy is eating, it can continue to eat until it's so full that it will fall back asleep (after ~2 or 3 times), or if you can distract it with a ball, then it will play with you.
- When the puppy is playing, you can continue playing with it until it gets so tired that it falls asleep again (~2 or 3 times).
- You should not be able to play with puppy when it is asleep (it should continue sleeping) and you should not be able to feed the puppy when it is playing (it should ignore the food).

Use the following UML diagram to create your puppy simulator:



Classes:

1. Puppy – the object that the user interacts with.
    a. Attributes: _state, _feeds, _plays – add properties for feed and plays.
    b. __init__(self) – initializes the state to the asleep state, and then initializes the number of feeds and plays.
    c. properties for feed and plays.
    d. change_state(self, new_state) – updates the puppy's state to the new state.
    e. throw_ball(self) – calls the play method for whichever state the puppy is in.
    f. give_food(self) – calls the feed method for whichever state the puppy is in.
    g. inc_feeds(self) – increments the number of times the puppy has been fed in a row.
    h. inc_plays(self) – increments the number of times the puppy has played in a row.
    i. reset(self) – reinitializes the feeds and plays attributes.
2. PuppyState – interface
    a. feed(self, puppy) – abstract (no code)
    b. play(self, puppy) – abstract (no code)
3. The Three Concrete States: StateAsleep, StatePlay, StateEat –
    a. feed(self, puppy) – use the state diagram to implement the puppy's reaction to feeding according to which state class you're writing. Returns a string describing the puppy's reaction.
    b. play(self, puppy) – use the state diagram to implement the puppy's reaction to playing according to which state class you're writing. Returns a string describing the puppy's reaction.
4. Main – construct a puppy object and then display a menu that allows the user to play with or feed the puppy. Display the puppy's reaction to the user's choice. Repeat until the user chooses to quit.

Example Output:

```
Congratulations on your new puppy!
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 2

The puppy is asleep.  It doesn't
want to play right now.
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 1

The puppy wakes up and comes
running to eat.
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 1

The puppy continues to eat as you
add another scoop of kibble to its
bowl.
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 1

The puppy continues to eat as you
add another scoop of kibble to its
bowl.
The puppy at so much it fell
asleep!
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 1

The puppy wakes up and comes
running to eat.
```

```
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 2

The puppy looks up from its food
and chases the ball you threw.
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 1

The puppy is too busy playing with
the ball to eat right now.
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 2

You throw the ball again and the
puppy excitedly chases it.
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 2

You throw the ball again and the
puppy excitedly chases it.
The puppy played so much it fell
asleep!
What would you like to do?
1. Feed the puppy
2. Play with the puppy
3. Quit
Enter choice: 3
```

Notes:

1. You should have 6 different files: main.py, puppy.py, puppy_state.py, state_eat.py, state_play.py, and state_asleep.py
2. Check all user input using the get_int_range function in the check_input module.
3. Do not create any extra methods, attributes, functions, parameters, etc.
4. Please do not create any global variables or use any of the attributes globally (ie. do not access any of the attributes using the underscores).
5. Use docstrings to document each of the classes, their attributes, and their methods.
6. Place your names, the date, and a brief description of the program in a comment block at the top of your main file. Place brief comments throughout your code.
7. Thoroughly test your program before submitting:
    a. Make sure your puppy starts in the asleep state.
    b. Make sure it can only be woken up if it is fed.
    c. Make sure that it falls back asleep if it is fed too much.
    d. Make sure that it can only move to the play state from the eating state.
    e. Make sure that it cannot go back to the eating state from the play state.
    f. Make sure that it if it is played with several times, it falls asleep.
    g. Make sure that the user can quit the program.

Puppy Simulator – Time estimate: 4 hours

Puppy Simulator **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
PuppyState Interface (separate file)**:** 1. Inherits from ABC 2. Has abstract methods. 3. Has feed and play methods.					
Concrete States (separate files)**:** 1. Has 3 states: Asleep, Play, Eating. 2. Each inherit from PuppyState. 3. Each correctly override feed/play. 4. feed and play return a string descr. 5. Changes state according to state machine diagram. 6. Increments and resets counters correctly.					
Puppy Class (separate file)**:** 1. Initializes puppy in Asleep state. 2. Calls feed and play on the current state. 3. Correctly increments and resets the puppy's counters.					
Main file (in separate file)**:** 1. Constructs Puppy object. 2. Has a repeating menu to interact with the puppy. 3. Displays correct output based on Puppy's current state. 4. Puppy reacts correctly to user's selections.					
Code Formatting: 1. Correct documentation. 2. Meaningful variable names. 3. No exceptions thrown. 4. No global variables or accessing attributes directly. 5. Correct spacing.					