Name: Soriano, Dexter G. 7OOP
BSCS - C204

Finals Task 3. Simple Polymorphism

Problem. Chirp and Tweet

Create a simple program to demonstrate basic polymorphism with bird sounds.

Class - Bird:

- Methods:
  - `def make_sound(self) -> None`: An abstract method that represents making a sound. It doesn't have a specific implementation in the base class `Bird`.

Class - Sparrow (extends Bird):

- Methods:
  - `def make_sound(self) -> None`: Overrides the `make_sound` method from the base class `Bird`. It prints the sound "Chirp Chirp" when called.

Class - Parrot (extends Bird):

- Methods:
  - `def make_sound(self) -> None`: Overrides the `make_sound` method from the base class `Bird`. It prints the sound "Tweet Tweet" when called.

Class - BirdCage:

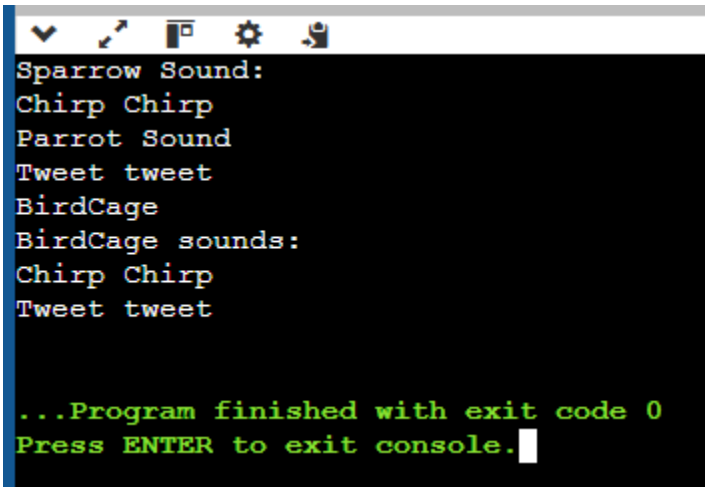
- Methods:
  - `def make_bird_sounds(self, birds: List) -> None`: Accepts a list of `Bird` objects as input. Iterates through the list of birds and calls the `make_sound` method on each bird to make its sound.

Note:

- *The test cases are not outputs of your main file but of a hidden test file. Create and implement the classes instructed to test your code.*
- *Each class should be defined in its own file, with the file name following camelCase conventions (e.g., `bankAccount.py`).*

TEST CASES:

SAMPLE OUTPUT:



SAMPLE CODE:

main.py | bird.py | sparrow.py | parrot.py | birdCage.py

```python
from sparrow import Sparrow
from parrot import Parrot
from birdCage import BirdCage

def main():
    sparrow = Sparrow()
    parrot = Parrot()
    print("Sparrow Sound:")
    sparrow.make_sound()
    print("Parrot Sound")
    parrot.make_sound()

    print("BirdCage")
    cage = BirdCage()
    birds = [sparrow,parrot]

    print("BirdCage sounds:")
    cage.make_bird_sounds(birds)

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

main.py | bird.py | sparrow.py | parrot.py | birdCage.py

```python
from abc import ABC, abstractmethod

class Bird:
    @abstractmethod
    def make_sound(self) -> None:
        pass
```

main.py | bird.py | sparrow.py | parrot.py | birdCage.py

```python
from bird import Bird

class Sparrow(Bird):
    def make_sound(self) -> None:
        print("Chirp Chirp")
```

main.py | bird.py | sparrow.py | parrot.py | birdCage.py

```python
from bird import Bird

class Parrot(Bird):
    def make_sound(self) -> None:
        print("Tweet tweet")

```

main.py | bird.py | sparrow.py | parrot.py | birdCage.py

```python
from bird import Bird
from typing import List

class BirdCage(Bird):
    def make_bird_sounds(self, birds: List[Bird]) -> None:
        for bird in birds:
            bird.make_sound()

```